Verifiable SelfMix

Doron Zarchy

APSIA, SnT, University of Luxembourg, Luxembourg
doronz@gmail.com

Abstract. Anonymous communication systems aim to hide which user sent which message. Existing designs span efficient mixnets that rely on at least one honest mix server and decentralized protocols such as Dining Cryptographers networks (DC-nets) or secure multi-party computation (MPC)-based shuffles, which typically require greater communication or interaction. We introduce *verifiable self-mix* (VSM), an anonymity architecture for privately placing messages in a public bulletin-board table. VSM separates oblivious slot allocation from anonymous message placement: *Unique Number Selection* (UNS) assigns each user a distinct hidden location, and *Secure Mapping of Private Permutation* (SMPP) places each encrypted message at its assigned location without revealing the user-to-location mapping. Because each user learns their own final location, VSM provides unconditional individual verifiability after the table is decrypted. We define VSM and prove anonymity, integrity, and self-verifiability in a static malicious model. We instantiate UNS using either trusted hardware or multi-server plaintext-equivalence tests, and SMPP using ElGamal, Boneh–Goh–Nissim (BGN), and a theoretical fully homomorphic encryption (FHE) construction. For n users and m slots, the vector based SMPP constructions require $O(m)$ ciphertext upload per user and $O(nm)$ public aggregation. We also present an FHE based variant that reduces the client upload to $\tilde{O}(\log m)$ for fixed size messages. These constructions offer different tradeoffs between trust, communication, and computation, while preserving the modular structure of VSM and its unconditional individual verifiability.

1 Introduction

Anonymous communication networks (ACNs) aim to protect message senders' identities against a powerful network observer. Mixnets are the most widely used ACN primitive: a set of mix servers repeatedly permute and re-encrypt batches of ciphertexts so that an outside adversary cannot link individual inputs to outputs [12, 23, 35, 39, 1, 41]. While mixnets are efficient and well-understood, they are typically centralized (C-ACNs): anonymity relies on the assumption that at least one mix server in the chain does not collude with the adversary. This assumption is challenging to justify in practice, especially when mixes are operated by a single organization or deployed in low-assurance environments, as illustrated by the Norwegian trial Internet voting system [24].

Trusted hardware (TH), such as Intel Software Guard Extensions (SGX), is often proposed as a way to harden a single mixing server: cryptographic state is

kept inside an isolated enclave that can be remotely attested. However, current trusted execution environments (TEEs) offer only limited private memory (e.g., SGX enclaves expose about 128 MB of protected memory and incur substantial overhead when paging) [40]. Mixnets conceptually hold and shuffle entire batches of ciphertexts, so they tend to require substantial memory for global permutations. This tension makes it difficult to deploy large-scale mix-based ACNs or e-voting systems directly inside TEEs.

A different line of work removes the trusted mixer entirely and instead builds fully decentralized ACNs (D-ACNs), often via secure multi-party computation (MPC). The classic example is the dining cryptographers network (DC-net) [10], along with many jamming-resistant variants and MPC-based anonymous broadcast systems. While these protocols do not assume a trusted third party, they have well-known drawbacks: the number of rounds often grows with the number of participants or adversaries, honest users may send $\Omega(n)$-$\Omega(n^2)$ messages per round, and even a single malicious participant can disrupt the protocol unless heavy accountability mechanisms and zero-knowledge proofs are added [32]. Multi-party sorting-based D-ACNs can improve round complexity [32], but typically require a strong honest-majority assumption among the participants.

Our goal is to explore a different point in this design space: an ACN architecture that (i) keeps the trusted component small enough to execute in a trusted execution environment or a small set of threshold servers, (ii) gives users a simple, non-cryptographic way to verify that their messages were included correctly, and (iii) allows users to submit their encrypted messages without interacting with one another after the preprocessing phase. We analyze the communication, computation, and trust costs of VSM, thereby making explicit the tradeoffs introduced by its modular design and individual verifiability.

Verifiable Self-Mix (VSM). We introduce *verifiable self-mix* (VSM), a semi-centralized ACN architecture for privately shuffling messages over a public bulletin board. Instead of having a mix server permute an entire batch, VSM lets each user "self-mix" their encrypted message by writing it to a *secret* location in a public table. Intuitively, the system assigns each user an anonymous "slot" in the table, and users then place their ciphertexts into their own slots without revealing which slot they control.

VSM consists of two generic modules:

- **Unique Number Selection (UNS).** A preprocessing protocol that runs once per epoch to assign every user a distinct *mix location* λ_i in an enlarged index space. The value λ_i is hidden from all other parties but learned by user i. Our UNS constructions use either a TEE with a misuse-detection mechanism, or a set of servers that jointly run plaintext-equivalence tests (PETs); in both cases, the trusted component operates only on encrypted indices and needs $O(1)$ private memory, making it suitable for constrained trusted hardware.
- **Secure Mapping of Private Permutation (SMPP).** Given encrypted pairs $(\lambda_i, E(m_i))$ from n users, SMPP produces a public table T such that,

for each user i, the entry at position λ_i is an encryption of m_i, and all other entries contain encryptions of the neutral value. We present SMPP constructions based on ElGamal and Boneh–Goh–Nissim (BGN), using either non-interactive zero-knowledge proofs (NIZKs) or homomorphic checks to ensure that no one learns the mapping from users to locations.

A key property of VSM is that *each user knows their own final location.* If we additionally decrypt the entries of T (SMPP-Dec), every user can *unconditionally* verify that their message appears in the correct slot λ_i simply by checking the published table—no cryptographic tools or hardness assumptions are required on the user side. This form of individual verifiability is particularly attractive for applications such as e-voting, petitions, and anonymous feedback systems, where participants want both anonymity and a straightforward way to confirm that their inputs were not altered or dropped.

Contributions. In summary, this paper makes the following contributions:

- **A formal VSM architecture.** We define verifiable self-mix (VSM) as the composition of UNS and SMPP, and give security definitions and proofs for anonymity, integrity, and self-verifiability in the malicious model.
- **TEE-friendly unique number selection.** We design UNS protocols that assign unique encrypted locations to users while using only constant private memory on the trusted component. We show how UNS can be instantiated either in a TEE with misuse detection, or by multiple servers running PET with a threshold honest-server assumption.
- **Secure mapping of private permutations.** We introduce SMPP as a reusable primitive and give an ElGamal construction using NIZKs and a BGN construction using homomorphic checks. We also sketch a fully homomorphic encryption (FHE) variant that reduces fixed-payload client upload to $\tilde{O}(\log m)$, while retaining $O(m)$ routing work and requiring FHE evaluation and consistency proofs.
- **Analysis and comparison with existing ACNs.** We prove that VSM achieves anonymity and integrity under standard cryptographic assumptions, and that SMPP-Dec provides unconditional individual verifiability. We analyze the communication and computation costs of the proposed instantiations. For n users and m slots, the vector based SMPP constructions require $O(m)$ ciphertext upload per user and $O(nm)$ public aggregation. The FHE based construction reduces the client upload to $\tilde{O}(\log m)$ for fixed size messages, while its routing computation remains linear in m.

2 Related Work

Mixnets can be traced back to the pioneering work of Chaum [12] which was based on a decryption mix. Park et al. were the first to propose a construction based on homomorphic properties of re-encryption [36]. It was further developed by [23, 35, 39, 1]. Universally verifiable mixnets based on zero-knowledge shuffle

proofs were later developed by Neff [34] and Groth [26]. Wikstrom proposed a new method for mixnets based on partial decryption and proved its security via universal-composability framework [41]. Adida et al. and Wikstrom proposed methods for public shuffling where the permutation is computed before the actual shuffle [3, 4]. Chaum et al. proposed cMix, a mixnet variant that minimizes real-time public-key operations by moving most expensive work to preprocessing [11].

A major line of works on ACN starts with the DC-net [10], and subsequent works mostly focus on detecting and preventing cheating parties from jamming the broadcasts [25, 16, 38]. Dunning and Kresman propose a novel technique hiding the message using Newton identities [20], though the scheme is secure only in the honest but curious setting. DiceMix [38] adapts the Newton-identity technique of Dunning and Kresman [20] to a peer-to-peer mixing setting and achieves security in the malicious model. Many DC-net style protocols, including Dissent and DiceMix, require a number of communication rounds that grows linearly with the number of disrupted messages or misbehaving users in order to identify jammers [16, 38]. More recent systems such as PriFi and OrgAn achieve constant-round anonymity by adding additional trust assumptions (e.g., access points or coordinators) and precomputation phases [7, 19].

Another approach is based on multi-party sorting where the sequence of comparisons (for sorting) is set in advance, regardless of the outcome of previous comparisons [5, 32, 30, 9, 27]. These schemes are based on data-oblivious algorithms and verifiable secret-sharing schemes (VSS). These protocols are efficient but assume that the number of adversaries has to be less than one-third of the parties [32, 30, 9].

SMPP vs. DC-nets. At first glance, SMPP resembles a one-round DC-net because both aggregate per-user contributions into a combined output. In many DC-net constructions, users maintain pairwise pads or related shared state whose contributions cancel in the aggregate, and active disruption may require a shared blame phase [16, 38]. SMPP instead uses public-key ciphertexts. Each submission is independently verifiable, so malformed inputs can be rejected without a shared disruption-resolution round; users also maintain no pairwise secrets. This does not imply uniformly lower total work: vector-based SMPP requires $O(m)$ client upload and $O(nm)$ public aggregation. Its contribution is therefore a different trust, interaction, and verifiability tradeoff rather than general asymptotic dominance over DC-net systems.

On evaluation scope. End-to-end performance comparisons across anonymous communication systems are difficult because latency and throughput depend strongly on deployment assumptions beyond the cryptographic core [18]. Accordingly, we report an analytic cost breakdown of VSM's modules in terms of client upload, bulletin-board storage, and dominant cryptographic operations; the bulletin-board realization cost is outside our scope.

Detailed VSM operation counts appear in Table 2.

Table 1. Coarse comparison of representative ACN families and VSM variants. Deployment-specific costs, churn, cover traffic, and bulletin-board realization are excluded.

System family	**Online client upload**	**Online interaction rounds**	**Additional trust assumption**
Classic mixnets	compact ciphertext(s)	mix cascade / batching	at least one honest mix server
DC-net style systems	protocol-dependent, often $\Omega(n)$	protocol-dependent	depends on disruption model and assisting-server architecture
VSM (TEE-UNS)	$O(m)$ for SMPP	one bulletin-board round after UNS	one trusted TEE (or its vendor root of trust)
VSM (PET-UNS)	$O(m)$ for SMPP	one bulletin-board round after UNS	at least one honest server among the PET servers
VSM (FHE-SMPP)	$\tilde{O}(\log m)$ in theory	one bulletin-board round after UNS	same as the chosen UNS instantiation

Multi-server anonymous broadcast and metadata hiding. Riposte and Express use distributed point functions (DPFs) and non-colluding servers for anonymous writes [15, 22]. Atom and Trellis provide scalable mix-based anonymous communication [28, 29], while Blinder and Clarion use MPC or multiparty shuffling [2, 21]. VSM similarly assumes a small server set and an authenticated append-only board, but differs in giving each user a private final slot and therefore an unconditional individual inclusion check.

Low-latency mixnets and DC-net-style group communication. Loopix [37] uses cover traffic and Poisson delays to resist traffic analysis. Dissent and Verdict provide accountable anonymous group messaging on DC-nets [16, 17], while PriFi and OrgAn adapt DC-nets to organizational networks via a coordinator preprocessing phase reused across rounds [7, 19]. In comparison, VSM uses an epochal preprocessing step that must be rerun when cross-epoch unlinkability is required (Section 8); otherwise slot reuse yields pseudonymous rather than freshly anonymous positions.

3 System Overview

Suppose that n users have private input and wish to write on the public database while preserving their anonymity. In contrast to a mixnet, where the server shuffles the messages, here each user *self-mixes*: it autonomously selects a distinct slot in an m-entry table and writes its message there. There are two challenges regarding this approach. First, there may be collisions, i.e., when more than one user selects the same location. Second, the users may lose their anonymity when performing the actual writing on the bulletin board. The bulletin board is trusted only for availability and append-only delivery: it need not be trusted for privacy, and any standard authenticated broadcast or publish primitive suffices. Practical instantiations include public ledgers, server-maintained append-only logs with

digital signatures, or transparency-log infrastructure; the cost of maintaining this abstraction is outside our scope. VSM addresses these two challenges.

The general architecture of VSM is built on two decentralized modules (i.e., sub-protocols): unique number selection (UNS) and Secure Mapping of Private Permutation (SMPP).

1. UNS is a preprocessing phase executed once per *epoch* before the users send a batch of messages. The goal of UNS is to assign a unique number to each user. These numbers are called mix-locations and should be kept private. One mode of UNS is suitable for a trusted execution environment and the other is designed to be executed by multiple parties.
2. SMPP is an anonymization module that allows users to place an encrypted message into the table entry indexed by their UNS-assigned mix location, without revealing which user controls that location. We provide two instantiations based on two encryption schemes. The first is ElGamal-based, which does not depend on any trusted setup beyond the bulletin board and threshold decryption, though it requires relatively more expensive zero-knowledge proofs; the second is BGN-based, whose privacy depends on an honest-majority setup MPC, though it avoids per-user NIZKs.

Participation failures. UNS is executed at the beginning of each epoch for the users registered for that epoch. Users that do not submit their UNS inputs by the submission deadline are excluded before UNS is executed. After UNS, only users that submit valid SMPP inputs by the message deadline participate in SMPP. A user that completes UNS but does not submit a valid SMPP input is excluded from the SMPP execution, and its assigned location remains unused. This does not require UNS to be repeated, since removing a user preserves the distinctness of the locations assigned to the remaining users. Users that join after the epoch begins participate in the next epoch.

If the same mix-locations are reused across multiple message batches, the resulting outputs remain anonymous with respect to user identity but become linkable to the same pseudonymous slot. Formal unlinkability across epochs therefore requires rerunning UNS for each epoch.

4 Preliminaries

We use the notation $[n] = \{1, \ldots, n\}$ and write $\hat{x} := E(x)$ for an encryption of x under the relevant public key. For a vector $v = (v_1, \ldots, v_m)$ we write $\hat{v} := (E(v_1), \ldots, E(v_m))$. All communication to the bulletin board is authenticated, and the bulletin board is assumed public and append-only.

We write $\tilde{O}(f)$ to suppress polylogarithmic factors in f and in the security parameter. Our constructions use standard public-key tools. The ElGamal-based instantiation uses threshold ElGamal for additively homomorphic combination and rerandomization together with NIZKs for proving that encrypted vectors are well formed. The BGN-based instantiation uses the Boneh-Goh-Nissim cryptosystem, which supports arbitrarily many additions and one multiplication via

pairings, so that some of the ElGamal proofs can be replaced by homomorphic checks. When servers must compare encrypted candidates during UNS, they either execute a plaintext-equivalence test (PET) over threshold ElGamal ciphertexts or rely on a trusted enclave that performs equality testing on decrypted values.

The VSM architecture is designed for an authenticated bulletin-board setting. We therefore separate three kinds of cost throughout the paper: (i) client upload to the bulletin board, (ii) public aggregation work performed by the servers or any third party reading the board, and (iii) the final threshold decryption cost when the aggregated table is opened. This level of abstraction is better suited to VSM than end-to-end latency claims, which depend heavily on the surrounding system architecture.

5 Unique Number Selection

Unique Number Selection (UNS) is the preprocessing phase of VSM. Its goal is to assign each surviving user a distinct hidden slot (also called a *mix integer*) from an expanded location space $[m]$, where $m = a \cdot n$ for a fixed constant $a > 1$. User i samples $K = O(\log n)$ candidate slots $\ell_{i,1}, \ldots, \ell_{i,K} \in [m]$ and encrypts them under the servers' public key, obtaining

$$\hat{P}_i = (E_{pk}(\ell_{i,1}), \ldots, E_{pk}(\ell_{i,K})).$$

UNS receives $\hat{P} = (\hat{P}_1, \ldots, \hat{P}_n)$ and outputs encrypted slots $(\hat{\lambda}_1, \ldots, \hat{\lambda}_n)$ for the surviving users, such that no two surviving users receive the same slot. Each user recovers its slot in plaintext by identifying which of its posted ciphertexts appears in the public output.

During execution, the servers maintain for each user a pointer to the *current* candidate. Whenever two current candidates collide, both users advance to their next candidates. If a user's list is exhausted, the user is removed. This prevents malicious users from forcing non-termination by repeatedly colliding with others.

5.1 Performance of UNS

We measure the runtime of UNS by the number of collision tests TC.

Lemma 1. *The expected number of collision events (while-loop iterations that result in a collision) in Protocol 1 is at most* $n\frac{a}{a-2}$.

Proof. We model the process as a balls-into-bins walk. Each currently active candidate is a ball and the $m = an$ possible slots are bins. A collision corresponds to drawing an occupied bin, in which case both colliding users advance and the number of unresolved candidates does not decrease; a non-collision corresponds to placing a ball in an empty bin, thereby resolving one candidate. Since at most n bins can be occupied at any time, the probability of collision is at most $1/a$, while the probability of non-collision is at least $(a-1)/a$. Thus the expected drift toward termination is at least $(a-2)/a$ per test, and the expected number of collision events needed before all n users are resolved is at most $n\frac{a}{a-2}$.

Protocol 1 Unique Mix Integers $(\hat{P}_1, \ldots, \hat{P}_n)$

Servers:

1. For each user i, initialize a pointer $SI[i] \leftarrow 1$ and current candidate $S[i] \leftarrow \hat{P}_i[1]$.
2. Initialize the active queue $S_A \leftarrow \{1, \ldots, n\}$.
3. While $S_A \neq \emptyset$:
   (a) Pop some user index i from S_A.
   (b) For each active user $j \neq i$, compute $TC(S[i], S[j])$.
   (c) If a collision is found between i and j, increment both pointers, update the corresponding current candidates, and push the updated users back to S_A.
   (d) If a pointer exceeds K, remove that user from the protocol.
4. If all users are removed, return FAIL.
5. Otherwise return the surviving encrypted candidates $(S[i])_i$ together with the surviving pointers SI. Each surviving user i recovers λ_i in plaintext as the candidate $\ell_{i,SI[i]}$ it originally submitted, identified by observing which of its ciphertexts appears in the public output.

For the concrete choice $a = 4$, the expected number of collision events is at most $2n$.

Remark 1. The lemma bounds collision events, not total TC invocations. Each while-loop iteration tests all active users $j \neq i$, so the total number of TC invocations is $O(n^2)$ in the worst case. The cap $K = O(\log n)$ limits how many times any single user can be involved in a collision, bounding the maximum disruption by corrupted users.

Corrupted users can force at most $K \cdot |\mathsf{corrupted}|$ additional candidate advances before their lists are exhausted and they are removed. Each advance may require comparisons against up to $n-1$ active users, consistent with the $O(n^2)$ worst-case collision-test bound.

5.2 Handling Denial of Service

If preference lists were unbounded, colluding users could repeatedly submit colliding values and prevent termination. We therefore cap each list at $K = \lceil \log n \rceil$ candidates and remove a user whose entire list is consumed. The following lemma shows that an honest user is removed only with negligible probability.

Lemma 2. *Assume $m \geq 4n$ and each user submits $K = \lceil \log n \rceil$ independently sampled candidates. Then the probability that Protocol 1 consumes the entire list of at least one honest user is at most $n \cdot (1/4)^K$.*

Proof. Fix an honest user i with candidates $\ell_{i,1}, \ldots, \ell_{i,K}$. For $t \in [K]$, let C_t be the event that the t-th candidate collides when it becomes active. Condition on any protocol history up to the moment that $\ell_{i,t}$ is checked. At that time, there are at most $n - 1$ active candidates belonging to other users. Since the honest

preference lists are encrypted before UNS begins, a computationally bounded adversary cannot choose those active values as a function of $\ell_{i,t}$. As $\ell_{i,t}$ is independently uniform over $[m]$,

$$\Pr[C_t \mid \text{history}] \leq \frac{n-1}{m} \leq \frac{1}{4}.$$

For user i to be removed, all K candidates must collide. By the chain rule,

$$\Pr[\text{user } i \text{ removed}] = \prod_{t=1}^{K} \Pr[C_t \mid C_1, \ldots, C_{t-1}] \leq (1/4)^K.$$

Applying the union bound over all honest users gives the claim.

Throughout the paper we set the expansion factor to $a = 4$, which satisfies both the condition of Lemma 1 and the condition of Lemma 2. For intuition, when $n = 1000$ the resulting upper bound on accidental removal of an honest user is below 1/439,000.

5.3 Testing Collision

The collision-testing primitive TC can be instantiated either by trusted hardware or by a small set of servers.

TC via Trusted Hardware. A TEE can decrypt two ciphertexts and test equality of the underlying plaintexts. To prevent a malicious host from probing arbitrary ciphertext pairs, the enclave maintains a signed hash chain over all invocations: on input (c_0, c_1) it decrypts, outputs the equality bit t, and updates $S \leftarrow H(S, c_0, c_1, t)$; upon termination it signs the final state and halts. Because the next legal UNS query is determined by the public transcript, any unauthorized query changes the signed state and is detected. Only the secret key and the running state reside inside the enclave, so the private TEE memory is $O(1)$.

Lemma 3. *No malicious TEE host can gain information by misusing TC without being detected.*

Proof. Any deviation from the publicly determined sequence of legal TC invocations changes the enclave's signed hash state. By collision resistance of H and the enclave's refusal to process further queries after termination, such misuse is detected except with negligible probability.

TC executed by servers. Alternatively, k servers run a plaintext-equivalence test (PET) on threshold ElGamal ciphertexts [13]. The test reveals whether two candidates are equal. Since colluding servers could combine secret-key shares and decrypt user inputs, at least one PET server must be honest. The outcome is posted to the bulletin board, so deviations are publicly detectable.

6 Secure Mapping of Private Permutation

Definition 1. *A protocol is a* Secure Mapping of Private Permutation *(SMPP) if, given input pairs* $((\lambda_1, E(M_1)), \ldots, (\lambda_n, E(M_n)))$ *from* n *users with pairwise distinct slots* $\lambda_1, \ldots, \lambda_n \in [m]$*, it outputs a table* T *such that* $T[\lambda_i] = E(M_i)$ *for every* $i \in [n]$*. In the composed VSM protocol, pairwise distinctness is guaranteed by UNS integrity.*

For slots $j \notin \{\lambda_1, \ldots, \lambda_n\}$ not assigned to any user, the table entry $T[j]$ contains $E(0)$ (the encryption of the neutral element). SMPP should preserve anonymity: when user i writes $E(M_i)$ to slot λ_i, no polynomial-time observer should learn which user produced the ciphertext stored at $T[\lambda_i]$. To achieve this, each user publishes two encrypted vectors:

1. a *location vector* $L_i^{\lambda_i}$ with a single 1 at coordinate λ_i and 0 elsewhere, and
2. a *message vector* $V_i^{\lambda_i, m_i}$ with entry M_i at coordinate λ_i and the neutral value elsewhere, namely 0 in the additive ElGamal encoding.

We denote their encryptions by $\hat{L}_i$ and $\hat{V}_i$. The user also proves that $\hat{L}_i$ encrypts a unit vector and that $\hat{V}_i$ is consistent with the same hidden slot. The servers then aggregate columns homomorphically to obtain the shuffled encrypted table T.

Linking UNS to SMPP. After user i receives its encrypted mix location $\hat{\lambda}_i$ from UNS, it must prove that the published vector $\hat{L}_i$ is consistent with the same hidden slot. Let B_λ be the unit vector with a single 1 at coordinate λ. Since

$$\lambda = \sum_{j=1}^{m} j\, b_j$$

for the bits of B_λ, consistency can be expressed as a linear relation between the encrypted slot $\hat{\lambda}_i$ and the encrypted unit vector $\hat{L}_i$. User i therefore provides a NIZK proof π_i^T for the relation

$$\mathbb{R}_\lambda = \{((g, h, \hat{L}, \hat{\lambda}), (R, B_\lambda, r')) : ((g, h, \hat{L}), R, B_\lambda) \in \mathbb{R}_{unit},$$
$$\hat{\lambda} = (g^{r'}, h^{r'} \prod_{j=1}^{m} (g^j)^{b_j})\}.$$

This relation is efficiently handled by Sigma protocols for linear relations [31]. By soundness, a malicious user cannot prove consistency between $\hat{\lambda}_i$ and a unit vector corresponding to a different slot. The formal integrity argument using π_i^T appears in Appendix A under *Integrity composition.*

We define *SMPP-Dec* as SMPP followed by joint decryption of the aggregated table T. Each participating user publishes

$$I_i(m_i, \lambda_i) := (\hat{L}_i, \hat{V}_i, \Pi_i, E(\lambda_i), \pi_i^T),$$

where Π_i denotes the well-formedness proof appropriate to the chosen SMPP instantiation.

Protocol 2 Additively-Homomorphic VSM

Setup.

1. The servers cooperatively generate a threshold ElGamal public key pk and secret-key shares.

Each user i:

1. Select a mix location $\lambda_i \in [m]$. (In the composed VSM protocol, λ_i is assigned by UNS rather than chosen randomly here.)
2. Construct the unit vector L_i and the message vector V_i for slot λ_i.
3. Encrypt them to obtain $\hat{L}_i$ and $\hat{V}_i$.
4. Publish $(\hat{L}_i, \hat{V}_i, \Pi_i^{EG})$, where Π_i^{EG} proves that $\hat{L}_i$ is a unit vector and $\hat{V}_i$ is consistent with the same hidden slot.

Servers.

1. Verify the proofs and discard malformed tuples.
2. For each column j, compute $T(j) := \prod_i \hat{V}'_{ij}$.
3. Publish $(T(1), \ldots, T(m))$.

6.1 ElGamal-based VSM

In the ElGamal instantiation, users encrypt the entries of $\hat{L}_i$ and $\hat{V}_i$ under a threshold ElGamal public key. User-submitted ciphertexts are authenticated (e.g., via signed ElGamal) so that a malicious user cannot malleate another user's submission; homomorphic aggregation is performed on the underlying ElGamal components only. Non-malleability is required only for user-submitted ciphertexts: the aggregated column products are public server-computed outputs and do not require Schnorr-style authentication tags.

Each user proves in zero knowledge that $\hat{L}_i$ encrypts a unit vector and that $\hat{V}_i$ is consistent with the same hidden slot: every non-special entry is an encryption of the neutral element, while the special entry contains the encrypted message. The proof reveals only that there exists a unique special position and that the message vector is consistent with it; by zero knowledge it does not reveal the position itself.

Given verified inputs, the servers aggregate column-wise:

$$T(j) := \prod_i \hat{V}'_{ij} = E\left(\sum_i V'_{ij}\right).$$

Because all non-special entries encrypt the neutral element, the unique user whose hidden slot equals j contributes its message, so $T(j) = E(M_i)$ for the user with $\lambda_i = j$.

6.2 BGN-based VSM

Trust assumption. The BGN setup requires an honest-majority MPC among the servers to generate the public parameters: specifically, strictly more than half of the k BGN servers must be honest, i.e., at least $\lfloor k/2 \rfloor + 1$ servers. This honest-majority threshold is the additional trust assumption relative to the threshold-ElGamal variant, which requires only one honest server.

Protocol 3 Somewhat Homomorphic VSM

Setup.
1. The servers run distributed key generation for the BGN public parameters and secret-key shares.
2. The servers select random challenge values $r_1, \ldots, r_m$ and fix a deterministic encryption [1] of 1.

Users.
1. Select $\lambda_i \in [m]$, construct L_i and V_i, and publish $\hat{L}_i, \hat{V}_i$.

Servers.
1. For each user i, verify bitness and unit-sum of $\hat{L}_i$.
2. If the checks pass, compute for each column j the value $E'(M_j) = \prod_i e(\hat{L}_{ij}, \hat{V}_{ij})$.
3. Publish the resulting encrypted table.

In the BGN instantiation, users publish only encryptions of their location and message vectors; the servers check well formedness homomorphically rather than via per-user NIZKs. The key observation is that BGN supports arbitrarily many additions and one multiplication via pairings [8]. To protect user-submitted ciphertexts against malleability, we use chosen-ciphertext-attack (CCA)-secure wrappers such as Naor-Yung [33].

For each user i, the servers verify that $\hat{L}_i$ encrypts a unit vector by testing: (i) bitness of each entry via the paired product $\prod_j e(\hat{L}_{ij}, \hat{L}_{ij} \cdot [1]^{-1})^{r_j} = E'(0)$, where [1] denotes the fixed encryption of 1 and $r_1, \ldots, r_m$ are the fresh random server challenges (so the check is randomized and non-malleable), and (ii) that the encrypted sum of all entries equals 1, checked as $\prod_j \hat{L}_{ij} \stackrel{?}{=} E(1)$. If these checks pass, the message at slot j is obtained from the encrypted scalar product between the j-th columns of $\hat{L}$ and $\hat{V}$:

$$E'(M_j) = \prod_{i \in [n]} e(\hat{L}_{ij}, \hat{V}_{ij}).$$

Since only the user whose hidden slot equals j contributes a non-neutral message entry, this yields the desired encrypted table.

6.3 SMPP via Fully Homomorphic Encryption

Another possible instantiation of SMPP uses fully homomorphic encryption (FHE), such as the Brakerski–Gentry–Vaikuntanathan (BGV) scheme. The purpose of this variant is to avoid sending explicit m-dimensional encrypted vectors. Instead of publishing the full vectors $\hat{L}_i$ and $\hat{V}_i$, user i publishes encryptions of a binary description of its mix location λ_i, together with encrypted payload bits that are routed homomorphically to the selected leaf. This reduces the client upload from $O(m)$ to $\tilde{O}(\log m)$, while requiring FHE evaluation and consistency proofs for the compressed input. Correctness of the compressed encoding can be proved using polylogarithmic-size lattice-based Σ-protocols [6].

The construction is based on *homomorphic traversal* [14]. Consider a complete binary tree with m leaves. A value initially placed at the root is routed to one leaf using encrypted controller bits. At each internal node, the homomorphic selection subroutines

$$\text{HS-L}(b, c) := (1 - c) \cdot b \quad \text{and} \quad \text{HS-R}(b, c) := c \cdot b$$

copy the current value b either to the left child or to the right child, according to the controller bit c. All controller bits and intermediate node values remain encrypted with indistinguishability under chosen-plaintext attack (IND-CPA) throughout the traversal. Since a complete binary tree with m leaves has $m - 1$ internal nodes, the traversal uses $m - 1$ homomorphic selection steps and has depth $\log m$.

To compress the location vector, user i decomposes its mix location $\lambda_i \in [m]$ into bits

$$b_{i,0}, \ldots, b_{i,d-1}, \qquad d = \lceil \log m \rceil, \qquad \lambda_i = \sum_{j=0}^{d-1} 2^j b_{i,j}.$$

The encrypted bits $\hat{b}_{i,j} = E(b_{i,j})$ serve as the demultiplexer controllers for the traversal circuit. To compress the message vector, the user also decomposes a fixed-size ℓ-bit payload representation into bits $v_{i,0}, \ldots, v_{i,\ell-1}$ and publishes encryptions $\hat{v}_{i,j} = E(v_{i,j})$. The same encrypted controller bits route these encrypted payload bits to the leaf indexed by λ_i, thereby producing under encryption the same sparse placement that the explicit vectors $\hat{L}_i$ and $\hat{V}_i$ would have represented.

The user publishes the compressed location encoding

$$(\hat{\lambda}_i, \hat{b}_{i,0}, \ldots, \hat{b}_{i,d-1}, \pi_i^{\lambda})$$

and the compressed payload encoding

$$(\hat{V}_i, \hat{v}_{i,0}, \ldots, \hat{v}_{i,\ell-1}, \pi_i^{V}).$$

The accompanying lattice-based zero-knowledge proofs certify that the submitted ciphertexts encrypt bits and that the encrypted location is consistent with its binary decomposition. Equivalently, the proof enforces the relation

$$\hat{\lambda}_i = h^r g^{\sum_{j=0}^{d-1} 2^j b_{i,j}},$$

in the corresponding encrypted-domain notation, together with the bitness of the $b_{i,j}$'s and the well-formedness of the encrypted payload bits. Such relations can be proved using compressed lattice-based proof systems [6]. The FHE traversal requires encrypting the relevant bits under a ring learning with errors (RLWE)-based encryption scheme and using the RLWE-to-RGSW (Ring Gentry-Sahai-Waters) conversion procedure of [14] for the encrypted controller values used by the selection circuit.

Routing one encrypted value through a complete binary tree with m leaves uses $m - 1$ selection nodes. If both children are materialized using HS-L and

Table 2. Online SMPP costs for n users and m slots (one batch).

Instantiation	Client upload	Public aggregation	Output decryption
ElGamal+NIZK	$O(m)$	$O(nm)$ group ops	m threshold dec.
BGN checks	$O(m)$	$O(nm)$ pairing/group ops	m threshold dec.
FHE traversal	$\tilde{O}(\log m + \ell)$	$O(\ell m)$ mults; depth $\log m$	decrypt m slots

HS-R, this requires $2(m-1)$ ciphertext multiplications and has multiplicative depth $\log m$. Routing one location marker and ℓ payload bits therefore requires $2(\ell+1)(m-1)$ ciphertext multiplications.

To ensure malicious security, the user additionally proves that the encrypted controller values are bits and that they form a valid binary decomposition of the encrypted location. These are circuit level estimates and do not include the concrete costs of RLWE to RGSW conversion or of generating and verifying these consistency proofs. Thus, the FHE construction reduces client upload asymptotically, while its concrete computation cost depends on the selected FHE and proof systems. An online cost summary for SMPP with n users and m slots in one batch is shown in Table 2.

The FHE row treats ℓ as the payload bit length; for fixed-size payloads, the client-upload term is $\tilde{O}(\log m)$.

7 VSM

We are finally ready to define VSM. VSM is the composition of UNS and SMPP. UNS is executed once per *epoch*: each user selects at random $K = \lceil \log n \rceil$ candidate mix locations (as justified by Lemma 2), and the servers run UNS so that each surviving user obtains an encrypted mixed location together with its plaintext interpretation from its own submitted list. The users then run SMPP for the current batch using their plaintext slots λ_i. The SMPP outputs a table T that realizes the permutation determined by the mixed locations $\lambda_{i i\in[n]}$, i.e., $T(\lambda_i) = E(m_i)$.

8 Security Analysis

Adversarial model. The analysis applies to one epoch and considers a static probabilistic polynomial-time (PPT) adversary that fixes its corruptions before the epoch begins. Corrupted parties share their state and may deviate arbitrarily, including choosing malformed inputs, omitting submissions, or causing aborts. The adversary may corrupt all but two users. In the threshold-ElGamal and PET-based variants, anonymity tolerates all but one corrupt server, while successful completion and decryption require the configured threshold of servers to participate. The BGN setup additionally assumes an honest majority (Section 6.2). In the TEE-based UNS variant, the host may be malicious, but the attested enclave code, its secret key, and the hardware root of trust are assumed secure.

Protocol 4 VSM

Phase I: Preprocessing by UNS

1. At the epoch cutoff, fix the participant set $\mathcal{U}_e$ and set $n := |\mathcal{U}_e|$; users that do not submit a candidate list are excluded.
2. Set a list-length parameter $K = \lceil \log n \rceil$.
3. Each user $i \in [n]\mathcal{U}_e$ randomly selects K candidate locations and computes $\hat{P}_i := (E_{pk}(l_{i1}), E_{pk}(l_{i2}), \ldots, E_{pk}(l_{iK}))$, which it sends to the servers.
4. The servers compute $UNS(\hat{P}_1, \ldots, \hat{P}_n)$ and publish the resulting encrypted mix locations. Each surviving user i then recovers λ_i in plaintext by identifying which of its posted ciphertexts appears in the public UNS output.

Phase II: Message submission by SMPP in the current epoch

1. Each participating user computes the SMPP input $I_i(m_i, \lambda_i)$ and publishes it on the bulletin board.
2. The servers execute the verification and aggregation steps of SMPP over the received inputs.
3. Let $\mathcal{S}_e \subseteq \mathcal{U}_e$ be the users whose valid SMPP inputs were received by the message deadline. The servers verify and aggregate $(I_i(m_i, \lambda_i))_{i \in \mathcal{S}_e}$ into a public table T. Users outside $\mathcal{S}_e$ are excluded from the SMPP execution, and their assigned locations remain unused.
4. The servers jointly decrypt the messages in T.

We assume authenticated channels and a public, authenticated, append-only bulletin board that remains available during the epoch. Board-level censorship, adaptive corruptions, and dynamic membership within an epoch are outside the model. Reusing UNS slots across epochs yields linkable pseudonymous slots; fresh unlinkability requires rerunning UNS.

Security properties. *Anonymity* means that the adversary cannot link honest users to their hidden locations or messages beyond the final output multiset. *Integrity* means that every honest user's validly submitted message appears at its assigned location, or the protocol aborts. *Self-verifiability* means that after SMPP-Dec, each honest user verifies correctness by checking the plaintext at their own secret location.

Theorem 1. *Protocol 1 satisfies UNS anonymity and integrity.*

Proof (Proof sketch). UNS anonymity follows from the semantic security of the encryption used for the candidate lists together with the fact that the collision transcript exposes only equality bits between currently active encrypted candidates. All users submit their encrypted candidate lists $\hat{P}_i$ simultaneously before UNS begins, so a computationally bounded adversary cannot adapt its candidates to target a specific honest user's ciphertexts. Since honest candidate lists are sampled independently from the same distribution, swapping the labels of two honest users leaves the joint distribution of the entire equality-bit transcript invariant; this is formalized in Appendix A. Integrity follows from the collision-testing mechanism and the fact that the protocol outputs only surviving current candidates; by construction no two such candidates are equal.

Theorem 2. *Protocols 2 and 3 satisfy SMPP anonymity and integrity.*

Proof (Proof sketch). For the ElGamal instantiation, anonymity follows by a standard hybrid: first replace the real NIZK transcript by a simulated one using zero knowledge, then use ciphertext indistinguishability to swap the hidden slots/messages of two honest users. Integrity follows from the soundness of the well-formedness proofs and of the linking proof π_i^T, which prevents a malicious user from proving consistency with a slot different from the one returned by UNS. For the BGN instantiation, the public checks reveal only whether the posted vectors satisfy permutation-invariant predicates (bitness, unit sum, and consistency), while the final encrypted table is obtained from semantically secure Naor-Yung protected ciphertexts; thus the hidden location remains computationally concealed.

Theorem 3. *Assume UNS satisfies anonymity and SMPP satisfies anonymity. Then the composed VSM protocol satisfies anonymity for a single epoch. Furthermore, VSM satisfies integrity, and SMPP-Dec provides unconditional individual verifiability for every honest user.*

Proof (Proof sketch). Anonymity. The output of UNS is an encrypted list of pairwise distinct slots, used as part of the SMPP input. By UNS anonymity, swapping the two challenge honest users changes only the hidden assignment of their encrypted slots. By SMPP anonymity, the adversary cannot then distinguish which honest user wrote to which hidden slot. Hence the sequential composition UNS→SMPP preserves single-epoch anonymity. *Integrity.* By the soundness of π_i^T, no PPT adversary can produce a valid translation proof for a location inconsistent with the UNS output, except with negligible probability. SMPP integrity then gives $T(\lambda_j) = E(m_j)$ for all honest j. *Self-verifiability.* After SMPP-Dec, each honest user checks the plaintext entry at its own secret slot; this check is purely local and unconditional.

Rerunning UNS with fresh randomness is intended to provide fresh slots across epochs; a full multi-epoch, multi-message treatment with adaptive corruptions and corrupted-sender scheduling is left for future work.

9 Conclusions and Future Work

We introduced Verifiable SelfMix, a modular anonymous communication architecture that separates private location assignment from anonymous message placement. VSM allows users to verify that their messages appear at their assigned locations without performing cryptographic verification, and supports different instantiations of UNS and SMPP with different trust and efficiency tradeoffs.

Our analysis is theoretical and considers a fixed participant set and static corruptions within each epoch. The primary SMPP constructions require $O(m)$

ciphertext upload per user and $O(nm)$ public aggregation, while the FHE construction reduces client upload at the cost of additional homomorphic computation. Implementing the proposed constructions, evaluating their concrete performance, and extending the model to dynamic participation and adaptive corruptions are natural directions for future work.

References

1. Abe, M.: Universally verifiable mix-net with verification work independent of the number of mix-servers. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 437–447. Springer (1998)
2. Abraham, I., Pinkas, B., Yanai, A.: Blinder: MPC based scalable and robust anonymous committed broadcast. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS 2020). ACM (2020). https://doi.org/10.1145/3372297.3417261
3. Adida, B., Wikström, D.: How to shuffle in public. In: Theory of Cryptography Conference. pp. 555–574. Springer (2007)
4. Adida, B., Wikström, D.: Offline/online mixing. In: International Colloquium on Automata, Languages, and Programming. pp. 484–495. Springer (2007)
5. Alexopoulos, N., Kiayias, A., Talviste, R., Zacharias, T.: Mcmix: Anonymous messaging via secure multiparty computation. In: USENIX security symposium. pp. 1217–1234 (2017)
6. Attema, T., Cramer, R., Kohl, L.: A compressed σ-protocol theory for lattices. In: Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part II. pp. 549–579. Springer (2021)
7. Barman, L., Dacosta, I., Zamani, M., Zhai, E., Pyrgelis, A., Ford, B., Feigenbaum, J., Hubaux, J.: Prifi: Low-latency anonymity for organizational networks. Proceedings on Privacy Enhancing Technologies **2020**(4), 24–47 (2020). https://doi.org/10.2478/popets-2020-0061
8. Boneh, D., Goh, E.J., Nissim, K.: Evaluating 2-dnf formulas on ciphertexts. In: Theory of cryptography conference. pp. 325–341. Springer (2005)
9. Boyle, E., Goldwasser, S., Tessaro, S.: Communication locality in secure multiparty computation. In: Theory of Cryptography Conference. pp. 356–376. Springer (2013)
10. Chaum, D.: The dining cryptographers problem. J. cryptology **1**, 65–75 (1988)
11. Chaum, D., Das, D., Javani, F., Kate, A., Krasnova, A., De Ruiter, J., Sherman, A.T.: cmix: Mixing with minimal real-time asymmetric cryptographic operations. In: Applied Cryptography and Network Security, ACNS. Springer (2017)
12. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM **24**(2), 84–90 (1981)
13. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: Toward a secure voting system. In: 2008 IEEE Symposium on Security and Privacy (sp 2008). pp. 354–368. IEEE (2008)
14. Cong, K., Das, D., Park, J., Pereira, H.V.: Sortinghat: Efficient private decision tree evaluation via homomorphic encryption and transciphering. Cryptology ePrint Archive (2022)
15. Corrigan-Gibbs, H., Boneh, D., Mazières, D.: Riposte: An anonymous messaging system handling millions of users. In: 2015 IEEE Symposium on Security and Privacy. pp. 321–338. IEEE (2015)

16. Corrigan-Gibbs, H., Ford, B.: Dissent: accountable anonymous group messaging. In: Proceedings of the 17th ACM conference on Computer and communications security. pp. 340–350 (2010)
17. Corrigan-Gibbs, H., Wolinsky, D.I., Ford, B.: Proactively accountable anonymous messaging in verdict. In: 22nd USENIX Security Symposium (USENIX Security 2013). pp. 147–162. USENIX Association (2013), https://www.usenix.org/conference/usenixsecurity13/technical-sessions/presentation/corrigan-gibbs
18. Danezis, G., Diaz, C.: A survey of anonymous communication channels. Tech. Rep. MSR-TR-2008-35, Microsoft Research (Feb 2008), https://www.microsoft.com/en-us/research/publication/a-survey-of-anonymous-communication-channels/
19. Das, D., Mangipudi, E.V., Kate, A.: Organ: Organizational anonymity with low latency. Proceedings on Privacy Enhancing Technologies **2022**(3), 582–605 (2022). https://doi.org/10.56553/popets-2022-0087
20. Dunning, L.A., Kresman, R.: Privacy preserving data sharing with anonymous id assignment. IEEE transactions on information forensics and security **8**(2), 402–413 (2012)
21. Eskandarian, S., Boneh, D.: Clarion: Anonymous communication from multiparty shuffling protocols. In: Network and Distributed System Security Symposium (NDSS 2022). Internet Society (2022). https://doi.org/10.14722/ndss.2022.24141
22. Eskandarian, S., Corrigan-Gibbs, H., Zaharia, M., Boneh, D.: Express: Lowering the cost of metadata-hiding communication with cryptographic privacy. In: 30th USENIX Security Symposium (USENIX Security 2021). pp. 1775–1792. USENIX Association (2021), https://www.usenix.org/conference/usenixsecurity21/presentation/eskandarian
23. Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. In: International Workshop on the Theory and Application of Cryptographic Techniques. pp. 244–251. Springer (1992)
24. Gjøsteen, K., Lund, A.S.: An experiment on the security of the norwegian electronic voting protocol. Annals of Telecommunications **71**(7), 299–307 (2016)
25. Golle, P., Juels, A.: Dining cryptographers revisited. In: Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 456–473. Springer (2004)
26. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. Journal of Cryptology **23**(4), 546–579 (2010)
27. Hamada, K., Kikuchi, R., Ikarashi, D., Chida, K., Takahashi, K.: Practically efficient multi-party sorting protocols from comparison sort algorithms. In: Information Security and Cryptology–ICISC 2012: 15th International Conferenc. pp. 202–216. Springer (2013)
28. Kwon, A., Corrigan-Gibbs, H., Devadas, S., Ford, B.: Atom: Horizontally scaling strong anonymity. In: Proceedings of the 26th ACM Symposium on Operating Systems Principles (SOSP 2017). pp. 406–422. ACM (2017). https://doi.org/10.1145/3132747.3132755
29. Langowski, S., Servan-Schreiber, S., Devadas, S.: Trellis: Robust and scalable metadata-private anonymous broadcast. In: Network and Distributed System Security Symposium (NDSS 2023). Internet Society (2023). https://doi.org/10.14722/ndss.2023.23088
30. Mardi, D., Tanwar, S., Howlader, J.: Multiparty protocol that usually shuffles. Security and Privacy (2021)
31. Maurer, U.: Zero-knowledge proofs of knowledge for group homomorphisms. Designs, Codes and Cryptography **77**(2), 663–676 (2015)

32. Movahedi, M., Saia, J., Zamani, M.: Secure multi-party shuffling. In: International Colloquium on Structural Information and Communication Complexity. pp. 459–473. Springer (2015)
33. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: Proceedings of the twenty-second annual ACM symposium on Theory of computing. pp. 427–437 (1990)
34. Neff, C.A.: A verifiable secret shuffle and its application to e-voting. In: Proceedings of the 8th ACM conference on Computer and Communications Security. pp. 116–125 (2001)
35. Niemi, V., Renvall, A.: How to prevent buying of votes in computer elections. In: International Conference on the Theory and Application of Cryptology. pp. 164–170. Springer (1994)
36. Park, C., Itoh, K., Kurosawa, K.: Efficient anonymous channel and all/nothing election scheme. In: Workshop on the Theory and Application of of Cryptographic Techniques. pp. 248–259. Springer (1993)
37. Piotrowska, A.M., Hayes, J., Elahi, T., Meiser, S., Danezis, G.: The loopix anonymity system. In: 26th USENIX Security Symposium (USENIX Security 2017). pp. 1199–1216. USENIX Association (2017), https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/piotrowska
38. Ruffing, T., Moreno-Sanchez, P., Kate, A.: P2p mixing and unlinkable bitcoin transactions. In: Network and Distributed System Security Symposium (NDSS 2017). Internet Society (2017)
39. Sako, K., Kilian, J.: Receipt-free mix-type voting scheme. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 393–403. Springer (1995)
40. Taassori, M., Shafiee, A., Balasubramonian, R.: Vault: Reducing paging overheads in sgx with efficient integrity verification structures. In: Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems. pp. 665–678 (2018)
41. Wikström, D.: A sender verifiable mix-net and a new proof of a shuffle. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 273–292. Springer (2005)

A Formal Games and Proof Notes

We record the anonymity games used in the proof sketches and the missing proof details referenced in Section 8.

Anon-UNS. The challenger samples all honest users' candidate lists, receives the corrupted users' encrypted lists from $\mathcal{A}$, chooses two honest challenge users p_0, p_1 and two target slots λ_0, λ_1, tosses $b \xleftarrow{R} \{0,1\}$, assigns p_b to λ_0 and p_{1-b} to λ_1, runs UNS, and gives $\mathcal{A}$ the public transcript and outputs. The adversary wins if it guesses b.

Proof note for Theorem 1. The only public leakage during UNS beyond the final encrypted slots is the equality-bit transcript produced by collision tests. Conditioned on the corrupted users' inputs, the honest candidate lists are sampled independently from the same distribution. Therefore, for any deterministic scheduler whose next comparison depends only on the prior transcript, swapping the labels of two honest users leaves the joint distribution of the entire equality-bit transcript invariant. Hence the challenge worlds differ only in the encryptions of the two honest users' lists and final slots, which are computationally hidden by IND-CCA security of the underlying encryption.

Anon-SMPP. The challenger receives the corrupted users' valid SMPP inputs. The adversary $\mathcal{A}$ chooses two honest challenge users p_0, p_1 and two messages m_0, m_1. The challenger then independently selects two distinct challenge slots $\lambda_0 \neq \lambda_1$, tosses $b \xleftarrow{R} \{0,1\}$, sets $\tilde{I}_0 := I_0(m_b, \lambda_0)$ and $\tilde{I}_1 := I_1(m_{1-b}, \lambda_1)$, completes the bulletin-board instance with these challenge inputs, and returns the public SMPP transcript and output table to $\mathcal{A}$. The adversary wins if it guesses b.

Proof note for Theorem 2. For the ElGamal instantiation, the proof proceeds via the standard hybrid in which the real NIZK transcript is first replaced by a simulated one using zero knowledge; ciphertext indistinguishability then hides the swap of the two honest users' hidden slots/messages. For the BGN instantiation, the public checks reveal only whether the posted vectors satisfy permutation-invariant predicates (bitness, unit sum, and consistency). Moreover, the pairing checks are randomized by fresh server exponents chosen after the ciphertexts are posted, so the public outcome certifies validity without exposing the position of the unique nontrivial entry. Together with semantic security of the Naor–Yung protected BGN ciphertexts, this yields SMPP anonymity.

Integrity composition. Let R_λ be the relation proved by the translation proof π_i^T linking UNS output $\hat{\lambda}_i$ to the SMPP vectors $(\hat{L}_i, \hat{V}_i)$. By soundness of the Sigma protocol for R_λ, no PPT adversary can produce a valid π_i^T for a location vector inconsistent with $\hat{\lambda}_i$, except with negligible probability. Combined with UNS integrity and SMPP integrity, this implies VSM integrity. Self-verifiability follows because after SMPP-Dec each honest user checks the plaintext table entry at its own secret slot.