Oblivious Signaling

Mirza Kamrul Bashar Shuhan
George Mason University

Foteini Baldimtsi
George Mason University & Mysten Labs

Giuseppe Ateniese
George Mason University

Abstract

An anonymous messaging service has to solve a basic routing problem: a server must deliver an encrypted message to its recipient without learning who the recipient is. Broadcasting all ciphertexts hides the destination but forces every recipient to constantly scan for new messages. Oblivious Message Retrieval (OMR; CRYPTO '22) tackles this by using fully homomorphic encryption (FHE) to let an untrusted server perform message retrieval on a recipient's behalf without learning which messages are pertinent.

We introduce *Oblivious Signaling*, which shifts this cost from *retrieval* to *sending*. The server maintains a fixed-size encrypted inbox for each recipient. When a sender submits a message, the server applies the same homomorphic update to every inbox: the intended inbox absorbs the message, and the rest remain unchanged at the plaintext level. The update is uniform, can be parallelized across inboxes, and ties the delivery cost strictly to the size of the anonymity set rather than global traffic. Recipients retrieve by fetching and decrypting their inbox, so checking for new messages is independent of the global traffic.

We formalize receiver privacy against an untrusted server, even when it colludes with other users, give a concrete construction based on fully homomorphic encryption, and analyze the resulting "digital postage" trade-off: delivery is expensive, but checking is cheap. Our prototype identifies practical regimes in which this cost-model shift is preferable to scan-based retrieval, even with highly optimized OMR implementations. This cost model is well-suited to settings where recipients check frequently, and messages arrive sporadically, and it naturally discourages high-volume spam.

1 Introduction

Most interactions with an asynchronous messaging service are not actual message deliveries but recipients *checking whether any message has arrived*.[1] This imbalance is inherent to asynchronous communication: recipients check frequently, whereas any individual user receives new messages only occasionally. To support these frequent checks, a conventional messaging service maintains a per-recipient inbox and routes each incoming message into the appropriate inbox. But routing is precisely what exposes communication patterns. Even if message contents are encrypted end-to-end, the server still observes which inbox each message is placed into, and can learn who is communicating with whom. This leads to the question we study: *can a single untrusted server support asynchronous delivery without learning message destinations?*

A natural first attempt to hide message destinations is to avoid routing altogether. Instead of placing messages into per-recipient inboxes, senders can simply post ciphertexts to a public bulletin board, leaving it to recipients to identify their own messages. This removes destination leakage at the server, but it replaces routing with scanning. As the board grows, recipients must scan an ever-growing stream of ciphertexts, and in the simplest instantiations, trial-decrypt a large fraction of them just to determine whether anything new has arrived.

In a recent line of work, Oblivious Message Retrieval (OMR) [36] was introduced as a primitive that lets recipients outsource bulletin-board scans to an untrusted server using fully homomorphic encryption (FHE). The server processes a range of board items under a recipient-provided evaluation key and returns a compact encrypted digest, while remaining oblivious to which messages were pertinent. A series of follow-up works improve the efficiency of OMR through different building blocks, parameter choices, and system-level optimizations [4, 31, 33, 38], while GOMR extends the OMR paradigm to a group setting [37].

The OMR framework fundamentally treats the bulletin board as a global log where message retrieval is the central operation. The expensive homomorphic work is paid each time a recipient checks for new messages: to learn whether anything has arrived, the system must homomorphically process the newly appended portion of the global log under that

[1] As an example, Charzinski's analysis of POP3/IMAP traffic traces reports that only 2.5%–25% of retrieval connections carried at least one e-mail (depending on trace and protocol), and that the remaining connections were used only to check for new mail [8].

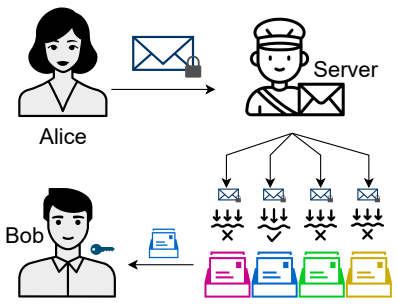

Figure 1: Alice messages Bob via Oblivious Signaling

recipient's key. This design is reasonable when retrievals are periodic and predictable, but it is misaligned with the common regime where recipients poll frequently and usually have nothing new, causing the system to repeatedly incur the full cost of oblivious message retrieval even when no new messages are present.

Our Approach. Observing this misalignment, we adopt a different design philosophy that places the privacy-preserving computation at the moment a message is *sent*, rather than at each moment a recipient *checks* for new messages. We introduce Oblivious Signaling, a push-based paradigm in which the server maintains an encrypted inbox for each recipient (Figure 1). When a sender submits a message, the server applies the same homomorphic update to every inbox. Only the inbox *pertinent* to the intended recipient changes at the plaintext level (it absorbs the message); all other inboxes remain unchanged except with negligible probability due to false positives. Rare false-positive collisions may corrupt the inserted plaintext for the targeted inbox (Section 4). Since the server performs an identical update for every inbox, it cannot determine which recipient was targeted. The server state consists only of encrypted inboxes and public parameters, with no correctness-critical hidden state across executions, simplifying replication and recovery.

Cost model and trade-off. The key trade-off is that Oblivious Signaling moves the expensive privacy-preserving work from retrieval to delivery. Recipients check for messages by fetching and decrypting a fixed-size inbox, so their cost is *independent* of the unread global traffic. Delivery is more expensive than in OMR because each delivered message induces server work proportional to the recipient population. Thus, Oblivious Signaling increases sender-triggered server work in exchange for cheap, frequent recipient checks. This is particularly attractive in polling-heavy settings such as email, where checks are frequent and actual deliveries are sparse.

Our Contributions. We summarize our main contributions towards building Oblivious Signaling as follows:

- We formalize Oblivious Signaling (Π_{OS}) as a cryptographic primitive. Our definition captures the structure of per-recipient encrypted inboxes, the semantics of a uniform inbox update, the role of pertinency tags, and the requirement that only the inbox pertinent to a delivered message undergoes a semantic change.
- We define receiver privacy against a malicious server, allowing collusion with corrupted senders and receivers, and we characterize correctness failures arising from false positives and collisions.
- We give a concrete construction (OSig) from fully homomorphic encryption that realizes a uniform inbox update in which only the intended inbox changes semantically. Our construction supports *batched* message delivery.
- We implement a prototype using TFHE-rs [48] and evaluate it in Section 6 to demonstrate OSig's fundamental cost-model improvements over OMR. By decoupling computational overhead from global system traffic, OSig establishes a proportional architecture where delivery costs scale strictly with the anonymity set, effectively eliminating the parasitic scanning overhead of pull-based retrieval.

1.1 Related Work

No single system fits all communication settings, and different designs for metadata protection make different trade-offs. We target a common asynchronous sender-server-receiver model under a "digital postage" cost structure, where senders bear the cost of delivery. This reflects practical settings in which recipients may receive messages sporadically or in bursts (e.g., email) and naturally discourages abuse, such as spam. Our goal is to support this regime using a single honest-but-curious server and strong, purely cryptographic privacy guarantees that fully protect the message-recipient linkage. Crucially, our design achieves this desired level of privacy while strictly avoiding stronger environmental and trust assumptions: it requires no Trusted Execution Environments (TEEs), relies on no multiple non-colluding servers, mandates no live secret state maintained by the server across executions, and requires no pre-shared secrets or out-of-band coordination between the sender and receiver. This model aligns most closely with single-server OMR systems, but fundamentally shifts the cost of privacy from receiver-initiated scans to sender-driven delivery.

Oblivious Message Retrieval (OMR). The OMR line of work starting with [36] and extended/optimized in [4, 31, 33, 37, 38] develops a single-server retrieval primitive in which recipients outsource message detection to an untrusted server. OMR achieves strong receiver privacy and has seen several efficiency improvements, including group messaging, more optimized homomorphic circuits, and hardware acceleration. Group OMR [37] extends the primitive to settings where messages may have multiple recipients. PerfOMR [38] improves efficiency by leveraging RLWE-based constructions and more efficient homomorphic decryption circuits, while

SophOMR [31] further optimizes performance by exploiting SIMD-style packing and linear-algebraic structure in the underlying homomorphic encryption scheme. Recent work [4] accelerates this line further using FPGA hardware. Another direction [34] studies robustness against denial-of-service and spamming attacks by malicious senders who craft clues that induce excessive false positives, and builds on PerfOMR to provide formal guarantees against such attacks with moderate overhead. Concurrent with our work, InstantOMR [33] revisits the server bottleneck through a hybrid design that combines TFHE programmable bootstrapping with RLWE operations, enabling finer-grained parallelism and substantially reducing latency relative to earlier batch-oriented BFV-based schemes. Departing from the single-server model we consider, Homerun [24] proposes an OMR construction based on two communicating but non-colluding servers. This approach reduces cryptographic overhead but relies on stronger trust assumptions and a semi-honest model to ensure privacy.

Because the original single-server OMR most closely matches our privacy goals and system model, it serves as our primary baseline. Like Oblivious Signaling, this setting uses no trusted hardware, no non-colluding servers, and no live server-held secrets. Streaming OMR [35, Sections 7.5 and 11] moves part of retrieval to message-arrival time, but it requires the server to keep secret randomness that fixes the digest structure until finalization. Since correctness then depends on preserving this hidden state across executions, we exclude streaming OMR as a baseline under our model. In Section 2.1 we provide a more detailed technical comparison, and in Section 6 an empirical comparison with InstantOMR [33] (the most latency-optimized design to date).

Fuzzy Message Detection (FMD). Fuzzy Message Detection (FMD) [3] enables a server to return a probabilistic superset of potentially relevant messages using fuzzy tags that induce false positives. While highly efficient, FMD provides weaker privacy: recipients' anonymity depends on the presence of false matches and degrades under intersection and traffic-analysis attacks [45].

Private Signaling (PS). Private Signaling (PS) [23, 39] achieves strong recipient privacy with recipient-side work independent of system size. Proposed constructions include (1) a two-server protocol based on garbled circuits and (2) a single-server design relying on a trusted execution environment (TEE). The two-server model requires non-colluding servers, a strong assumption in practice. TEE-based systems offer good scalability but rely on hardware trust and are vulnerable to side-channel leakage [7, 20]. Our work shares the push-based philosophy of PS but operates in a single-server, purely cryptographic setting; accordingly, we do not include PS in our direct performance comparisons due to its stronger trust assumptions.

Additional Relevant Systems. We briefly discuss other metadata-private communication systems that rely on different assumptions or protect different parties.

DC-nets [9, 16, 17] provide strong metadata privacy by having participants jointly generate ciphertexts such that only one message is revealed per round without exposing the sender. However, they require costly all-to-all synchronous communication. Mixnets anonymize senders by routing messages through chains of servers that shuffle and re-encrypt traffic, breaking input-output linkage. While early systems such as *Riffle* [28] and *cMix* [10] faced scalability challenges, newer designs including *Atom* [27], *Loopix* [42], and *Trellis* [29] achieve horizontal scalability. However, mixnets primarily address sender anonymity and do not directly support private asynchronous message retrieval without additional trust or system assumptions.

Private information retrieval (PIR) [13] is another direction for anonymous messaging that hides which indices are fetched but presumes the recipient already knows which indices to ask for. PIR-based approaches enable asynchronous communication but typically assume multiple non-colluding servers. Systems such as *Riposte* [15] allow anonymous writes to a shared database using distributed point functions (DPFs), while audit mechanisms prevent disruption. Later systems, including *Express* [18] and *Sabre* [47], improve efficiency through optimized DPF constructions and lightweight auditing. Despite their scalability, these DPF-based systems also rely on multiple non-colluding servers, a stronger trust assumption than the single-server cryptographic model we consider.

2 Technical Overview

Instead of organizing the system around a global message log that recipients must repeatedly scan, Oblivious Signaling maintains an encrypted inbox for each registered receiver at the server. We describe the overall framework of Oblivious Signaling (OSig) in Figure 2.

Each receiver r runs *KeyGen* to obtain a keypair (pk_r, sk_r) and registers pk_r with the server. The server maintains a *postbox* $\mathsf{POSTBOX}$, which stores for every registered receiver r a fixed-capacity encrypted inbox INBOX_r together with an encrypted inbox tag iTag_r. The tag has length $\ell = \Theta(\log(1/\varepsilon_p))$, chosen so that unintended tag matches occur with probability at most ε_p.

To send messages to distinct receivers $r_1, \ldots, r_g$, the sender forms a batch request $\mathsf{B}_{\mathsf{req}}$ whose ith entry is a ciphertext pair $(\mathsf{signal}_i, \mathsf{pTag}_i)$, where signal_i encrypts msg_{r_i} under pk_{r_i} and pTag_i encrypts the fixed tag 1^ℓ under the same key. Every entry in $\mathsf{B}_{\mathsf{req}}$ is intended for a distinct recipient, so no batch contains two messages for the same receiver.

When processing a batch, the server applies the same homomorphic update procedure to every inbox $\mathsf{INBOX}_j \in \mathsf{POSTBOX}$. Fix a receiver r and consider the update of INBOX_r. For each batch entry, the procedure computes (under encryption) whether the entry is intended for r by testing whether its encrypted tag matches the stored inbox tag iTag_r. Intuitively, this "match test" is keyed to the receiver:

the one entry prepared for r matches iTag_r, while any other entry matches only with probability at most ε_p. If a match occurs, the server homomorphically right-shifts existing slots of the fixed-size inbox and places the corresponding ciphertext signal on top of the fixed-size inbox; if no match occurs, the inbox contents (plaintext level) are unchanged (except with negligible probability). Because the same update procedure is evaluated for every inbox, the server cannot tell which receivers were targeted by the batch.

Retrieval is local and non-interactive: receiver r periodically downloads INBOX_r and decrypts using sk_r. The cost of checking for new messages is therefore independent of the global traffic volume or other users' activity. Unlike scan-based systems that repeatedly process an ever-growing message stream, OSig performs the receiver-privacy work once at delivery time and amortizes it across future reads. This yields a "digital postage" model: sending incurs the dominant cost, while inbox checking remains constant and lightweight.

Oblivious Signaling supports payloads in two operational modes. In **direct-inbox mode**, the object inserted into the recipient's encrypted inbox is the FHE-encrypted payload itself. This is the mode evaluated in our prototype. Larger direct payloads can be handled by packing the data across multiple ciphertext chunks or by increasing the plaintext modulus, at the corresponding cryptographic cost. In **signal mode**, the inbox carries only an FHE-encrypted handle to a payload stored outside the inbox, for example, on a public bulletin board as in OMR. The privacy-preserving delivery mechanism is unchanged: the server still updates all encrypted inboxes uniformly, but the signal now encodes a pointer rather than the full message. Thus, large-message support does not require the FHE plaintext space to contain the whole application message; it only needs to contain a fixed-size handle that lets the recipient retrieve the externally stored payload.

2.1 Comparison with OMR

A core difference between OMR and our approach is that OMR pays its cost when recipients scan the bulletin board rather than when messages are delivered. One additional complication arises during OMR retrieval: the recipient must specify an upper bound $\bar{k}$ on the number of pertinent messages in the current retrieval window. Let k denote the *actual* number of pertinent messages for a recipient. Retrieval succeeds only if $k \le \bar{k}$; otherwise the recipient obtains an overflow result, learns the exact value of k, and must re-issue the request with a larger bound. This behavior increases cost, *leaks* information about message volumes to the server, and creates pressure to overestimate $\bar{k}$, which in turn enlarges OMR's digest and client-side work since both scale with $\bar{k}$. Proposed mitigations rely on unlinkable detection keys or two non-colluding servers [35], which introduce environmental assumptions outside the single-server model. Streaming OMR [35, Sections 7.5 and 11] moves some retrieval work to

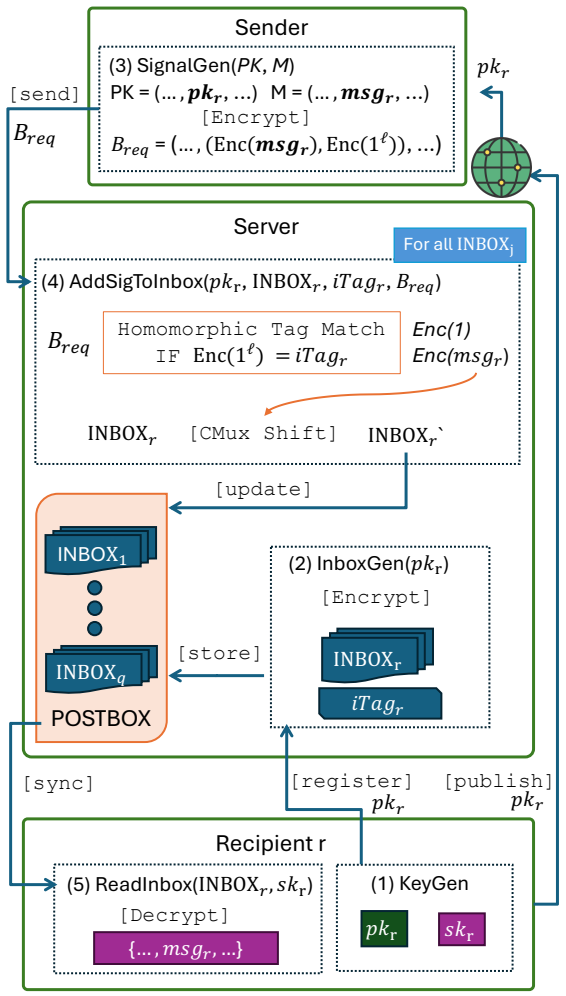

Figure 2: **Overview of Oblivious Signaling.** Each receiver has an encrypted inbox stored at the server. Senders submit a batch of ciphertext pairs (signal, tag) under recipients' public keys. The server applies the same homomorphic update procedure to every inbox, inserting the encrypted message only into the matching recipient inbox and leaving all others unchanged (except with negligible false positives). Receivers periodically download and decrypt their own inbox.

message-arrival time, but still requires later finalization and server-side state tied to the unread traffic window. Our Oblivious Signaling framework avoids this mechanism entirely by not requiring any estimate of recipient traffic, and recipients always retrieve at a fixed cost.

Asymptotic comparison. Across OMR constructions [31, 33, 36, 38], performance is governed by a few API-level parameters: the number of bulletin-board messages scanned N, the clue[2] length $\ell = \Theta(\log(1/\varepsilon_p))$ determined by the target false-positive rate, and the recipient-chosen bound $\bar{k}$ on perti-

[2] A short encrypted tag that allows the OMR server to test homomorphically whether a message is pertinent to a given recipient.

Metric	OMR [31, 33, 36, 38]	Oblivious Signaling
Access Model	Scan-based	Push-based
Cost Phase	Retrieval	Delivery
Scales With	Number of bulletin board messages scanned, N	Number of recipients in the system, q
Server Work	Scans N incoming messages to build digest $\widetilde{O}(N\ell)$	Delivers g messages (inbox size of ν) $O\big(q(g\ell+\nu)\big)$

Table 1: **Conceptual comparison of asymptotic server-side cost models.** OMR incurs scan-based retrieval cost $\widetilde{O}(N\ell)$, where N is the number of bulletin-board messages scanned in a retrieval window and ℓ is the tag length. Oblivious Signaling performs uniform push-based updates with per-batch cost $O\big(q(g\ell+\nu)\big)$, where q is the number of recipients, g the batch size, and ν the inbox capacity.

nent messages. The server scans N messages in the bulletin board (i.e., the messages in the recipient's retrieval window, for example the new messages posted since its previous scan) and evaluates a clue-check predicate of cost $O(\ell)$, giving a baseline detection cost $O(N\ell)$; additional compression and aggregation contribute only polylogarithmic overhead, so server time is summarized as $\widetilde{O}(N\ell)$. Digest size is provisioned for the expected number of detected items (recipient-bounded true positives plus false positives), $\hat{k} = \widetilde{O}(\bar{k} + \varepsilon_p N)$, yielding communication $\widetilde{O}(\bar{k} + \varepsilon_p N)$. When explicit indices are returned, each requires $\Theta(\log N)$ bits; otherwise, this is absorbed by compression. Receiver-side computation is at least linear in the recovered set and may involve operations polynomial in $\bar{k}$.

OMR exhibits a fixed structural trade-off: retrieval requires scanning all N messages in the retrieval window, while the returned digest scales only with the number of pertinent messages. This cost model is inherently unfair to low-volume recipients, since recipients with few or no messages still incur the full retrieval cost.

We now analyze asymptotics for Oblivious Signaling. Let q be the number of registered recipients (number of INBOXs), let ν be the receiver-chosen inbox capacity (number of retained pertinent messages), and let $\ell = \Theta(\log(1/\varepsilon_p))$ be the tag length determined by the target false-positive rate ε_p. A sender submission is a batch $\mathsf{B}_{\mathsf{req}}$ of size $g \le \bar{g}$ containing ciphertext pairs $(\mathsf{signal}_i, \mathsf{pTag}_i)$. Abstracting the underlying FHE costs and treating each homomorphic primitive as unit cost, batch creation is $O(g(\ell + |m|))$ work and communication, where $|m|$ is the message bitlength.

For each batch, the server performs a uniform update over all q recipients. For every inbox, it (i) checks each tag in the batch against the stored tag at cost $O(g\ell)$, (ii) accumulates the unique pertinent message using $O(g)$ conditional selects, and (iii) performs an oblivious inbox update via a ν-step shift costing $O(\nu)$. The per-batch server time is therefore $O\big(q(g\ell+\nu)\big)$. Receiver-side reading requires $O(\nu)$ decryptions of the fixed-size inbox. Thus Oblivious Signaling replaces OMR's N-message scan with a push-based uniform update whose cost depends on (q, g, ℓ, ν): linear in the recipient population and batch size, logarithmic in precision via ℓ, and linear in the receiver-chosen capacity ν. This cost separation allows Oblivious Signaling to *scale gracefully* in heterogeneous systems, ensuring that recipients incur work in proportion to their inbox storage, while sender submissions induce server work in proportion to the recipient population (the anonymity set), rather than the global traffic scanned at retrieval. A high-level comparison of how costs scale in OMR versus Oblivious Signaling is summarized in Table 1.

3 Preliminaries

Notation. Let λ be the security parameter, $poly(\cdot)$ a polynomial, and $\mathrm{negl}(\lambda)$ a negligible function. Let $\mathcal{K}$ denote the secret-key space, $\mathcal{M}$ the plaintext space, and $\mathcal{C}$ the ciphertext space. PPT stands for probabilistic polynomial-time. For $n \in \mathbb{N}$, let $[n] \triangleq \{1, \ldots, n\}$ and let $\mathbb{C}_n$ be the set of all polynomial-size circuits that take n inputs and produce one output. For $\ell \in \mathbb{N}$, 1^ℓ denotes the all-ones bitstring of length ℓ. We write $c = [\![m]\!]_{sk}$ for a ciphertext $c \in \mathcal{C}$ that decrypts to $m \in \mathcal{M}$ under secret key $sk \in \mathcal{K}$. Throughout the paper, log denotes the base-2 logarithm unless otherwise specified.

3.1 Fully Homomorphic Encryption

A Fully Homomorphic Encryption (FHE) scheme allows the evaluation of a circuit over encrypted data [21]. Formally:

Definition 3.1 (Fully Homomorphic Encryption)**.** An FHE scheme Π_{FHE} consists of the following PPT algorithms:

- $\Pi_{FHE}.\mathsf{ParamsGen}(1^\lambda) \to PP$: Given a security parameter $\lambda \in \mathbb{N}$, outputs public parameters PP that specify the plaintext space $\mathcal{M}$, ciphertext space $\mathcal{C}$, and the family of supported evaluation circuits (e.g., $\mathbb{C}_n$ for each $n \in \mathbb{N}$).
- $\Pi_{FHE}.\mathsf{KGen}(1^\lambda) \to (pk, sk)$: Given a security parameter $\lambda \in \mathbb{N}$, outputs a public key pk and a secret key $sk \in \mathcal{K}$.
- $\Pi_{FHE}.\mathsf{Enc}(pk, m) \to c$: Given a public key pk and a plaintext message $m \in \mathcal{M}$, outputs a ciphertext $c \in \mathcal{C}$.
- $\Pi_{FHE}.\mathsf{Dec}(sk, c) \to m$: Given a secret key $sk \in \mathcal{K}$ and a ciphertext $c \in \mathcal{C}$, outputs the plaintext $m \in \mathcal{M}$.
- $\Pi_{FHE}.\mathsf{Eval}(pk, C, (c_1, \ldots, c_n)) \to c$: Given a public key pk, a circuit $C \in \mathbb{C}_n$, and ciphertexts $(c_1, \ldots, c_n) \in \mathcal{C}^n$ for some $n \in \mathbb{N}$, outputs a ciphertext $c \in \mathcal{C}$.

An FHE scheme should satisfy the standard definition of correctness (decryption and evaluation correctness), which we recall in Appendix B. For our setting, we define IK-IND-CPA security to guarantee not only standard IND-CPA security, but also that an adversary cannot distinguish whether the challenge ciphertext was produced under pk_0 or pk_1, even in the presence of encrypted secret keys. This prevents an adversary from learning the intended recipient (i.e., the message destination) from the ciphertext alone. We use the IK-IND-CPA security notion of Chow et al. [14] adapted to our setting.

Definition 3.2 (IK-IND-CPA Security)**.** Let Π_{FHE} be an FHE scheme such that $\mathcal{K} \subseteq \mathcal{M}$. The scheme offers *IK-IND-CPA-security* (Indistinguishability of Keys and Messages) if for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$:

$$\mathbf{Adv}^{\text{ik-cpa}}_{\Pi_{FHE},\mathcal{A}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right| \leq \text{negl}(\lambda).$$

The experiment proceeds as follows:

1. **Setup:** The challenger generates key pairs $(pk_0, sk_0) \leftarrow \Pi_{FHE}.\mathsf{KGen}(1^\lambda)$ and $(pk_1, sk_1) \leftarrow \Pi_{FHE}.\mathsf{KGen}(1^\lambda)$. It also computes encrypted secret keys $c_{sk,0} \leftarrow \Pi_{FHE}.\mathsf{Enc}(pk_0, sk_0)$ and $c_{sk,1} \leftarrow \Pi_{FHE}.\mathsf{Enc}(pk_1, sk_1)$ and gives $(pk_0, c_{sk,0}, pk_1, c_{sk,1})$ to $\mathcal{A}_1$.
2. **Challenge Query:** $\mathcal{A}_1$ outputs two messages $m_0, m_1 \in \mathcal{M}$ and state information st.
3. **Encryption:** The challenger samples a random bit $b \leftarrow \{0,1\}$. It encrypts message m_b under key pk_b to compute $c^* \leftarrow \Pi_{FHE}.\mathsf{Enc}(pk_b, m_b)$.
4. **Guess:** $\mathcal{A}_2$ receives (st, c^*) and outputs a guess b'.

Given that in our protocol, public keys also include the *encryption* of the corresponding secret keys, we additionally require that exposing these encryptions does not compromise the confidentiality of ciphertexts produced by $\Pi_{FHE}.\mathsf{Enc}$. This is captured by Circular Security (Definition 3.3).

Definition 3.3 (Circular Security)**.** Let Π_{FHE} be an FHE scheme such that $\mathcal{K} \subseteq \mathcal{M}$. The scheme is *Circular Secure* if for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$:

$$\mathbf{Adv}^{\text{circ}}_{\Pi_{FHE},\mathcal{A}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right| \leq \text{negl}(\lambda).$$

The experiment proceeds as follows:

1. **Setup:** The challenger runs $(pk, sk) \leftarrow \Pi_{FHE}.\mathsf{KGen}(1^\lambda)$ and computes an encryption of the secret key $c_{sk} \leftarrow \Pi_{FHE}.\mathsf{Enc}(pk, sk)$. It gives (pk, c_{sk}) to $\mathcal{A}_1$.
2. **Challenge Query:** $\mathcal{A}_1$ outputs two messages $m_0, m_1 \in \mathcal{M}$ and state information st.
3. **Encryption:** The challenger samples a random bit $b \leftarrow \{0,1\}$ and computes $c^* \leftarrow \Pi_{FHE}.\mathsf{Enc}(pk, m_b)$.
4. **Guess:** $\mathcal{A}_2$ receives (st, c^*) and outputs a guess b'.

Standard notions of correctness for FHE are typically guaranteed only over the space of *well-formed ciphertexts*, denoted as $\mathcal{C}_{pk}$. This space consists of all ciphertexts generated by valid encryption or homomorphic evaluation under a specific public key pk (we recall the formal definition in Appendix B). However, the full ciphertext space $\mathcal{C}$ is a superset (i.e., $\mathcal{C} \supseteq \bigcup_{pk} \mathcal{C}_{pk}$ for all $pk : (pk, sk) \leftarrow \Pi_{FHE}.\mathsf{KGen}(1^\lambda)$) containing invalid ciphertexts or those encrypted under different keys. While standard decryption $\Pi_{FHE}.\mathsf{Dec}(sk, \cdot)$ is well-behaved on $\mathcal{C}_{pk}$, it is not necessarily well-defined over the entirety of $\mathcal{K} \times \mathcal{C}$.

In our scheme, the server applies the homomorphic evaluation algorithm under a public key pk_A to ciphertexts originally encrypted under a different key pk_B. Despite this key mismatch, the evaluation must still output a syntactically valid ciphertext rather than failing or producing undefined errors. This requires a common FHE property called *Strong Correctness* which we recall in Definition 3.4. Whereas standard correctness only concerns ciphertexts in $\mathcal{C}_{pk}$, Strong Correctness requires the decryption algorithm to be *well defined* over all of $\mathcal{K} \times \mathcal{C}$, returning some message in $\mathcal{M}$ even when decrypting under the wrong key or decrypting an arbitrary bitstring.[3] It further requires that evaluation correctness continue to hold even when input ciphertexts lie outside $\mathcal{C}_{pk}$.

Definition 3.4 (Strong Correctness)**.** An FHE scheme is *strongly correct* if it satisfies the following properties:

1. *Decryption Correctness:* Standard correctness holds for well-formed ciphertexts. For all $(pk, sk) \leftarrow \Pi_{FHE}.\mathsf{KGen}(1^\lambda)$ and all $m \in \mathcal{M}$:
$$\Pr[\Pi_{FHE}.\mathsf{Dec}(sk, \Pi_{FHE}.\mathsf{Enc}(pk, m)) = m] \geq 1 - \text{negl}(\lambda).$$
2. *Well-Defined Decryption:* The decryption algorithm is well-defined over the entire space $\mathcal{K} \times \mathcal{C}$. That is, for any secret key $sk \in \mathcal{K}$ and any (possibly invalid) ciphertext $c \in \mathcal{C}$, there exists a message $m \in \mathcal{M}$ such that: $\Pi_{FHE}.\mathsf{Dec}(sk, c) = m$.
3. *Strong Evaluation Correctness:* Evaluation correctness holds for *all* ciphertexts in $\mathcal{C}$, including those not in $\mathcal{C}_{pk}$. For any $(pk, sk) \leftarrow \Pi_{FHE}.\mathsf{KGen}(1^\lambda)$, any circuit $C \in \mathbb{C}_n$, and any ciphertext vector $\mathbf{c} = (c_1, \ldots, c_n) \in \mathcal{C}^n$, if we let $m_i = \Pi_{FHE}.\mathsf{Dec}(sk, c_i)$ for all $i \in [n]$, then: $\Pr[\Pi_{FHE}.\mathsf{Dec}(sk, c') = C(m_1, \ldots, m_n)] \geq 1 - \text{negl}(\lambda)$, where $c' \leftarrow \Pi_{FHE}.\mathsf{Eval}(pk, C, \mathbf{c})$.

Formally, an FHE scheme Π_{FHE} is *bootstrappable* if it can homomorphically evaluate its own decryption circuit (augmented by a standard logic gate) [21]. This capability enables a `Recrypt` mechanism that "refreshes" a ciphertext c by evaluating the decryption circuit $\mathsf{Dec}(\cdot, c)$ on an encryption of the secret key c_{sk} (often called the bootstrapping key):

$$c' \leftarrow \Pi_{FHE}.\mathsf{Eval}(pk, \mathsf{Dec}(\cdot, c), c_{sk})$$

[3] The output of a wrong-key decryption is formally a valid element of $\mathcal{M}$ but has no semantic relation to the original plaintext.

Strong Correctness ensures that decryption and evaluation are well defined even when a ciphertext is not well formed under the evaluation key. In our construction, the server bootstraps a ciphertext using receiver r's bootstrapping key even when the ciphertext was generated under a different receiver's key. To bound false positives in the tag-matching step, we require that such wrong-key bootstrapped decryption yields a plaintext that is (almost) uniformly distributed over $\mathcal{M}$. We therefore assume the wrong-key decryption property used in OMR [36], formalized in Definition 3.5.

Definition 3.5 (Wrong-Key Decryption)**.** Let Π_{FHE} be a bootstrappable FHE scheme with plaintext space $\mathcal{M}$. We say that Π_{FHE} satisfies *wrong-key decryption* if decrypting a ciphertext under an independently generated, unrelated secret key yields an (almost) uniformly random plaintext. Formally, for $(pk, sk) \leftarrow \Pi_{FHE}.\mathsf{KGen}(1^\lambda)$ and $(pk', sk') \leftarrow \Pi_{FHE}.\mathsf{KGen}(1^\lambda)$, any message $m \in \mathcal{M}$, and ciphertext $c \leftarrow \Pi_{FHE}.\mathsf{Enc}(pk, m)$, we define:

$$c^\star \leftarrow \Pi_{FHE}.\mathsf{Eval}(pk', \mathsf{Dec}(\cdot, c), c_{sk'}),$$

where $c_{sk'} \leftarrow \Pi_{FHE}.\mathsf{Enc}(pk', sk')$.
Let $m^\star \leftarrow \Pi_{FHE}.\mathsf{Dec}(sk', c^\star)$. Then for all $\hat{m} \in \mathcal{M}$:

$$\Pr[m^\star = \hat{m}] \leq \frac{1}{|\mathcal{M}|} + \mathrm{negl}(\lambda).$$

While the requirement for Strong Correctness is stringent, Chow et al. [14] demonstrate that any FHE scheme Π_{FHE} that is (1) circular-secure and (2) has a well-defined decryption algorithm over $\mathcal{K} \times \mathcal{C}$ can be generically transformed into a strongly correct scheme Π'_{FHE}. This is achieved by defining an augmented evaluation algorithm that performs a bootstrapping step prior to evaluating the circuit C. We recall this design in Appendix B.

3.2 Controlled Multiplexer (CMux)

The Controlled Multiplexer (CMux) gate is a fundamental primitive in FHE-based schemes that implements homomorphic conditional selection [12]. It serves as the encrypted equivalent of an `if-else` statement.

Definition 3.6 (CMux Gate)**.** Let c_s be a ciphertext encrypting a selector bit $s \in \{0,1\}$, and let $c_0, c_1 \in \mathcal{C}$ be ciphertexts encrypting messages $m_0, m_1 \in \mathcal{M}$, respectively. The algorithm $\mathsf{CMux}(c_s, c_0, c_1)$ outputs a ciphertext c' such that:

$$\Pi_{FHE}.\mathsf{Dec}(sk, c') = \begin{cases} m_0 & \text{if } s = 0 \\ m_1 & \text{if } s = 1 \end{cases}$$

Arithmetically, this is typically evaluated as: $\mathsf{CMux}(c_s, c_0, c_1) := c_s \cdot c_1 + (1 - c_s) \cdot c_0$ where the addition and multiplication operations are performed homomorphically.

4 Model and Definitions

4.1 Model

We consider a single-server asynchronous messaging service with senders S, receivers R, and a server Srv. Each receiver $r \in R$ generates a key pair (pk_r, sk_r) and registers pk_r with Srv. For each registered receiver, the server maintains an encrypted inbox INBOX and an encrypted inbox tag iTag. We write $\mathsf{POSTBOX}$ for the database of per-receiver tuples $(\mathsf{INBOX}, \mathsf{iTag}, pk)$.

Senders submit messages in batches. A batch request $\mathsf{B_{req}}$ is an ordered sequence of g ciphertext pairs

$$\mathsf{B_{req}} = ((\mathsf{signal}_1, \mathsf{pTag}_1), \ldots, (\mathsf{signal}_g, \mathsf{pTag}_g)),$$

where $1 \leq g \leq \bar{g}$. We require that the g entries in a batch target g distinct receivers. Formally, there exists an injective mapping $f : [g] \to R$ assigning each entry to its intended receiver. Let $R_g = \{f(i)\}_{i \in [g]}$ denote the set of targeted receivers.

We model each signal as an encryption of an encoded message $msg \in \mathcal{M}$ and each pTag as an encryption of a fixed pertinency tag in $\{0,1\}^\ell \subseteq \mathcal{M}$ (in our construction, the all-ones string 1^ℓ). We assume messages are encoded injectively into $\mathcal{M}$ (for example, by including a random nonce), so collisions between distinct deliveries occur only with negligible probability. The definitions can be generalized to handle duplicates by including an explicit message identifier. We also reserve a distinguished dummy message value 1 used only for inbox initialization, distinct from the tag 1^ℓ. We assume real messages are encoded to avoid this value.

The public parameters PP include the inbox capacity ν (the number of slots in each INBOX), the tag length ℓ, the maximum batch size $\bar{g}$, and error parameters ε_p and ε_c. We interpret ε_p as an upper bound on the probability that a batch entry intended for a different receiver is mistakenly deemed pertinent when processed against a receiver's inbox tag. The parameter ε_c bounds the probability of a collision for a targeted receiver: that at least one additional entry in the same batch is also deemed pertinent, which can corrupt the inserted plaintext. The algorithm $\mathit{ParamsGen}(1^\lambda, \varepsilon_p, \varepsilon_c, \nu) \to PP$ chooses ℓ and $\bar{g}$ as functions of ε_p and ε_c so that this collision probability is at most ε_c. In the asymptotic definitions below, we treat $\varepsilon_p = \varepsilon_p(\lambda)$ as negligible in λ (equivalently, we choose $\ell = \ell(\lambda)$ so that false positives are negligible), and we restrict attention to polynomially bounded executions.

Since each inbox stores only ν ciphertext slots, our completeness guarantee applies to messages that are among the most recent ν deliveries to the receiver. Older messages may be evicted by the fixed-size inbox.

4.1.1 Threat Model

Adversarial server. The adversary is computationally bounded and controls the server Srv. It observes public parameters, all registered public keys, all sender batch requests, and

the encrypted inbox state stored on *Srv*. The adversary also controls the delivery schedule, including the order in which batches are processed.

Correctness vs. privacy. For completeness and soundness (Definitions 4.2 and 4.3), we assume that *Srv*, senders, and receivers follow the protocol specification. For receiver privacy (Definition 4.4), we allow *Srv* to be malicious and to use arbitrary auxiliary information independent of the challenge linkage.

Communication channels. We assume sender-to-server communication is carried out over an anonymous channel so that network identifiers do not trivially reveal the sender. Receiver-to-server communication is end-to-end encrypted and authenticated, enabling the server to maintain per-receiver state and prevent trivial impersonation attacks.

Leakage. We allow the server to learn unavoidable system-level information needed to operate the service: the public parameters (e.g., $\lambda, \ell, \bar{g}, \nu$), the set of registered receivers and their public keys, the batch size $|\mathsf{B}_{\mathsf{req}}|$, and the timing of batch submissions and receiver reads. Beyond this explicit leakage, *Srv* learns no additional information about which receivers are targeted by incoming batch entries.

4.2 Definitions

Definition 4.1 (Oblivious Signaling)**.** An Oblivious Signaling (Π_{OS}) scheme consists of the following PPT algorithms. The public parameters *PP* are an implicit input to all algorithms.

- $\mathit{ParamsGen}(1^\lambda, \varepsilon_p, \varepsilon_c, \nu) \to \mathit{PP}$: Takes as input the security parameter λ, a target false positive rate ε_p, a collision probability bound ε_c, and an inbox capacity ν. It outputs public parameters *PP* that include ℓ and $\bar{g}$ derived from ε_p and ε_c.
- $\mathit{KeyGen}() \to (pk, sk)$: Run by a receiver. Outputs a public key pk and a secret key $sk \in \mathcal{K}$. In our construction, pk includes an FHE public key and an encrypted secret key used to enable homomorphic evaluation of decryption.
- $\mathit{InboxGen}(pk) \to (\mathsf{INBOX}, \mathsf{iTag})$: Takes a public key pk as input. Outputs an initialized inbox structure $\mathsf{INBOX} \in \mathcal{C}^\nu$ and an associated inbox tag $\mathsf{iTag} \in \mathcal{C}$.
- $\mathit{SignalGen}(PK, M) \to (\mathsf{B}_{\mathsf{req}})$: Takes as input a vector of recipient public keys $PK = (pk_1, \ldots, pk_g)$ and a vector of plaintext messages $M = (msg_1, \ldots, msg_g)$, where $1 \le g \le \bar{g}$ and $pk_i \neq pk_j$ for all $i \neq j$. It outputs a request batch

$$\mathsf{B}_{\mathsf{req}} = ((\mathsf{signal}_1, \mathsf{pTag}_1), \ldots, (\mathsf{signal}_g, \mathsf{pTag}_g)),$$

  where each $\mathsf{signal}_i \in \mathcal{C}$ encrypts msg_i under the recipient's FHE public key component of pk_i, and each $\mathsf{pTag}_i \in \mathcal{C}$ encrypts the corresponding pertinency tag (in our construction, the all-ones string 1^ℓ) under the same component.
- $\mathit{AddSigToInbox}(pk, \mathsf{INBOX}, \mathsf{iTag}, \mathsf{B}_{\mathsf{req}}) \to \mathsf{INBOX}'$: Run by the server for a given receiver. Takes as input pk, the associated INBOX, the inbox tag iTag, and a batch $\mathsf{B}_{\mathsf{req}}$. It outputs an updated inbox state INBOX'. It homomorphically tests pertinency of each batch entry against iTag and conditionally updates INBOX.
- $\mathit{ReadInbox}(sk, \mathsf{INBOX}) := (\{msg_1, \ldots, msg_\nu\})$: Takes a secret key sk and an inbox INBOX as input. Outputs a length-ν vector $MV \in \mathcal{M}^\nu$ of decrypted slot contents; dummy entries can be discarded by the receiver.

An Π_{OS} scheme must satisfy completeness, soundness, and receiver-message unlinkability, defined below. Appendix D gives the correctness experiment used for completeness and soundness.

Definition 4.2 (Completeness)**.** Let $\varepsilon_p = \varepsilon_p(\lambda)$ be negligible in λ, and let $PP \leftarrow \mathit{ParamsGen}(1^\lambda, \varepsilon_p, \varepsilon_c, \nu)$. Consider any honest sequence of sender-generated batch requests $\mathsf{Q}_{recv} = (\mathsf{B}^1_{\mathsf{req}}, \ldots, \mathsf{B}^m_{\mathsf{req}})$ such that $\sum_{j=1}^m |\mathsf{B}^j_{\mathsf{req}}| \le \mathit{poly}(\lambda)$. Assume each batch $\mathsf{B}^j_{\mathsf{req}}$ satisfies $|\mathsf{B}^j_{\mathsf{req}}| \le \bar{g}$ and targets distinct receivers.

Fix any receiver $r \in R$ with key pair (pk, sk) and initial state $(\mathsf{INBOX}^{init}, \mathsf{iTag}) \leftarrow \mathit{InboxGen}(pk)$. Let INBOX^{final} be the inbox state obtained by processing every batch in Q_{recv} for receiver r by iterating $\mathsf{INBOX} \leftarrow \mathit{AddSigToInbox}(pk, \mathsf{INBOX}, \mathsf{iTag}, \mathsf{B}^j_{\mathsf{req}})$ over $j = 1, \ldots, m$. Let $MV \leftarrow \mathit{ReadInbox}(sk, \mathsf{INBOX}^{final})$.

Fix any index $j_0 \in \{1, \ldots, m\}$ and any message $m_{sent} \in \mathcal{M}$ sent to r in batch $\mathsf{B}^{j_0}_{\mathsf{req}}$. Assume that m_{sent} is among the most recent ν deliveries to r within the sequence. The scheme is *complete* if:

$$\Pr[m_{sent} \text{ appears in } MV] \ge (1 - \varepsilon_c) - \mathrm{negl}(\lambda).$$

Definition 4.3 (Soundness)**.** Let *PP* and Q_{recv} be as in Definition 4.2, and fix any receiver $r \in R$ with key pair (pk, sk) and initial state $(\mathsf{INBOX}^{(0)}, \mathsf{iTag}) \leftarrow \mathit{InboxGen}(pk)$. For $j = 1, \ldots, m$, let $\mathsf{INBOX}^{(j)}$ denote the inbox state after processing the first j batches in Q_{recv}, i.e.,

$$\mathsf{INBOX}^{(j)} \leftarrow \mathit{AddSigToInbox}\big(pk, \mathsf{INBOX}^{(j-1)}, \mathsf{iTag}, \mathsf{B}^j_{\mathsf{req}}\big).$$

For each $j \in \{1, \ldots, m\}$, let $R^j_g \subseteq R$ denote the set of receivers targeted by batch $\mathsf{B}^j_{\mathsf{req}}$. Fix any index $j \in \{1, \ldots, m\}$ such that $r \notin R^j_g$. Let $MV^{(j-1)} \leftarrow \mathit{ReadInbox}(sk, \mathsf{INBOX}^{(j-1)})$ and $MV^{(j)} \leftarrow \mathit{ReadInbox}(sk, \mathsf{INBOX}^{(j)})$.

The scheme is *sound* if

$$\Pr\big[MV^{(j)} \neq MV^{(j-1)}\big] \le \mathrm{negl}(\lambda).$$

Definition 4.4 (Receiver-Message Unlinkability ($RM\bar{L}$))**.** The $RM\bar{L}$ game is played between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$ controlling the server.

1. **Setup:** The challenger samples $PP \leftarrow \mathit{ParamsGen}(1^\lambda, \varepsilon_p, \varepsilon_c, \nu)$. It generates key pairs $(pk_0, sk_0) \leftarrow \mathit{KeyGen}()$ and $(pk_1, sk_1) \leftarrow \mathit{KeyGen}()$,

and inbox states $(\mathsf{INBOX}_0, \mathsf{iTag}_0) \leftarrow \mathit{InboxGen}(pk_0)$ and $(\mathsf{INBOX}_1, \mathsf{iTag}_1) \leftarrow \mathit{InboxGen}(pk_1)$. It gives PP and $(pk_0, \mathsf{INBOX}_0, \mathsf{iTag}_0, pk_1, \mathsf{INBOX}_1, \mathsf{iTag}_1)$ to $\mathcal{A}$.

2. **Challenge:** $\mathcal{A}$ chooses a plaintext message $msg \in \mathcal{M}$. The challenger samples $b \leftarrow \{0,1\}$ and forms a single-entry batch by running: $\mathsf{B}_{\mathsf{req}} \leftarrow \mathit{SignalGen}((pk_b),(msg))$. It sends $\mathsf{B}_{\mathsf{req}}$ to $\mathcal{A}$.
3. **Observation:** $\mathcal{A}$ obtains the updated inboxes by running the protocol-defined server update for each $j \in \{0,1\}$:

$$\mathsf{INBOX}'_j \leftarrow \mathit{AddSigToInbox}\big(pk_j, \mathsf{INBOX}_j, \mathsf{iTag}_j, \mathsf{B}_{\mathsf{req}}\big).$$

4. **Guess:** $\mathcal{A}$ outputs a bit b'.

We say that Π_{OS} satisfies *Receiver-Message Unlinkability* if for any PPT adversary $\mathcal{A}$,

$$\left|\Pr[b' = b] - \frac{1}{2}\right| \leq \mathrm{negl}(\lambda).$$

Remark 4.1. We state the $RM\overline{L}$ game for a single-entry batch, where the challenge message is explicitly fixed. This captures *routing privacy* (hiding the message destination) rather than message privacy; message confidentiality follows independently from the IND-CPA security of the underlying FHE scheme. By a hybrid argument over batch entries, indistinguishability for single-entry batches implies indistinguishability for any batch of size g, with at most a factor-g loss in the adversary's advantage. Repeating the game for a polynomial number of single-entry challenges incurs only a polynomial loss, preserving negligible advantage.

5 Construction

In this section, we present OSig, an Oblivious Signaling construction based on a Fully Homomorphic Encryption (FHE) scheme Π_{FHE} supporting Boolean circuit evaluation and satisfying the *IK-IND-CPA Security*, *Strong Correctness*, *Circular Security*, and *Wrong-Key Decryption* properties from Section 3. The server maintains an encrypted inbox for each recipient and homomorphically processes incoming batches to obliviously insert only the pertinent messages.

OSig consists of receiver setup and inbox initialization, sender-side batch generation, server-side oblivious inbox updates, and receiver-side inbox decryption. Each receiver runs *KeyGen* to obtain an FHE key pair and registers an encrypted inbox INBOX together with an encrypted inbox tag iTag (Algorithms 1 and 2). To send messages, a sender runs *SignalGen* to produce a batch request $\mathsf{B}_{\mathsf{req}}$ consisting of ciphertext pairs $(\mathsf{signal}_i, \mathsf{pTag}_i)$ (Algorithm 3). Given $\mathsf{B}_{\mathsf{req}}$, the server invokes *AddSigToInbox* for each registered receiver to homomorphically test pertinency and obliviously update the corresponding inbox: if a signal is *pertinent*, it is inserted into INBOX; otherwise, the plaintext inbox contents are unchanged (Algorithm 4). Finally, each receiver periodically runs *ReadInbox* to decrypt INBOX and recover received messages (Algorithm 5).

Algorithm 1 OSig Setup and Key Generation

procedure *ParamsGen*$(1^\lambda, \varepsilon_p, \varepsilon_c, \nu)$
  $\ell = \lceil \log(\varepsilon_p^{-1}) \rceil$
  $\bar{g} = \left\lfloor 1 + \frac{\varepsilon_c}{\varepsilon_p} \right\rfloor$
  $PP_{\mathsf{FHE}} \leftarrow \Pi_{FHE}.\mathsf{ParamsGen}(1^\lambda)$
  return $PP = (1^\lambda, \ell, \bar{g}, \varepsilon_p, \varepsilon_c, \nu, PP_{\mathsf{FHE}})$
end procedure
procedure *KeyGen*( )
  $(\mathsf{pk}_{\mathsf{fhe}}, \mathsf{sk}_{\mathsf{fhe}}) \leftarrow \Pi_{FHE}.\mathsf{KGen}(1^\lambda)$
  $c_{sk} \leftarrow \Pi_{FHE}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{fhe}}, \mathsf{sk}_{\mathsf{fhe}})$
  return $pk = (\mathsf{pk}_{\mathsf{fhe}}, c_{sk}), \quad sk = \mathsf{sk}_{\mathsf{fhe}}$
end procedure

Algorithm 2 OSig Inbox Initialization

procedure *InboxGen*(pk)
  $(\mathsf{pk}_{\mathsf{fhe}}, c_{sk}) \leftarrow pk$
  for all $i \in [\nu]$ **do**
    $\mathsf{c}_i \leftarrow \Pi_{FHE}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{fhe}}, 1)$ ▷ dummy slot
  end for
  $\mathsf{INBOX} \leftarrow (\mathsf{c}_1, \ldots, \mathsf{c}_\nu)$
  $\mathsf{iTag} \leftarrow \Pi_{FHE}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{fhe}}, 1^\ell)$ ▷ all-ones tag
  return INBOX, iTag
end procedure

Setup and Key Generation (Algorithm 1). The system parameters are established via *ParamsGen*, which derives the tag length ℓ and maximum size of batch $\bar{g}$ as a function of the desired false positive rate ε_p and collision bound ε_c.

Each receiver then generates a key pair via *KeyGen*. Crucially, the public key pk contains two components: (i) a standard FHE public key $\mathsf{pk}_{\mathsf{fhe}}$, and (ii) an encryption of the corresponding secret key $c_{sk} \leftarrow \Pi_{FHE}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{fhe}}, \mathsf{sk}_{\mathsf{fhe}})$. This encrypted secret key enables the server to homomorphically *evaluate the decryption function* (i.e., bootstrap) on ciphertexts without learning any underlying plaintext. This requires circular security, since *KeyGen* publishes an encryption of the FHE secret key under the corresponding public key.

Inbox Initialization (Algorithm 2). A receiver initializes its storage via *InboxGen*. An INBOX consists of ν encrypted slots, each initialized to an encryption of a fixed dummy value (denoted by 1). The receiver also generates iTag, an encryption of the all-ones tag 1^ℓ, later used by the server to homomorphically test whether an incoming signal is pertinent.

Message Generation (Algorithm 3). To send messages, a sender runs *SignalGen* to construct a batch request $\mathsf{B}_{\mathsf{req}}$. For each intended recipient i, the sender encrypts the message msg_i under the recipient's FHE public key to obtain signal_i, and attaches a pertinency tag pTag_i (an encryption of the all-ones string 1^ℓ) that will later be used by the server in the homomorphic pertinency test (without revealing the intended

Algorithm 3 OSig Message Generation

procedure *SignalGen*(PK, M)
  $g \leftarrow |PK|$ ▷ number of recipients in the batch
  for $i \leftarrow 1$ **to** g **do**
    $(pk_{fhe}^{(i)}, c_{sk}^{(i)}) \leftarrow PK[i]$
    $msg_i \leftarrow M[i]$
    $\mathsf{pTag}_i \leftarrow \Pi_{FHE}.\mathsf{Enc}(pk_{fhe}^{(i)}, 1^{\ell})$
    $\mathsf{signal}_i \leftarrow \Pi_{FHE}.\mathsf{Enc}(pk_{fhe}^{(i)}, msg_i)$
    $\mathsf{B_{req}}[i] \leftarrow (\mathsf{signal}_i, \mathsf{pTag}_i)$
  end for
  return $\mathsf{B_{req}} = ((\mathsf{signal}_1, \mathsf{pTag}_1), \ldots, (\mathsf{signal}_g, \mathsf{pTag}_g))$
end procedure

recipient). The sender then sends the resulting ordered sequence of ciphertext pairs $\mathsf{B_{req}}$ to the server. By definition of $\mathsf{B_{req}}$, each batch targets distinct receivers, so there is at most one ciphertext pair $(\mathsf{signal}_i, \mathsf{pTag}_i)$ per receiver in $\mathsf{B_{req}}$.

Server Processing (Algorithm 4). Upon receiving a batch request $\mathsf{B_{req}}$, the server processes it against all INBOX via *AddSigToInbox*, which evaluates a fixed homomorphic circuit and performs a uniform update for every receiver inbox. Concretely, *AddSigToInbox* consists of four FHE steps:

1. **Rectification:** Using the encrypted secret key c_{sk} in pk, the server homomorphically evaluates the FHE decryption circuit on incoming signal and pTag values. This yields "rectified" ciphertexts $(\mathsf{signal}', \mathsf{pTag}')$ under $\mathsf{pk_{fhe}}$, which can be used as inputs to subsequent homomorphic logic.
2. **Pertinency Check:** The server evaluates an equality circuit between the rectified tag pTag' and the receiver's inbox tag iTag, producing a Pertinency Indicator $\mathsf{PI}_i \in \mathcal{C}$, that encrypts 1 iff the batch entry is intended for the target recipient (and 0 otherwise).
3. **Filtering:** The server maintains an accumulator sigAcc initialized to an encryption of 0. For each entry, it uses PI_i as the selector bit in a CMux gate, overwriting sigAcc with signal'_i iff PI_i encrypts 1. Under the uniqueness constraint that at most one batch entry is pertinent per recipient, sigAcc becomes an encryption of the pertinent message if it exists, and otherwise remains an encryption of 0.
4. **Inbox Update:** Finally, the server updates the INBOX via a CMux-based shift register controlled by the cumulative Pertinency Flag PF. If PF encrypts 1, sigAcc is inserted at the head of the INBOX and existing slots are shifted; otherwise, the plaintext inbox contents are unchanged.

Message Reading (Algorithm 5). To retrieve delivered messages, a receiver periodically runs *ReadInbox* on its current INBOX using the secret key sk. This procedure locally decrypts all ν encrypted slots in the INBOX and outputs the corresponding plaintext message vector MV. Dummy entries (from initialization) decrypt to the fixed dummy value and can be discarded by the receiver.

Algorithm 4 OSig Server Processing

procedure *AddSigToInbox*$(pk, \mathsf{INBOX}, \mathsf{iTag}, \mathsf{B_{req}})$
  $(\mathsf{pk_{fhe}}, c_{sk}) \leftarrow pk$
  $(\mathsf{c}_1, \ldots, \mathsf{c}_\nu) \leftarrow \mathsf{INBOX}$
  $\mathsf{sigAcc} \leftarrow \Pi_{\mathsf{FHE}}.\mathsf{Enc}(\mathsf{pk_{fhe}}, 0)$
  $\mathsf{PF} \leftarrow \Pi_{\mathsf{FHE}}.\mathsf{Enc}(\mathsf{pk_{fhe}}, 0)$
  for $i \leftarrow 1$ **to** $|\mathsf{B_{req}}|$ **do**
    $(\mathsf{signal}_i, \mathsf{pTag}_i) \leftarrow \mathsf{B_{req}}[i]$
  ▷ *Step 1: Rectification*
    $\mathsf{signal}'_i \leftarrow \Pi_{\mathsf{FHE}}.\mathsf{Eval}(\mathsf{pk_{fhe}}, \mathsf{Dec}(\cdot, \mathsf{signal}_i), c_{sk})$
    $\mathsf{pTag}'_i \leftarrow \Pi_{\mathsf{FHE}}.\mathsf{Eval}(\mathsf{pk_{fhe}}, \mathsf{Dec}(\cdot, \mathsf{pTag}_i), c_{sk})$
  ▷ *Step 2: Pertinency check*
    $\mathsf{PI}_i \leftarrow \Pi_{\mathsf{FHE}}.\mathsf{Eval}(\mathsf{pk_{fhe}}, \mathsf{EQUAL}, (\mathsf{iTag}, \mathsf{pTag}'_i))$
  ▷ *Step 3: Filter and accumulate*
    $\mathsf{sigAcc} \leftarrow \mathsf{CMux}(\mathsf{PI}_i, \mathsf{sigAcc}, \mathsf{signal}'_i)$
    $\mathsf{PF} \leftarrow \Pi_{\mathsf{FHE}}.\mathsf{Eval}(\mathsf{pk_{fhe}}, \mathsf{OR}, (\mathsf{PF}, \mathsf{PI}_i))$
  end for
  ▷ *Step 4: Inbox update*
  $\mathsf{c}'_1 \leftarrow \mathsf{CMux}(\mathsf{PF}, \mathsf{c}_1, \mathsf{sigAcc})$
  for $j \leftarrow 2$ **to** ν **do**
    $\mathsf{c}'_j \leftarrow \mathsf{CMux}(\mathsf{PF}, \mathsf{c}_j, \mathsf{c}_{j-1})$
  end for
  return $\mathsf{INBOX}' = (\mathsf{c}'_1, \ldots, \mathsf{c}'_\nu)$
end procedure

Algorithm 5 OSig Message Reading

procedure *ReadInbox*$(\mathsf{sk}, \mathsf{INBOX})$
  $\mathsf{sk_{fhe}} \leftarrow sk$
  for $i \leftarrow 1$ **to** ν **do**
    $MV[i] \leftarrow \Pi_{FHE}.\mathsf{Dec}(\mathsf{sk_{fhe}}, \mathsf{INBOX}[i])$
  end for
  return MV
end procedure

5.1 Security Analysis

We now analyze the security of Oblivious Signaling and show that OSig achieves the privacy and correctness guarantees defined in Section 4. At a high level, the server processes each batch request $\mathsf{B_{req}}$ by evaluating a fixed homomorphic circuit over every recipient inbox. We prove completeness (Theorem 5.1), soundness (Theorem 5.2), and receiver-message unlinkability (Theorem 5.3). Our correctness proofs (completeness and soundness) rely on the *Strong Correctness* and *Wrong-Key Decryption* properties of Π_{FHE} (Section 3). Our privacy proof (receiver-message unlinkability) additionally relies on the *IK-IND-CPA* and *Circular Security* of Π_{FHE} (Section 3). Below we provide the Theorem statements and refer to Appendix E for the security proofs.

Theorem 5.1 (Completeness)**.** *The* OSig *protocol satisfies Completeness according to Definition 4.2, assuming* Strong Correctness *and* Wrong-Key Decryption *of the underlying*

FHE scheme Π_{FHE} *and that the batch size satisfies* $|\mathsf{B}_{\mathsf{req}}| \leq \bar{g}$.

Theorem 5.2 (Soundness)**.** *The* OSig *protocol satisfies Soundness according to Definition 4.3, assuming* Strong Correctness *and* Wrong-Key Decryption *of* Π_{FHE}.

Theorem 5.3 (Receiver-Message Unlinkability)**.** *The* OSig *protocol satisfies Receiver-Message Unlinkability (RM$\bar{L}$) according to Definition 4.4, assuming the* Circular Security *and* IK-IND-CPA Security *of* Π_{FHE}.

6 Implementation

We instantiate our Oblivious Signaling construction as a research prototype using the TFHE [12] fully homomorphic encryption framework (presented in Appendix B.2). Remark 6.1 discusses our rationale behind this choice.

6.1 Methodology

We implement OSig in Rust [46] on top of TFHE-rs (v1.4.3) [48], using the short-integer API. [4] Our prototype follows the algorithms in Section 5 and evaluates the system across multiple parameter settings. All experiments were run on a Google Cloud Compute Engine `c2d-highcpu-112` instance (112 vCPUs / 56 physical cores, 224 GB RAM). All experiments were repeated at least 5 times; error bars were negligible and are omitted for readability.

Remark 6.1. (Rationale for using TFHE.) TFHE satisfies the *IK-IND-CPA Security*, *Strong Correctness*, *Circular Security*, and *Wrong-Key Decryption* properties required by our construction (Section 5). Because our server must update encrypted INBOXs over an unbounded sequence of executions, leveled schemes like BFV [5, 19] are unsuitable due to their fixed circuit depth limits. We instead employ TFHE, utilizing its programmable bootstrapping (PBS) to refresh ciphertexts dynamically and enable the unbounded computation required for persistent state. We prioritize TFHE over CKKS [11] because our pipeline requires precise integer semantics; while CKKS supports bootstrapping [2], adapting it for exact integers is an active research area [26].

Remark 6.2 (Anonymity Set and Scalability)**.** Implementing a single global POSTBOX serving q recipients yields an anonymity set of size q. However, the server computation cost to deliver a batch of messages from a sender scales linearly in q, as discussed in Section 2.1, which can be prohibitive at web scale. A practical deployment can instead shard the POSTBOX into β disjoint instances, each serving $q_{\mathsf{shard}} = q/\beta$ recipients. Cryptographically and functionally, this sharding is equivalent to running independent instances of the protocol. Senders then submit batches to the appropriate shard, and the server processes each batch only within that shard, reducing computation by roughly a factor of β. This optimization comes with a natural privacy trade-off: the effective anonymity set is reduced from q to q_{shard}.

Application parameters. Our protocol exposes application-level parameters that control the correctness/privacy trade-off: the target system-wide collision rate ε_c, the false-positive rate ε_p, and the derived maximum batch size $\bar{g}$ and tag size ℓ (plaintext size of pTag and iTag). In our experiments, we target the false positive rate $\varepsilon_p = 2^{-32}$, which sets $\ell = 32$ (Section 5). We use $\bar{g} = 25$ ($\varepsilon_c \approx 2^{-27.42}$) for the OMR comparison (Section 6.3) and $\bar{g} = 20$ ($\varepsilon_c \approx 2^{-27.75}$) elsewhere to maintain uniform batch sizes and avoid threading anomalies. Unless stated otherwise, the maximum inbox capacity is $\nu = 4$.

Workload generation. For each experiment, we initialize q receivers, each maintaining an encrypted INBOX with ν slots. We set a shard size $q_{\mathsf{shard}} \leq q$ and allocate the inboxes into different shards. We then generate the same number of messages per receiver and pack them into batches, where each batch request $\mathsf{B}_{\mathsf{req}}$ contains $g \leq \bar{g}$ ciphertext pairs $(\mathsf{signal}_i, \mathsf{pTag}_i)$ targeting g *distinct* receivers. We dedicate one thread per INBOX to process the incoming batches of requests. After running the experiment, we test each receiver's INBOX to see if they received all pertinent messages. In terms of recipient resources, inbox decryption is performed using a single CPU thread to model client hardware.

Internal FHE parameters. Our prototype uses the *shortint* API of the `TFHE-rs` library and instantiates all ciphertexts with the compact public-key variant (Remark A.1) using the KS→PBS evaluation pipeline. In this setting, ciphertexts are first key-switched, and then programmable bootstrapping is performed, which improves evaluation performance at the cost of moderately larger ciphertexts [48]. In the default configuration, each slot in the INBOX and signal stores 2 shortint ciphertexts. Remark A.2 details how these protocol-level objects are realized in code.

We use parameters with LWE dimension $n = 866$, GLWE dimension $k_{GLWE} = 1$, polynomial size $N = 2048$, and Gaussian noise standard deviations $\sigma_{\mathsf{LWE}} = 2.046151696979124 \cdot 10^{-6}$ and $\sigma_{\mathsf{GLWE}} = 2.845267479601915 \cdot 10^{-15}$. Programmable bootstrapping uses decomposition $(\mathsf{base\_log}, \mathsf{level}) = (23, 1)$ and key switching $(3, 5)$. As a base case, we use TFHE parameters $\mathsf{MessageModulus} = 4$ and $\mathsf{CarryModulus} = 4$ for plaintext encoding, supporting two message bits and two carry bits per ciphertext. The resulting bootstrapping failure probability is approximately $2^{-128.6}$, and the underlying LWE instances target at least 128-bit concrete security under the lattice estimator [1]. With parameters discussed above, a receiver public key occupies ∼32.9 KB, a receiver secret key ∼23.5 KB, and a sender-generated ciphertext (pTag, signal) ∼16.6

[4] https://github.com/shuhanmirza/oblivious-signaling-usenix26

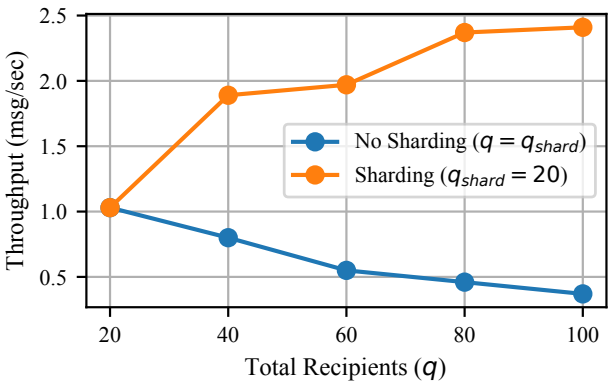

Figure 3: Delivery throughput as the number of recipients q grows (batch size $\bar{g} = 20$). Without sharding, throughput drops with q; sharding (20 inboxes per shard) restores near-linear scaling.

KB, whereas the server evaluation key occupies ∼127.8 MB, reflecting the intentional asymmetry that pushes heavy cryptographic state to the server.

6.2 Experimental Results

Throughput analysis (Delivery). Figure 3 analyzes the throughput of two scaling regimes as the total number of recipients q increases from 20 to 100, with the max batch size fixed at $\bar{g} = 20$. We evaluate delivery throughput as the server workload is parallelized via sharding, where each shard processes a disjoint subset of recipient inboxes. Increasing the number of shards amortizes the cost of the uniform inbox-update circuit across parallel resources. With a single shard, throughput drops from 1.03 to 0.37 messages/sec as the system scales from 20 to 100 recipients, reflecting the linear dependence of delivery cost on the number of inboxes. Under sharding, throughput improves substantially: with 5 shards at 100 recipients, throughput reaches 2.41 messages/sec, a 6.5× gain over the single-shard case. Across shards, we find that the parallel efficiency[5] remains high (0.98–0.99), indicating that OSig's delivery work, dominated by independent homomorphic inbox updates, scales effectively with parallel compute resources.

Runtime breakdown (Delivery). We analyze the server-side runtime composition of the delivery pipeline. Across all configurations, the homomorphic *filter* phase dominates processing time, accounting for roughly 92–98% of total server latency. The CMux-based inbox update contributes a much smaller fraction, typically around 2–3%. Ciphertext expansion and preprocessing remain negligible in most settings (often <1%), though they become slightly more visible (up to 5.8%) in higher-shard configurations due to amortization effects.

[5]Parallel efficiency is the ratio of total computational effort across all shards to the system capacity consumed during the wall-clock execution time, measuring the impact of load imbalance and synchronization overhead.

Recipient Cost Analysis (Retrieval). Retrieval cost is small and stable across all configurations: local inbox decryption takes on average only 97μs. Each check fetches a fixed-size encrypted inbox of $\nu = 4$ slots, so recipient bandwidth and local decryption cost are constant regardless of incoming traffic or workload on the server and the number of recipients q registered in the system. This confirms that OSig makes frequent checking inexpensive relative to delivery.

Impact of Inbox Capacity. To isolate the performance impact of the maximum inbox capacity (ν), we evaluated OSig under a fixed workload while scaling ν from 10 to 100. For this experiment, we maintained a constant recipient population of $q = 100$ and processed $N = 500$ total messages distributed across 25 batches ($\bar{g} = 20$, so $k = (N/q) = 5$ pertinent messages per receiver). As illustrated in Figure 4, although the total processing latency scales linearly with the inbox capacity as asymptotically expected, it grows at a significantly slower rate than the capacity itself. This scaling behavior follows from decomposing the server execution into two components: the *Filter* phase (Steps 1–3 of Algorithm 4) and the *CMux* phase (the oblivious inbox update in Step 4). Because ν dictates the size of the state updated during delivery, the latency of the Filter phase remains independent of the inbox capacity and effectively flat across all configurations. Consequently, the linear increase in total computational latency is driven entirely by the CMux phase, which scales in direct proportion to ν and runs parallel to the overall latency curve. Empirically, increasing ν tenfold (10 to 100) increases latency by only 68% (from 1.62s to 2.71s per message), showing that larger inbox capacities incur relatively small computational overhead while mitigating message overflows.

Impact of Message Size. To evaluate how OSig scales with larger messages, we varied the number of plaintext bits per signal under a fixed workload ($q = 100, N = 200, \bar{g} = 20, k = 2$). Rather than increasing the TFHE message and carry moduli, which would enlarge the underlying look-up tables (LUT) and make programmable bootstrapping (PBS) prohibitively expensive, we scale payload capacity by packing multiple ciphertexts per signal. With the compact public-key instantiation described above, this approach preserves sender-to-server bandwidth: the sender-generated ciphertext grows by a mere 64 bytes (from 16,608 bytes for a 2-bit message to 16,672 bytes for 20 bits). On the server side, Figure 5 illustrates that while processing latency scales linearly with payload size, the computational overhead is shallow. Empirically, a 10× expansion in message bits (from 2 to 20 bits) yields only a 75%

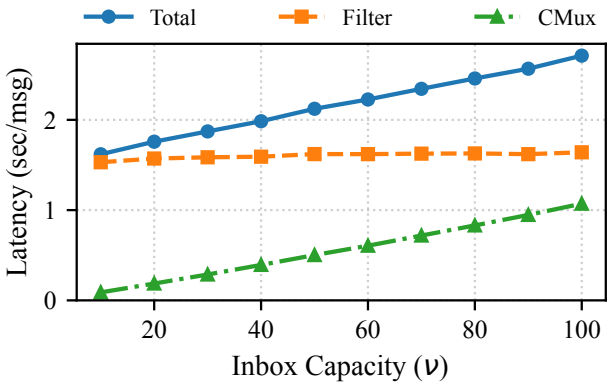

Figure 4: Processing latency of OSig as a function of maximum inbox capacity (ν) under a fixed workload ($q = 100, N = 500, \bar{g} = 20, k = 5$).

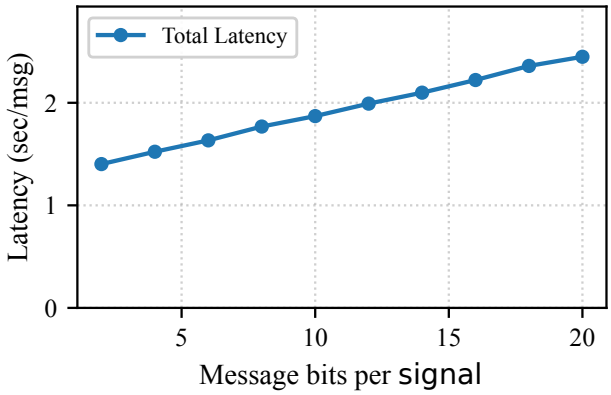

Figure 5: Processing latency per message as a function of payload size (message bits per signal) for OSig. By packing ciphertexts, the system achieves shallow, linear scaling as message capacity increases.

increase in per-message latency (from 1.40s to 2.45s). These results show that OSig can increase direct-inbox payload capacity with moderate additional server latency and almost no sender-to-server bandwidth growth in this parameter range.[6]

Trade-offs in Batch Size Selection. Batching in OSig balances amortized per-message server cost against the target collision rate, ε_c. The maximum batch size $\bar{g}$ is application-specific: it depends on the system's tolerance for collisions and the chosen false-positive rate ε_p, governed by the bound $\bar{g} \leq \lfloor 1 + \varepsilon_c/\varepsilon_p \rfloor$.

To establish a balanced baseline for our OMR comparison in Section 6.3, we analyze the computational inflection point. For q recipients, an inbox capacity ν, and a tag length ℓ, the amortized per-message server cost scales proportionally to $q(C_1 \cdot \ell + C_2 \cdot \nu/\bar{g})$, where C_1 and C_2 are constants representing the cost of the *Filter* phase (Steps 1–3 of Algorithm 4) and the *CMux* phase (the oblivious inbox update in Step 4). While accepting a higher ε_c permits larger batches to amortize the shift cost, increasing $\bar{g}$ yields strict diminishing returns. Empirically, our runtime breakdown shows that the *Filter phase* heavily dominates the *CMux phase* ($C_1 \gg C_2$), meaning the $C_2 \cdot \nu/\bar{g}$ term shrinks rapidly in wall-clock time, causing the total per-message cost to asymptote toward $q \cdot C_1 \cdot \ell$. By analytically modeling this decay in a normalized unitless setting ($q = \nu = C_1 = C_2 = 1$; Figure 6), we anchor our prototype near the normalized "knee" of the curve using the integer batch size $\bar{g} = 25$ (with $\varepsilon_p = 2^{-32}$ and $\varepsilon_c \approx 2^{-27.42}$). This prevents arbitrary over-batching and ensures a principled baseline for our comparative evaluation.

[6] In *signal-mode*, the OSig payload functions as an encrypted pointer to an external bulletin board. Consequently, a payload of X bits can index a bulletin board of up to 2^X entries. For example, the 20-bit payload evaluated here can support a bulletin board containing 2^{20} (over 1 million) messages.

6.3 Comparison with OMR

We compare OSig against InstantOMR [33], the most latency-optimized OMR design to date[7]. For our experiments, we use the authors' public implementation [32].

OSig and OMR have fundamentally different cost models, differing in when computation occurs and who bears the cost. OMR minimizes sender-side delivery cost by shifting substantial computation to recipient retrieval, whereas OSig performs the privacy-preserving computation during message delivery while keeping recipient retrieval constant-cost. Consequently, the meaningful comparison between the two paradigms is their *dominant server-side operation*. We therefore measure oblivious message *delivery* for OSig and message *retrieval* for OMR over N appended bulletin-board messages.

Experimental Setup. InstantOMR, like prior OMR systems, assumes that the recipient provides an upper bound $\bar{k}$ on the number of pertinent messages in the retrieval window. In their evaluation, the authors follow the common optimistic setting $k = \bar{k}$, ensuring no overflow and thus avoiding re-queries. To give InstantOMR the same advantage, we adopt this assumption to construct a baseline configuration, `omr-baseline`.

We also construct a second configuration, `omr-sporadic`, using a driver script built on their codebase to model the *sporadic-receipt* regime where recipients do not know k. To mitigate the leakage caused by adaptive retries after overflow, prior OMR work proposes unlinkable detection keys together with a two-server architecture [35]. We further dis-

[7] Since InstantOMR uses Primus-FHE [43] while OSig uses TFHE-rs [48], the concrete performance gap should not be interpreted as a direct microbenchmark comparison between FHE libraries.

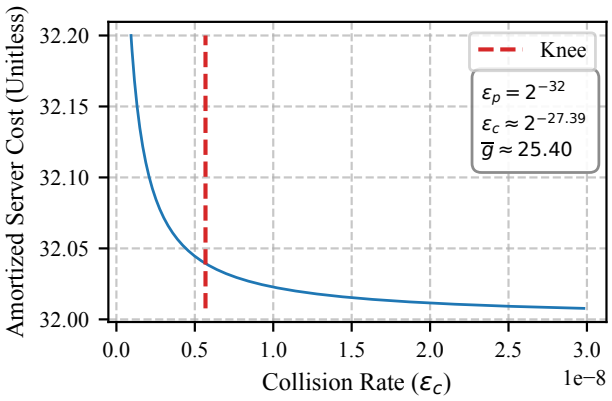

Figure 6: Normalized amortized server cost versus collision rate ε_c in the unitless setting $q = \nu = C_1 = C_2 = 1$, under the model $C_1\ell + C_2\nu/\bar{g}$ with $\varepsilon_p = 2^{-32}$. The curve illustrates diminishing returns from larger batches; its knee near $\bar{g} \approx 25.40$ motivates our choice $\bar{g} = 25$ for comparison.

cuss this in Remark A.3. We incorporate these mitigations in `omr-sporadic`. In this setting, each recipient holds two unlinkable detection keys and interacts with two non-colluding logical servers. The first server performs lightweight detection and returns the encrypted pertinency vector (PV), revealing the exact value of k (Remark A.4) ; the second server is then queried with an over-provisioned bound $\bar{k}' > k$ to construct the digest without overflow or retries. If the PV indicates $k = 0$, the second query is skipped.

For all experiments, we allocate one server CPU thread per recipient for a population of $q = 100$ recipients and vary the workload of N messages. To normalize the workload, we fix the frequency of pertinent messages to $k = N/q$ for each recipient, with their positions distributed uniformly at random. For OSig, messages arrive in batches of size at most $\bar{g} = 25$ with inbox capacity $\nu = 10$. For `omr-baseline`, the retrieval window spans the full workload ($D = N$) with exact bound $\bar{k} = k$. For `omr-sporadic`, the retrieval window is fixed to $D = 100$, requiring N/D queries per recipient. We retain the default cryptographic parameters from [33].

Overflow Semantics. The overflow behaviors of OMR and OSig are fundamentally different and are therefore excluded from this benchmark comparison. In OMR, overflow occurs at query time when the recipient underestimates $\bar{k}$, causing retrieval failure and requiring an adaptive retry, which may leak information about recipient traffic patterns (Section 2.1). In contrast, OSig implements a deterministic FIFO retention policy: once the inbox reaches capacity ν, older messages are evicted without requiring retries or additional interaction.

Cost Model Comparison. The final cost metric is the mean server processing time per recovered pertinent message. For OSig, we divide the average batch processing time by the number of delivered messages in the batch. For OMR, while the server works over scanned bulletin-board entries, we divide the total retrieval time for a window of N messages by the number of pertinent messages recovered from that window. Fixing the pertinent message frequency keeps this normalized cost comparable across both paradigms.

Figure 7 reports the mean server processing time per successful delivery or recovered pertinent message. OSig stays between 1.58 and 1.61 seconds per successful delivery across all tested workloads, because its delivery cost scales with the anonymity set rather than with global traffic. Thus, the sender-side charge is tied to the chosen anonymity set, not to system-wide noise. As shown in our inbox-capacity evaluation, even substantial increases in ν add only modest overhead relative to the retrieval costs incurred by OMR.

In contrast, the computation for pull-based OMR architectures scales directly with global traffic. The `omr-baseline` system incurs an average cost of 52 to 54 seconds per recovered pertinent message, while `omr-sporadic` requires between 87 and 111 seconds. Notably, while `omr-baseline` scans individual bulletin board entries efficiently (taking roughly 0.5 seconds per message), this computation is overwhelmingly wasteful: because 99 out of every 100 scanned messages do not belong to the recipient, the server's cost is heavily penalized by global noise. The OMR cost curves appear flat across varying loads in this experiment solely because we fixed the frequency of pertinent messages. Because OMR requires scanning the entire bulletin board, its cost per recovered pertinent message increases when pertinent messages are sparse and decreases when they are frequent.

Cost-Model Takeaway. Figure 7 shows that the dominant server-side costs occur at different phases in the two architectures. In OMR, message append is cheap, but retrieval requires scanning the unread global board segment. In OSig, the server performs the privacy-preserving work at delivery by uniformly updating the anonymity set; subsequent recipient checks are fixed-size inbox reads.

7 Discussion and Trade-Offs

Denial-of-Service Considerations. In Oblivious Signaling, heavy server-side computation occurs at message delivery, naturally aligning with a digital-postage model in which senders bear or are charged for the cost, and large-scale spam becomes expensive. In OMR, by contrast, appending messages is essentially free, while recipients incur the retrieval cost of repeatedly scanning the bulletin board, even when no messages are intended for them. This asymmetry makes OMR vulnerable to sender-driven denial-of-service attacks.

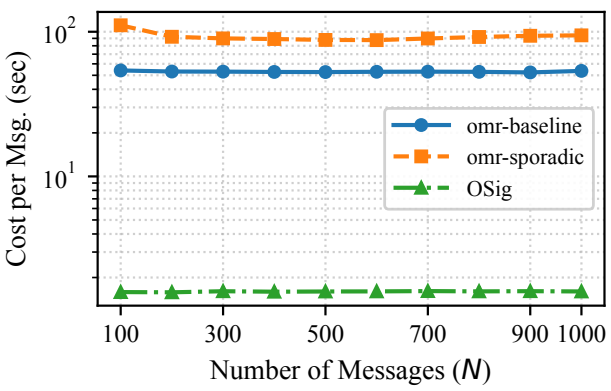

Figure 7: Average computational cost (in seconds) per successful message delivery (OSig) or recovered pertinent message (OMR) as a function of total workload N. The frequency of pertinent messages is fixed at $k = N/q$ across $q = 100$ recipients.

OMR also considers the *snake-eye attack*, where a single message appears pertinent to many recipients, forcing overprovisioning of $\bar{k}$ and amplifying server and recipient costs [36]. OSig can adopt the mitigation techniques from [34] by using snake-eye-resistant PKEs for pertinency tags. We describe a PKE-transciphering variant of OSig in Appendix C.2.

Scope and Limitations. Oblivious Signaling is designed to provide receiver privacy in a single-server asynchronous service, while leaving sender anonymity to external anonymous channels as in OMR. Accordingly, our evaluation focuses on the core cryptographic and architectural costs of this design point, rather than deployment mechanisms such as authenticated channels, PKI, network-level DoS defenses, timing side channels, storage systems, or fault tolerance. The threat model also excludes stronger environmental assumptions, including TEEs and two-server non-collusion. Privacy is defined against a malicious server that may collude with users, but correctness assumes protocol-following updates, so robustness against malicious server state corruption is outside our claims. Since recipients decrypt their inboxes locally and do not return decryption feedback to the server, post-read leakage and decryption-oracle behavior are also outside the protocol model.

The main operational trade-off is bounded storage. Each recipient has a fixed-capacity FIFO inbox that retains only the most recent ν pertinent messages, so if more than ν messages arrive before the recipient checks, older messages are evicted. This differs from OMR overflow, where underestimating $\bar{k}$ causes a failed retrieval and may force a retry that leaks information; in OSig, retrieval still succeeds, but it returns only the messages retained in the inbox. Deployments can address heavier users by assigning larger inboxes or by temporarily pausing delivery updates until the next retrieval. Our prototype evaluates *direct-inbox mode* with short messages, while larger messages can be represented by multiple ciphertext chunks or by *signal mode*, where the delivered object is an encrypted pointer to externally stored data. Finally, sharding improves throughput (Remark 6.2) but reduces the effective anonymity set from q to q/β.

Acknowledgments

We thank our shepherd and the anonymous USENIX Security reviewers for their insightful comments and constructive feedback, which significantly improved the final version of this paper. This research was supported by NSF #2143287.

Ethical Considerations

We study metadata-private message delivery. Our research did not involve human subjects, real users, or the collection of real-world communication data. All experiments use synthetic workloads, randomly generated ciphertexts, and controlled benchmarking infrastructure. We structure our ethical analysis around stakeholders, risks, mitigations, and justification.

Stakeholders. Relevant stakeholders include users of asynchronous messaging systems, messaging providers, researchers and implementers of privacy-preserving systems, operators responsible for abuse prevention, and society more broadly. Malicious actors are also relevant because privacy-enhancing technologies may be misused.

Benefits and risks. The primary benefit of OSig is defensive: it hides the message–recipient linkage by ensuring that the server performs a uniform homomorphic update across recipient inboxes. This protects sensitive communication relationships even when message contents are already encrypted. However, metadata privacy may also hinder spam or abuse detection. OSig has explicit limitations: sender anonymity is outside scope, bounded inboxes implement fixed-size retention, sharding reduces the effective anonymity set, and network-level metadata leakage remains possible.

Mitigations. We do not claim that OSig is deployment-ready or prevents abuse. Practical deployments would require operational safeguards such as reporting mechanisms, account controls, and abuse-triggered throttling. The protocol's delivery-time computation also creates a "digital postage" effect that increases the cost of spam or flooding attacks.

Justification. We believe publication is justified because OSig advances defensive privacy technologies while explicitly documenting its limitations and trade-offs to support responsible evaluation and future deployment.

Open Science

Our code for the implementation and experiments is available at the following public repository https://github.com/shuhanmirza/oblivious-signaling-usenix26. The repository contains build instructions, the code, and the bash scripts for running the experiments for OSig, `omr-baseline`, and `omr-sporadic`.

References

[1] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. Cryptology ePrint Archive, Paper 2015/046, 2015. URL: https://eprint.iacr.org/2015/046.

[2] Andreea Alexandru, Andrey Kim, and Yuriy Polyakov. General functional bootstrapping using CKKS. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology - CRYPTO 2025 - 45th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2025, Proceedings, Part III*, volume 16002 of *Lecture Notes in Computer Science*, pages 304–337. Springer, 2025. doi:10.1007/978-3-032-01881-6_10.

[3] Gabrielle Beck, Julia Len, Ian Miers, and Matthew Green. Fuzzy message detection. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021: 28th Conference on Computer and Communications Security*, pages 1507–1528, Virtual Event, Republic of Korea, November 15–19, 2021. ACM Press. doi:10.1145/3460120.3484545.

[4] Grant Bosworth, Keewoo Lee, and Sunwoong Kim. Leveraging fpgas for homomorphic matrix-vector multiplication in oblivious message retrieval, 2025. URL: https://doi.org/10.48550/arXiv.2512.11690, arXiv:2512.11690, doi:10.48550/ARXIV.2512.11690.

[5] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 868–886. Springer, 2012. doi:10.1007/978-3-642-32009-5_50.

[6] Anne Canteaut, Sergiu Carpov, Caroline Fontaine, Tancrède Lepoint, María Naya-Plasencia, Pascal Paillier, and Renaud Sirdey. Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. *J. Cryptol.*, 31(3):885–916, 2018. URL: https://doi.org/10.1007/s00145-017-9273-9, doi:10.1007/S00145-017-9273-9.

[7] David Cerdeira, Nuno Santos, Pedro Fonseca, and Sandro Pinto. SoK: Understanding the prevailing security vulnerabilities in TrustZone-assisted TEE systems. In *2020 IEEE Symposium on Security and Privacy*, pages 1416–1432, San Francisco, CA, USA, May 18–21, 2020. IEEE Computer Society Press. doi:10.1109/SP40000.2020.00061.

[8] Joachim Charzinski. Observations in e-mail performance. In *Proc. ITC Specialist Seminar*, pages 133–142, Würzburg, Germany, July 2002. Accessed 2026-01-23. URL: https://jcho.de/jc/Pubs/itc2002-col.pdf.

[9] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *J. Cryptol.*, 1(1):65–75, 1988. doi:10.1007/BF00206326.

[10] David Chaum, Debajyoti Das, Farid Javani, Aniket Kate, Anna Krasnova, Joeri de Ruiter, and Alan T. Sherman. cMix: Mixing with minimal real-time asymmetric cryptographic operations. In Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi, editors, *ACNS 2017: 15th International Conference on Applied Cryptography and Network Security*, volume 10355 of *Lecture Notes in Computer Science*, pages 557–578, Kanazawa, Japan, July 10–12, 2017. Springer, Cham, Switzerland. doi:10.1007/978-3-319-61204-1_28.

[11] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 409–437, Hong Kong, China, December 2017. Springer, Cham. doi:10.1007/978-3-319-70694-8_15.

[12] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, January 2020. doi:10.1007/s00145-019-09319-x.

[13] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th Annual Symposium on Foundations of Computer Science*, pages 41–50, Milwaukee, Wisconsin, October 23–25, 1995. IEEE Computer Society Press. doi:10.1109/SFCS.1995.492461.

[14] Sherman S. M. Chow, Katharina Fech, Russell W. F. Lai, and Giulio Malavolta. Multi-client oblivious RAM with poly-logarithmic communication. In Shiho Moriai

and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 160–190, Daejeon, South Korea, December 7–11, 2020. Springer, Cham, Switzerland. doi:10.1007/978-3-030-64834-3_6.

[15] Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. Riposte: An anonymous messaging system handling millions of users. In *2015 IEEE Symposium on Security and Privacy*, pages 321–338, San Jose, CA, USA, May 17–21, 2015. IEEE Computer Society Press. doi:10.1109/SP.2015.27.

[16] Henry Corrigan-Gibbs and Bryan Ford. Dissent: accountable anonymous group messaging. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 2010: 17th Conference on Computer and Communications Security*, pages 340–350, Chicago, Illinois, USA, October 4–8, 2010. ACM Press. doi:10.1145/1866307.1866346.

[17] Henry Corrigan-Gibbs, David Isaac Wolinsky, and Bryan Ford. Proactively accountable anonymous messaging in verdict. In Samuel T. King, editor, *USENIX Security 2013: 22nd USENIX Security Symposium*, pages 147–162, Washington, DC, USA, August 14–16, 2013. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity13/technical-sessions/presentation/corrigan-gibbs.

[18] Saba Eskandarian, Henry Corrigan-Gibbs, Matei Zaharia, and Dan Boneh. Express: Lowering the cost of metadata-hiding communication with cryptographic privacy. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021: 30th USENIX Security Symposium*, pages 1775–1792. USENIX Association, August 11–13, 2021. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/eskandarian.

[19] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. URL: https://eprint.iacr.org/2012/144.

[20] Shufan Fei, Zheng Yan, Wenxiu Ding, and Haomeng Xie. Security vulnerabilities of SGX and countermeasures: A survey. *ACM Comput. Surv.*, 54(6):126:1–126:36, 2022. doi:10.1145/3456631.

[21] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press. doi:10.1145/1536414.1536440.

[22] Jincheol Ha, Seongkwang Kim, Wonseok Choi, Jooyoung Lee, Dukjae Moon, Hyojin Yoon, and Jihoon Cho. Masta: An he-friendly cipher using modular arithmetic. *IEEE Access*, 8:194741–194751, 2020. doi:10.1109/ACCESS.2020.3033564.

[23] Sashidhar Jakkamsetti, Zeyu Liu, and Varun Madathil. Scalable private signaling. In *CSF 2025: IEEE 38th Computer Security Foundations Symposium*, pages 143–158, Santa Cruz, CA, USA, June 16–20, 2025. IEEE Computer Society Press. doi:10.1109/CSF64896.2025.00035.

[24] Yanxue Jia, Varun Madathil, and Aniket Kate. HomeRun: High-efficiency oblivious message retrieval, unrestricted. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024: 31st Conference on Computer and Communications Security*, pages 2012–2026, Salt Lake City, UT, USA, October 14–18, 2024. ACM Press. doi:10.1145/3658644.3670381.

[25] Marc Joye. TFHE public-key encryption revisited. In Elisabeth Oswald, editor, *Topics in Cryptology - CT-RSA 2024 - Cryptographers' Track at the RSA Conference 2024, San Francisco, CA, USA, May 6-9, 2024, Proceedings*, volume 14643 of *Lecture Notes in Computer Science*, pages 277–291. Springer, 2024. doi:10.1007/978-3-031-58868-6_11.

[26] Jaehyung Kim. Efficient homomorphic integer computer from ckks. Cryptology ePrint Archive, Paper 2025/066, 2025. URL: https://eprint.iacr.org/2025/066.

[27] Albert Kwon, Henry Corrigan-Gibbs, Srinivas Devadas, and Bryan Ford. Atom: Horizontally scaling strong anonymity. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*, pages 406–422. ACM, 2017. doi:10.1145/3132747.3132755.

[28] Albert Kwon, David Lazar, Srinivas Devadas, and Bryan Ford. Riffle: An efficient communication system with strong anonymity. *Proceedings on Privacy Enhancing Technologies*, 2016(2):115–134, April 2016. doi:10.1515/popets-2016-0008.

[29] Simon Langowski, Sacha Servan-Schreiber, and Srinivas Devadas. Trellis: Robust and scalable metadata-private anonymous broadcast. In *ISOC Network and Distributed System Security Symposium – NDSS 2023*, San Diego, CA, USA, February 27– March 3, 2023. The Internet Society. doi:10.14722/ndss.2023.23088.

[30] Kristin Lauter, Michael Naehrig, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? Cryptology ePrint Archive, Report 2011/405, 2011. URL: https://eprint.iacr.org/2011/405.

[31] Keewoo Lee and Yongdong Yeo. SophOMR: Improved oblivious message retrieval from SIMD-aware homomorphic compression. Cryptology ePrint Archive, Report 2024/1814, 2024. URL: https://eprint.iacr.org/2024/1814.

[32] Haofei Liang, Zeyu Liu, Eran Tromer, Xiang Xie, and Yu Yu. Implementation of InstantOMR. https://github.com/xiangxiecrypto/tfhe-omr, 2025. GitHub repository, accessed 2026-05-21.

[33] Haofei Liang, Zeyu Liu, Eran Tromer, Xiang Xie, and Yu Yu. InstantOMR: Oblivious message retrieval with low latency and optimal parallelizability. Cryptology ePrint Archive, Paper 2025/2317, 2025. Published elsewhere. Minor revision. USENIX Security 2026. URL: https://eprint.iacr.org/2025/2317.

[34] Zeyu Liu, Katerina Sotiraki, Eran Tromer, and Yunhao Wang. Snake-eye resistant PKE from LWE for oblivious message retrieval and robust encryption. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025, Part III*, volume 15603 of *Lecture Notes in Computer Science*, pages 126–156, Madrid, Spain, May 4–8, 2025. Springer, Cham, Switzerland. doi:10.1007/978-3-031-91131-6_5.

[35] Zeyu Liu and Eran Tromer. Oblivious message retrieval. Cryptology ePrint Archive, Report 2021/1256, 2021. URL: https://eprint.iacr.org/2021/1256.

[36] Zeyu Liu and Eran Tromer. Oblivious message retrieval. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part I*, volume 13507 of *Lecture Notes in Computer Science*, pages 753–783, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Cham, Switzerland. doi:10.1007/978-3-031-15802-5_26.

[37] Zeyu Liu, Eran Tromer, and Yunhao Wang. Group oblivious message retrieval. In *2024 IEEE Symposium on Security and Privacy*, pages 4367–4385, San Francisco, CA, USA, May 19–23, 2024. IEEE Computer Society Press. doi:10.1109/SP54263.2024.00115.

[38] Zeyu Liu, Eran Tromer, and Yunhao Wang. PerfOMR: Oblivious message retrieval with reduced communication and computation. In Davide Balzarotti and Wenyuan Xu, editors, *USENIX Security 2024: 33rd USENIX Security Symposium*, Philadelphia, PA, USA, August 14–16, 2024. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/liu-zeyu.

[39] Varun Madathil, Alessandra Scafuro, István András Seres, Omer Shlomovits, and Denis Varlakov. Private signaling. In Kevin R. B. Butler and Kurt Thomas, editors, *USENIX Security 2022: 31st USENIX Security Symposium*, pages 3309–3326, Boston, MA, USA, August 10–12, 2022. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity22/presentation/madathil.

[40] Pierrick Méaux, Claude Carlet, Anthony Journault, and François-Xavier Standaert. Improved filter permutators for efficient FHE: better instances and implementations. In Feng Hao, Sushmita Ruj, and Sourav Sen Gupta, editors, *Progress in Cryptology - INDOCRYPT 2019 - 20th International Conference on Cryptology in India, Hyderabad, India, December 15-18, 2019, Proceedings*, volume 11898 of *Lecture Notes in Computer Science*, pages 68–91. Springer, 2019. doi:10.1007/978-3-030-35423-7_4.

[41] Chao Niu, Benqiang Wei, Zhicong Huang, Zhaomin Yang, Cheng Hong, Meiqin Wang, and Tao Wei. SoK: FHE-friendly symmetric ciphers and transciphering. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2025(3):583–613, 2025. doi:10.46586/tches.v2025.i3.583-613.

[42] Ania M. Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. The loopix anonymity system. In Engin Kirda and Thomas Ristenpart, editors, *USENIX Security 2017: 26th USENIX Security Symposium*, pages 1199–1216, Vancouver, BC, Canada, August 16–18, 2017. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/piotrowska.

[43] Primus Labs. Primus-fhe: A fully homomorphic encryption library. GitHub repository. URL: https://github.com/primus-labs/primus-fhe.

[44] Ron Rothblum. Homomorphic encryption: From private-key to public-key. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 219–234, Providence, RI, USA, March 28–30, 2011. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-19571-6_14.

[45] István András Seres, Balázs Pejó, and Péter Burcsi. The effect of false positives: Why fuzzy message detection leads to fuzzy privacy guarantees? In Ittay Eyal and Juan A. Garay, editors, *FC 2022: 26th International Conference on Financial Cryptography and Data Security*, volume 13411 of *Lecture Notes in Computer Science*, pages 123–148, Grenada, May 2–6, 2022. Springer, Cham, Switzerland. doi:10.1007/978-3-031-18283-9_7.

[46] The Rust Team. The Rust programming language, 2024. https://www.rust-lang.org/.

[47] Adithya Vadapalli, Kyle Storrier, and Ryan Henry. Sabre: Sender-anonymous messaging with fast audits. In *2022 IEEE Symposium on Security and Privacy*, pages 1953–1970, San Francisco, CA, USA, May 22–26, 2022. IEEE Computer Society Press. doi:10.1109/SP46214.2022.9833601.

[48] Zama. TFHE-rs: A Pure Rust Implementation of the TFHE Scheme for Boolean and Integer Arithmetics Over Encrypted Data, 2022. https://github.com/zama-ai/tfhe-rs.

A Additional Remarks

Remark A.1. (Minimizing sender-side workload.) Our implementation leverages the fact that TFHE is a *hybrid* FHE scheme, combining LWE-type ciphertexts for fast homomorphic operations with RLWE/RGSW-style mechanisms that enable programmable bootstrapping and key-dependent transformations. To reduce sender overhead in our sender-batched setting, we instantiate public-key encryption using the public-key TFHE construction of Joye [25], which produces LWE-type ciphertexts under an RLWE-based security argument while yielding a shorter public key and less noisy ciphertexts than the direct "encryptions of zero" approach [44]. Concretely, this design reduces the size of the public key (published by the receivers) and the ciphertext (signal and pTag) transmitted by senders. The resulting system shifts part of the workload to the server: the server performs an *expansion* step to materialize the required internal forms used by homomorphic processing. This server-side preprocessing can be processed in parallel and amortized across batched requests, while significantly simplifying the sender's computation and bandwidth footprint.

Remark A.2. (Code organization.) Our Rust prototype mirrors the protocol roles, with Sender, Receiver, Server, and Postbox implemented as separate structs encapsulating state and cryptographic logic. signals, tags, and batched messages are wrapped in strongly typed abstractions to separate protocol flow from low-level FHE operations. Each encrypted value is a TFHE shortint ciphertext, and multi-word plaintexts are encoded as vectors of ciphertext "digits," with carry space reserved for subsequent homomorphic operations.

Because larger plaintext spaces significantly increase the cost of programmable bootstrapping and LUT-based operations, we represent multi-bit values using multiple small ciphertexts instead of a single wide one. Our parameters support two message bits per ciphertext, so an ℓ-bit tag is encoded as $\lceil \ell/2 \rceil$ ciphertext digits, reducing the cost of non-linear operations while preserving efficient composition. Dummy values use TFHE trivial ciphertexts, avoiding encryption overhead while remaining compatible with homomorphic processing.

We implement conditional selection arithmetically, expressing CMux via homomorphic MUL and ADD, which performs better in our workload by relying on cheaper gates. Independent ciphertext computations are parallelized using Rayon's work-stealing thread pool to efficiently utilize multiple cores for both per-message and batched processing.

Remark A.3. (Addressing Traffic-Volume Leakage in OMR.) For OMR designs, recall the problem of leaking information about message volume to the server, which we discussed in Section 2.1. Liu et al. [35] acknowledge these leakage vectors in detail and propose two mitigations: *Full-Key Unlinkability* and a *Two-Server Architecture*. OMR supports a recipient to generate fresh, unlinkable detection keys for every request [35, Section 9.3]. The authors suggest this prevents the server from linking a failed query (with $k > \bar{k}$) to a successful retry (with $k \leq \bar{k}'$). The authors also suggest separating the workload: one server performs a lightweight detection to return the exact count k, and a second server performs the retrieval using a noisy bound $\bar{k}' > k$ [35, Section 7.7]. However, these mitigations rely on the strong environmental assumption and are only effective to hide k if we assume a network-level unlinkability (even if the servers collude) between two requests from the same recipient.

Remark A.4. (PV in `omr-sporadic`.) In principle, the pertinency vector could be homomorphically aggregated into a single ciphertext to reduce bandwidth, at the cost of additional server-side FHE computation. Since our comparison runs all target systems on the same single-machine testbed and focuses on server processing time rather than network cost, we report results without this optimization, which would only marginally change the communication profile but not the dominant homomorphic workload.

B Additional Preliminaries

B.1 FHE Preliminaries

Definition B.1 (FHE Correctness)**.** An FHE scheme Π_{FHE} is correct if it satisfies the following properties for any security parameter $\lambda \in \mathbb{N}$ and any key pair $(pk, sk) \leftarrow \Pi_{FHE}.\mathsf{KGen}(1^\lambda)$:

1. *Decryption Correctness*: For any message $m \in \mathcal{M}$,

   $$\Pr[\Pi_{FHE}.\mathsf{Dec}(sk, \Pi_{FHE}.\mathsf{Enc}(pk, m)) = m] \geq 1 - \mathrm{negl}(\lambda).$$

2. *Evaluation Correctness*: For any circuit $C \in \mathbb{C}_n$ and any message vector $\mathbf{m} = (m_1, \ldots, m_n) \in \mathcal{M}^n$,

   $$\Pr[\Pi_{FHE}.\mathsf{Dec}(sk, c') = C(\mathbf{m})] \geq 1 - \mathrm{negl}(\lambda),$$

   where,

   $$c' \leftarrow \Pi_{FHE}.\mathsf{Eval}(pk, C, \mathbf{c}) \text{ , and}$$
   $$\mathbf{c} = (\Pi_{FHE}.\mathsf{Enc}(pk, m_1), \ldots, \Pi_{FHE}.\mathsf{Enc}(pk, m_n))$$

The probability is taken over the randomness of KGen, Enc, and Eval.

Well-formed Ciphertexts. For FHE schemes, the set of well-formed ciphertexts is formally defined recursively as:

$$\mathcal{C}_{pk} := \left\{ c \in \mathcal{C} \;\middle|\; \begin{array}{l} c \leftarrow \Pi_{FHE}.\mathsf{Enc}(pk, m) \text{ for any } m \in \mathcal{M} \\ \vee \\ c \leftarrow \Pi_{FHE}.\mathsf{Eval}(pk, C, (c_1, \ldots, c_n)) \\ \text{for any } n \in \mathbb{N}, C \in \mathbb{C}_n, \\ \text{ and } (c_1, \ldots, c_n) \in \mathcal{C}_{pk}^n \end{array} \right\}.$$

Strong Correctness Transformation Chow et al. [14] demonstrate that any FHE scheme Π_{FHE} that is (1) circular-secure and (2) has a well-defined decryption algorithm over $\mathcal{K} \times \mathcal{C}$ can be generically transformed into a strongly correct scheme Π'_{FHE}. This is achieved by defining an augmented evaluation algorithm that performs a bootstrapping step prior to evaluating the circuit C. The transformed scheme Π'_{FHE} retains the identical Enc and Dec algorithms of Π_{FHE}, but modifies the key generation and evaluation procedures as:

- $\Pi'_{FHE}.\mathsf{KGen}(1^\lambda)$: Execute the underlying key generation $(pk, sk) \leftarrow \Pi_{FHE}.\mathsf{KGen}(1^\lambda)$. Compute an encryption of the secret key $c_{sk} \leftarrow \Pi_{FHE}.\mathsf{Enc}(pk, sk)$. The new public key is $pk' = (pk, c_{sk})$ and the secret key remains $sk' = sk$.
- $\Pi'_{FHE}.\mathsf{Eval}(pk', C, (c_1, \ldots, c_n))$: The evaluation algorithm introduces a bootstrapping/rectification step:
  1. **Rectification:** For each input ciphertext c_i, homomorphically decrypt it using the encrypted secret key c_{sk}. This effectively computes:
  $$c'_i \leftarrow \Pi_{FHE}.\mathsf{Eval}(pk, \mathsf{Dec}(\cdot, c_i), c_{sk}).$$
  Here, $\mathsf{Dec}(\cdot, c_i)$ is a circuit being evaluated, with c_{sk} serving as the input.
  2. **Evaluation:** Evaluate the target circuit C on the resulting "refreshed" ciphertexts:
  $$c' \leftarrow \Pi_{FHE}.\mathsf{Eval}(pk, C, (c'_1, \ldots, c'_n)).$$
  3. Output the final ciphertext c'.

Justification. This construction ensures strong evaluation correctness because $\Pi_{FHE}.\mathsf{Dec}(\cdot, c_i)$ is well-defined over the entire space $\mathcal{K} \times \mathcal{C}$. Consequently, the rectification step (1) always yields a well-formed ciphertext $c'_i \in \mathcal{C}_{pk}$ that encrypts some message $m_i \in \mathcal{M}$, regardless of the validity of the original input c_i. The subsequent evaluation in step (2) therefore operates exclusively on valid ciphertexts, for which the standard correctness of Π_{FHE} guarantees the correct output.

Note that reliance on circular security is a standard assumption already required for FHE bootstrapping [21]. Furthermore, the requirement for a well-defined decryption algorithm is satisfied by most existing LWE-based constructions. In such schemes, decryption is typically computed as an inner product of vectors in $\mathbb{Z}_q^l$ followed by rounding, a mathematical operation that is inherently well-defined for any pair of vectors. Thus, by formally setting $\mathcal{C} = \mathcal{K} = \mathbb{Z}_q^l$, the decryption function is naturally well-defined over the full domain.

B.2 TFHE

We briefly recall the TFHE [12] scheme.

Notation. Let n denote the LWE dimension and N the ring dimension for RLWE ciphertexts. We write $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ for the real torus and $\mathbb{B} = \{0, 1\}$ for the binary set. Vectors are denoted in bold (e.g., $\mathbf{a}$), and $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ denotes the ring of integer polynomials.

TLWE Ciphertexts. A TFHE ciphertext is an LWE ciphertext over the torus, referred to as *TLWE*. Given a secret key $\mathbf{s} \in \{0, 1\}^n$, a TLWE encryption of a message $m \in \mathbb{T}$ is a vector

$$(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + m + e) \in \mathbb{T}^{n+1},$$

where $\mathbf{a} \leftarrow \mathbb{T}^n$ is uniform and e is sampled from a Gaussian error distribution. Decryption computes $b - \langle \mathbf{a}, \mathbf{s} \rangle$, which recovers m up to the small error term.

TRLWE Ciphertexts. TFHE uses Ring-LWE ciphertexts, denoted *TRLWE*, over the torus polynomial ring $\mathbb{T}_N[X] = \mathbb{R}[X]/(X^N + 1)$ modulo 1. In the canonical TFHE setting, the secret key satisfies $s(X) \in \mathbb{B}_N[X]$.[8] A TRLWE encryption of a message $\mu(X) \in \mathbb{T}_N[X]$ is a pair

$$(\mathbf{a}(X), b(X) = \mathbf{a}(X) \cdot s(X) + \mu(X) + e(X)) \in \mathbb{T}_N[X]^2,$$

where $\mathbf{a}(X)$ is uniform and $e(X)$ is a small Gaussian error polynomial.

Programmable Bootstrapping. The core primitive in TFHE is *programmable bootstrapping* (also known as gate bootstrapping), which allows for the evaluation of lookup tables (LUTs) while refreshing ciphertext noise. The bootstrapping key consists of *TGSW* ciphertexts, vectors of TRLWE samples that encrypt the secret key bits. To evaluate a function f, the scheme uses the *Blind Rotate* algorithm. This computes the external product between the TGSW bootstrapping keys and a TRLWE accumulator initialized with a "test vector" polynomial $v(X)$ encoding f. The rotation shifts $v(X)$ by the encrypted phase of the input, moving the coefficient corresponding to $f(m)$ to the constant term, which is then extracted as a fresh TLWE ciphertext.

C Variants

C.1 Transciphering

Transciphering, also known as Hybrid Homomorphic Encryption (HHE), is a pivotal technique designed to overcome the

[8] More generally, TFHE allows a vector secret in $\mathbb{B}_N[X]^k$; we describe the $k = 1$ case for simplicity.

communication and storage bottlenecks inherent in Fully Homomorphic Encryption (FHE) [30,41]. While FHE allows for computation on encrypted data, it suffers from a massive ciphertext expansion ratio—often requiring kilobytes to encrypt a single bit—which makes direct client-server communication prohibitively expensive for large datasets [41].

Mechanism and Workflow The primary goal of transciphering is to offload the heavy FHE encryption process from the resource-constrained client to the server. The protocol operates in three main steps [41]:

1. **Setup:** The client generates keys for both a symmetric cipher and an FHE scheme. The symmetric key k^{sym} is encrypted using the FHE public key pk, which produces $c_{k^{sym}}$ and is sent to the server.

2. **Encryption:** The client encrypts the data m using the efficient symmetric cipher Π_{sym}, producing a compact ciphertext c_{sym}. This ciphertext is sent to the server, incurring minimal bandwidth overhead (expansion ratio ≈ 1).

3. **Homomorphic Evaluation:** The server homomorphically evaluates the symmetric decryption circuit Dec_{sym} using the FHE-encrypted key $c_{k^{sym}}$ over c_{sym}:

$$c \leftarrow \Pi_{FHE}.\mathsf{Eval}(pk, \mathsf{Dec}_{sym}(\cdot, c_{sym}), c_{k^{sym}}) \qquad (1)$$

   The result c is a valid FHE ciphertext of m, ready for downstream homomorphic computations.

Security and Trade-offs Security in transciphering relies on the composition of the FHE scheme's semantic security and the symmetric cipher's robustness. While standard ciphers like AES are thoroughly battle-tested, FHE-friendly ciphers like Kreyvium [6], Masta [22], and FiLIP [40] are relatively recent. Therefore, selecting a cipher involves a trade-off between performance (favoring specialized FHE-friendly ciphers) and long-term security assurance (favoring AES).

C.2 Oblivious Signaling with Transciphering

We present a simple variant of Oblivious Signaling, denoted OSig2, which replaces sender-side FHE encryption with a public-key encryption (PKE) scheme and uses *transciphering* on the server to convert PKE ciphertexts into FHE ciphertexts. This variant is useful when (i) FHE encryption imposes high computation or bandwidth costs on senders, (ii) there exists an efficient homomorphic decryption circuit for the chosen PKE scheme, and (iii) the PKE instantiation provides snake-eye resistance, ensuring that a single ciphertext cannot be deemed pertinent to multiple recipients except with negligible probability (Recall the discussion on snake-eye attack in Section 7).

Algorithm 6 OSig2 Setup and Key Generation

procedure *ParamsGen*$(1^\lambda, \varepsilon_p, \varepsilon_c, \nu)$
  $\ell = \lceil \log(\varepsilon_p^{-1}) \rceil$
  $\bar{g} = \left\lfloor 1 + \frac{\varepsilon_c}{\varepsilon_p} \right\rfloor$
  $PP_{\mathsf{FHE}} \leftarrow \Pi_{FHE}.\mathsf{ParamsGen}(1^\lambda)$
  $PP_{\mathsf{PKE}} \leftarrow \Pi_{PKE}.\mathsf{ParamsGen}(1^\lambda)$
  return $PP = (1^\lambda, \ell, \bar{g}, \varepsilon_p, \varepsilon_c, \nu, PP_{\mathsf{FHE}}, PP_{\mathsf{PKE}})$

end procedure
procedure *KeyGen*( )
  $(\mathsf{pk}_{\mathsf{fhe}}, \mathsf{sk}_{\mathsf{fhe}}) \leftarrow \Pi_{FHE}.\mathsf{KGen}(1^\lambda)$
  $(\mathsf{pk}_{\mathsf{pke}}, \mathsf{sk}_{\mathsf{pke}}) \leftarrow \Pi_{PKE}.\mathsf{KGen}(1^\lambda)$
  $c_{sk} \leftarrow \Pi_{FHE}.\mathsf{Enc}(\mathsf{pk}_{\mathsf{fhe}}, \mathsf{sk}_{\mathsf{pke}})$ ▷ encrypt $\mathsf{sk}_{\mathsf{pke}}$
  return $pk = (\mathsf{pk}_{\mathsf{fhe}}, c_{sk}, \mathsf{pk}_{\mathsf{pke}}), \quad sk = (\mathsf{sk}_{\mathsf{fhe}}, \mathsf{sk}_{\mathsf{pke}})$
end procedure

Algorithm 7 OSig2 Message Generation

procedure *SignalGen*(PK, M)
  $g \leftarrow |PK|$ ▷ number of recipients in the batch
  for $i \leftarrow 1$ **to** g **do**
    $(pk_{fhe}^{(i)}, c_{sk}^{(i)}, pk_{\mathsf{pke}}^{(i)}) \leftarrow PK[i]$
    $msg_i \leftarrow M[i]$
    $\mathsf{pTag}_i \leftarrow \Pi_{PKE}.\mathsf{Enc}(pk_{\mathsf{pke}}^{(i)}, 1^\ell)$
    $\mathsf{signal}_i \leftarrow \Pi_{PKE}.\mathsf{Enc}(pk_{\mathsf{pke}}^{(i)}, msg_i)$
    $\mathsf{B}_{\mathsf{req}}[i] \leftarrow (\mathsf{signal}_i, \mathsf{pTag}_i)$
  end for
  return $\mathsf{B}_{\mathsf{req}} = ((\mathsf{signal}_1, \mathsf{pTag}_1), \ldots, (\mathsf{signal}_g, \mathsf{pTag}_g))$
end procedure

All procedures not listed below are identical to OSig.

Differences from OSig**.** The only changes introduced by OSig2 are: (i) senders encrypt signal and pTag using Π_{PKE} instead of Π_{FHE} (Algorithm 7); (ii) receivers publish an encryption of $\mathsf{sk}_{\mathsf{pke}}$ under Π_{FHE} to enable transciphering (Algorithm 6); and (iii) the server rectification step evaluates $\mathsf{Dec}_{\mathsf{pke}}$ homomorphically to convert PKE ciphertexts into FHE ciphertexts (Algorithm 8). All remaining steps are identical to OSig.

D Correctness Execution

Definitions 4.2 and 4.3 are evaluated over the following experiment. Let $PP \leftarrow ParamsGen(1^\lambda, \varepsilon_p, \varepsilon_c, \nu)$. For every registered receiver $r \in R$, generate $(pk_r, sk_r) \leftarrow KeyGen()$ and $(\mathsf{INBOX}_r^{(0)}, \mathsf{iTag}_r) \leftarrow InboxGen(pk_r)$. An honest delivery sequence is a sequence $\mathsf{D} = (D_1, \ldots, D_m)$, where $D_j = ((r_{j,1}, msg_{j,1}), \ldots, (r_{j,g_j}, msg_{j,g_j}))$, $g_j \le \bar{g}$, and each D_j contains no receiver twice. For each $j = 1, \ldots, m$, the sender

Algorithm 8 OSig2 Server Processing (Transciphering)

procedure *AddSigToInbox*$(pk, \mathsf{INBOX}, \mathsf{iTag}, \mathsf{B_{req}})$
  $(\mathsf{pk_{fhe}}, c_{sk}) \leftarrow pk$
  $(\mathsf{c}_1, \ldots, \mathsf{c}_\nu) \leftarrow \mathsf{INBOX}$
  $\mathsf{sigAcc} \leftarrow \Pi_{FHE}.\mathsf{Enc}(\mathsf{pk_{fhe}}, 0)$
  $\mathsf{PF} \leftarrow \Pi_{FHE}.\mathsf{Enc}(\mathsf{pk_{fhe}}, 0)$
  for $i \leftarrow 1$ **to** $|\mathsf{B_{req}}|$ **do**
    $(\mathsf{signal}_i, \mathsf{pTag}_i) \leftarrow \mathsf{B_{req}}[i]$
  ▷ *Step 1: Transciphering*
    $\mathsf{signal}'_i \leftarrow \Pi_{FHE}.\mathsf{Eval}(\mathsf{pk_{fhe}}, \mathsf{Dec_{pke}}(\cdot, \mathsf{signal}_i), c_{sk})$
    $\mathsf{pTag}'_i \leftarrow \Pi_{FHE}.\mathsf{Eval}(\mathsf{pk_{fhe}}, \mathsf{Dec_{pke}}(\cdot, \mathsf{pTag}_i), c_{sk})$
  ▷ *Step 2: Pertinency check*
    $\mathsf{PI}_i \leftarrow \Pi_{FHE}.\mathsf{Eval}(\mathsf{pk_{fhe}}, \mathsf{EQUAL}, (\mathsf{iTag}, \mathsf{pTag}'_i))$
  ▷ *Step 3: Filter and accumulate*
    $\mathsf{sigAcc} \leftarrow \mathsf{CMux}(\mathsf{PI}_i, \mathsf{sigAcc}, \mathsf{signal}'_i)$
    $\mathsf{PF} \leftarrow \Pi_{FHE}.\mathsf{Eval}(\mathsf{pk_{fhe}}, \mathsf{OR}, (\mathsf{PF}, \mathsf{PI}_i))$
  end for
  ▷ *Step 4: Inbox update*
  $\mathsf{c}'_1 \leftarrow \mathsf{CMux}(\mathsf{PF}, \mathsf{c}_1, \mathsf{sigAcc})$
  for $j \leftarrow 2$ **to** ν **do**
    $\mathsf{c}'_j \leftarrow \mathsf{CMux}(\mathsf{PF}, \mathsf{c}_j, \mathsf{c}_{j-1})$
  end for
  return $\mathsf{INBOX}' = (\mathsf{c}'_1, \ldots, \mathsf{c}'_\nu)$
end procedure

computes

$$\mathsf{B_{req}}_j \leftarrow \mathit{SignalGen}\big((pk_{r_{j,1}}, \ldots, pk_{r_{j,g_j}}), (msg_{j,1}, \ldots, msg_{j,g_j})\big).$$

This delivery sequence induces the request-batch sequence $\mathsf{Q}_{recv} = (\mathsf{B_{req}}_1, \ldots, \mathsf{B_{req}}_m)$. For each $j = 1, \ldots, m$ and every registered receiver r, the server updates the receiver state as:

$$\mathsf{INBOX}_r^{(j)} \leftarrow \mathit{AddSigToInbox}\big(pk_r, \mathsf{INBOX}_r^{(j-1)}, \mathsf{iTag}_r, \mathsf{B_{req}}_j\big)$$

Receiver r reads by computing $MV_r \leftarrow \mathit{ReadInbox}(sk_r, \mathsf{INBOX}_r^{(m)})$. A delivery (r, msg) is live after Q_{recv} if it is among the latest ν deliveries to r in D.

E Security Proofs

Theorem 5.1 (Completeness)**.** *The* OSig *protocol satisfies Completeness according to Definition 4.2, assuming* Strong Correctness *and* Wrong-Key Decryption *of the underlying FHE scheme* Π_{FHE} *and that the batch size satisfies* $|\mathsf{B_{req}}| \le \bar{g}$.

Proof. Fix any execution sequence $\mathsf{Q}_{recv} = (\mathsf{B}^1_{\mathsf{req}}, \ldots, \mathsf{B}^m_{\mathsf{req}})$, any receiver $r \in R$, any index j_0, and any message $m_{sent} \in \mathcal{M}$ sent to r in batch $\mathsf{B}^{j_0}_{\mathsf{req}}$, as quantified in Definition 4.2. Assume moreover that m_{sent} is among the most recent ν deliveries to r within Q_{recv}. Let $T \subseteq \{j_0+1, \ldots, m\}$ denote the indices of batches in which r is not targeted, and let $g_0 \triangleq |\mathsf{B}^{j_0}_{\mathsf{req}}|$.

Let Succ denote the event that, after processing Q_{recv}, the message m_{sent} appears in r's final decrypted inbox output MV. If m_{sent} is correctly inserted into r's INBOX during processing of $\mathsf{B}^{j_0}_{\mathsf{req}}$ and no subsequent batch in T triggers an inbox update for r, then m_{sent} appears in MV. Indeed, the assumption that m_{sent} is among the most recent ν deliveries to r implies that there are at most $\nu - 1$ later deliveries to r, so these later deliveries cannot evict m_{sent} from the ν-slot inbox. We upper bound $\Pr[\neg\mathrm{Succ}]$ by considering the following bad events.

1. **FHE evaluation error.** The homomorphic evaluation (including evaluation of the decryption circuit) may produce an incorrect ciphertext. By the strong correctness of Π_{FHE}, this event occurs with probability at most $\mathrm{negl}(\lambda)$.

2. **Pertinency collision:** During processing of the targeted batch $\mathsf{B}^{j_0}_{\mathsf{req}}$ against receiver r, more than one batch entry is deemed pertinent, i.e., at least one entry not intended for r yields $\mathsf{PI}_i = [\![1]\!]_{sk}$ in the equality check between pTag and iTag.

   In this case, the accumulator sigAcc may be overwritten by an entry not intended for r, causing insertion of an incorrect plaintext.

3. **Post-delivery false update:** For some $j \in T$, the batch $\mathsf{B}^{j}_{\mathsf{req}}$ (which does not target r) triggers an inbox update for r (i.e., the pertinency flag PF decrypts to 1), which may evict m_{sent} from the fixed-size inbox.

We check pertinency via an equality circuit EQUAL evaluated on pTag'_i and iTag. The resulting ciphertext PI_i decrypts to 1 iff the two underlying plaintext tags are equal, and to 0 otherwise. Now consider any entry $(\mathsf{signal}_i, \mathsf{pTag}_i)$ in a batch processed against receiver r that is not intended for r. Since this pTag was encrypted under a different receiver's key, Definition 3.5 implies that after the server's rectification step, the decrypted tag equals 1^ℓ with probability at most $1/|\mathcal{M}| + \mathrm{negl}(\lambda)$. Since we treat tags as ℓ-bit strings, we have $|\mathcal{M}| \ge 2^\ell$, and therefore $1/|\mathcal{M}| \le 2^{-\ell}$. By parameter selection, $2^{-\ell} \le \varepsilon_p$, so each entry not intended for r yields a false positive with probability at most $\varepsilon_p + \mathrm{negl}(\lambda)$.

By a union bound over the $g_0 - 1$ entries in the targeted batch $\mathsf{B}^{j_0}_{\mathsf{req}}$ that are not intended for r, the probability of a pertinency collision is at most $(g_0 - 1)\varepsilon_p + \mathrm{negl}(\lambda)$. Since *ParamsGen* sets $\bar{g} = \lfloor 1 + \varepsilon_c/\varepsilon_p \rfloor$ and we enforce $|\mathsf{B}^{j_0}_{\mathsf{req}}| \le \bar{g}$, we have $(g_0 - 1)\varepsilon_p \le \varepsilon_c$, and therefore the collision probability is at most $\varepsilon_c + \mathrm{negl}(\lambda)$.

For any $j \in T$, the receiver r is not targeted by $\mathsf{B}^{j}_{\mathsf{req}}$, so every entry in the batch is not intended for r. By a union bound over the $|\mathsf{B}^{j}_{\mathsf{req}}|$ entries, the probability that PF decrypts to 1 and triggers an update is at most $|\mathsf{B}^{j}_{\mathsf{req}}|\varepsilon_p + \mathrm{negl}(\lambda)$. Applying a union bound over $j \in T$ bounds the probability of a post-delivery false update by $\sum_{j \in T} |\mathsf{B}^{j}_{\mathsf{req}}|\varepsilon_p + \mathrm{negl}(\lambda)$.

Combining the bad events, we obtain

$$\Pr[\mathrm{Succ}] \;\ge\; (1 - \varepsilon_c) - \sum_{j \in T} |\mathsf{B}^{j}_{\mathsf{req}}|\varepsilon_p - \mathrm{negl}(\lambda),$$

where the sum term bounds the probability of a post-delivery false update over the non-targeted batches in T. Since the execution is polynomially bounded in λ and $\varepsilon_p(\lambda)$ is negligible (as required by Definition 4.2), the quantity $\sum_{j\in T}|\mathsf{B}^{\mathsf{j}}_{\mathsf{req}}|\,\varepsilon_p$ is negligible in λ and can be absorbed into $\mathrm{negl}(\lambda)$. Therefore, $\Pr[\mathsf{Succ}] \geq (1-\varepsilon_c) - \mathrm{negl}(\lambda)$, proving completeness. □

Theorem 5.2 (Soundness)**.** *The* OSig *protocol satisfies Soundness according to Definition 4.3, assuming* Strong Correctness *and* Wrong-Key Decryption *of* Π_{FHE}.

Proof. Fix any setup as in Definition 4.3, and fix any index j such that the receiver r under consideration is not targeted by the batch $\mathsf{B}^{\mathsf{j}}_{\mathsf{req}}$. Let $g \triangleq |\mathsf{B}^{\mathsf{j}}_{\mathsf{req}}|$.

In Algorithm 4, the inbox update is controlled by the pertinency flag PF. If PF decrypts to 0, then each CMux gate decrypts to the prior slot value, and therefore the inbox state is unchanged and $MV^{(j)} = MV^{(j-1)}$, except with negligible probability due to FHE evaluation error under Strong Correctness. Thus, the event $MV^{(j)} \neq MV^{(j-1)}$ can occur only if PF decrypts to 1, up to negligible probability.

Since receiver r is not targeted by $\mathsf{B}^{\mathsf{j}}_{\mathsf{req}}$, every batch entry is not intended for r. For each entry, the pertinency indicator PI_i is computed by an equality check between iTag (which encrypts 1^ℓ) and a rectified tag pTag'_i obtained by bootstrapped decryption using the encrypted secret key c_{sk}. By Definition 3.5, the wrong-key decryption output equals 1^ℓ with probability at most $1/|\mathcal{M}| + \mathrm{negl}(\lambda)$. Since tags are treated as ℓ-bit strings, $|\mathcal{M}| \geq 2^\ell$, and therefore this probability is at most $2^{-\ell} + \mathrm{negl}(\lambda) \leq \varepsilon_p + \mathrm{negl}(\lambda)$. By a union bound over the g entries, $\Pr[\text{PF decrypts to } 1] \leq g\,\varepsilon_p + \mathrm{negl}(\lambda)$.

Combining the above observations (and accounting for the negligible probability of FHE evaluation error under Strong Correctness), we obtain

$$\Pr\left[MV^{(j)} \neq MV^{(j-1)}\right] \leq g\,\varepsilon_p + \mathrm{negl}(\lambda),$$

where $g = |\mathsf{B}^{\mathsf{j}}_{\mathsf{req}}|$. Since Definition 4.3 (via Definition 4.2) assumes $\varepsilon_p(\lambda)$ is negligible and that each batch is polynomially bounded in λ, the product $g\,\varepsilon_p$ is negligible and can be absorbed into $\mathrm{negl}(\lambda)$. Therefore, $\Pr[MV^{(j)} \neq MV^{(j-1)}] \leq \mathrm{negl}(\lambda)$, as required. □

Theorem 5.3 (Receiver-Message Unlinkability)**.** *The* OSig *protocol satisfies Receiver-Message Unlinkability (RM$\bar{L}$) according to Definition 4.4, assuming the* Circular Security *and* IK-IND-CPA Security *of* Π_{FHE}.

Proof. Fix any PPT adversary $\mathcal{A}$ in the RM$\bar{L}$ game of Definition 4.4. We compare the two experiments corresponding to $b = 0$ and $b = 1$.

In the setup phase, the challenger samples PP, generates (pk_0, sk_0) and (pk_1, sk_1), and initializes inbox states $(\mathsf{INBOX}_0, \mathsf{iTag}_0)$ and $(\mathsf{INBOX}_1, \mathsf{iTag}_1)$. These values are independent of b and are identically distributed in the two experiments. The only value that depends on b is the single-entry challenge batch $\mathsf{B}_{\mathsf{req}} = ((\mathsf{signal}^\star, \mathsf{pTag}^\star))$.

Let $(\mathsf{pk}^{(0)}_{\mathsf{fhe}}, c^{(0)}_{sk}) \leftarrow pk_0$ and $(\mathsf{pk}^{(1)}_{\mathsf{fhe}}, c^{(1)}_{sk}) \leftarrow pk_1$. Note that the public keys in our protocol include the encrypted secret key components $c^{(0)}_{sk}$ and $c^{(1)}_{sk}$ used for bootstrapping. Accordingly, the indistinguishability claim below relies on IK-IND-CPA security (Definition 3.2), stated in the presence of these encrypted secret keys, and on Circular Security. The challenger forms

$$(\mathsf{signal}^\star, \mathsf{pTag}^\star) = \big(\Pi_{FHE}.\mathsf{Enc}(\mathsf{pk}^{(b)}_{\mathsf{fhe}}, msg),\ \Pi_{FHE}.\mathsf{Enc}(\mathsf{pk}^{(b)}_{\mathsf{fhe}}, 1^\ell)\big),$$

for the message msg chosen by $\mathcal{A}$. We claim that the distribution of $\mathsf{B}_{\mathsf{req}}$ is computationally indistinguishable for $b = 0$ and $b = 1$.

Consider the following hybrids for the challenge batch:

$$H_0 \triangleq \big(\Pi_{FHE}.\mathsf{Enc}(\mathsf{pk}^{(0)}_{\mathsf{fhe}}, msg),\ \Pi_{FHE}.\mathsf{Enc}(\mathsf{pk}^{(0)}_{\mathsf{fhe}}, 1^\ell)\big),$$
$$H_1 \triangleq \big(\Pi_{FHE}.\mathsf{Enc}(\mathsf{pk}^{(1)}_{\mathsf{fhe}}, msg),\ \Pi_{FHE}.\mathsf{Enc}(\mathsf{pk}^{(0)}_{\mathsf{fhe}}, 1^\ell)\big),$$
$$H_2 \triangleq \big(\Pi_{FHE}.\mathsf{Enc}(\mathsf{pk}^{(1)}_{\mathsf{fhe}}, msg),\ \Pi_{FHE}.\mathsf{Enc}(\mathsf{pk}^{(1)}_{\mathsf{fhe}}, 1^\ell)\big).$$

By IK-IND-CPA security (Definition 3.2), H_0 and H_1 are computationally indistinguishable (set both challenge messages to msg so that the only hidden bit is which key is used). Similarly, H_1 and H_2 are computationally indistinguishable (set both challenge messages to 1^ℓ). Therefore H_0 and H_2 are computationally indistinguishable, and hence the challenge batch distribution is indistinguishable for $b = 0$ versus $b = 1$.

In the observation phase, $\mathcal{A}$ computes the updated inboxes by running *AddSigToInbox* on $(pk_0, \mathsf{INBOX}_0, \mathsf{iTag}_0, \mathsf{B}_{\mathsf{req}})$ and $(pk_1, \mathsf{INBOX}_1, \mathsf{iTag}_1, \mathsf{B}_{\mathsf{req}})$. This is a fixed PPT computation on public inputs, so computational indistinguishability of $\mathsf{B}_{\mathsf{req}}$ is preserved. Thus $\mathcal{A}$'s entire view in the two experiments is computationally indistinguishable, and $\mathcal{A}$ has negligible advantage, i.e., $\left|\Pr[b' = b] - \frac{1}{2}\right| \leq \mathrm{negl}(\lambda)$. □