Verbeth: Secure Messaging with Metadata Minimization over Public Blockchain Logs

Marco Esposito, Andrea Rizzini, Francesco Bruschi, and Donatella Sciuto

***Abstract*—This work presents a private instant messaging protocol that leverages the public log layer of blockchains as the message transport layer, while the cryptographic state is kept only by client applications. Thanks to the properties of public ledgers, this approach achieves strong censorship resistance, while also revealing the economic and cryptographic limits of on-chain messaging. Notably, given the transparency of public ledgers, and since reading and writing operations are in most cases outsourced to third-party providers that may be curious, a well-known concern is direct metadata leakage. We address this both at first contact and during the conversation: for first contact, we propose two alternative discovery mechanisms, one based on long-term key encapsulation with trial decryption, the other on a private signaling service backed by trusted hardware. For the ongoing conversation, we show that topic rotation, driven by the off-chain cryptographic state, suffices to prevent topic and conversation linkability. As our main contribution, we provide an in-depth analysis of Verbeth's metadata leakage under different adversarial assumptions for both phases.**

***Index Terms*—blockchain, messaging, double-ratchet, end-to-end-encryption, metadata privacy.**

I. Introduction

SECURE software applications often rely on the dubious premise that operators will consistently preserve both service availability and user privacy. Recent client-side scanning mandates[1], the withdrawal of privacy features in certain jurisdictions[2], and the cost of maintaining a trustworthy centralized backend[3] show how easily either guarantee can disappear. Applications built on decentralized and open protocols can reduce this dependence on individual operators, thereby making it more difficult for any single party to withdraw access or functionality. This strength, however, comes with the difficulty that data processed or published through a shared public infrastructure may be required to stay private. Protocols have therefore begun to incorporate privacy-enhancing mechanisms that address this challenge without giving up the benefits of permissionlessness and decentralization [1].

In this paper, we question the status quo that private information, e.g. conversations and digital correspondence, should not be posted on a blockchain. This direction is not entirely without precedent, for example through broader European work on MPC-based threshold cryptographic primitives that has explored how distributed ledgers can act as verifiable infrastructure for privacy-preserving cryptographic protocols[4].

Here, we suggest that messaging systems can inherit properties from public blockchains, such as *global availability* and *censorship resistance*, that neither centralized nor current peer-to-peer and federated designs can provide to the same extent. To demonstrate these features and their tradeoffs, we present *Verbeth*, whose early proof-of-concept version appeared in a workshop [2], a secure messaging protocol that keeps its messaging logic and all cryptographic state and operations on the clients, outside Ethereum's execution layer, and uses only event logs for contact discovery and message delivery. We examine how this minimal design exploits simple protocols and cryptographic primitives, the costs it incurs under realistic usage patterns, and the security and privacy guarantees it ultimately provides.

A. Motivation and contributions

In general, once a transaction is finalized on a distributed ledger, its content availability no longer depends on the continued operation of any particular relay, indexer, or application backend. We say that publication of such content enjoys ledger-level survivability which depends on no third-party service. This implies that a ciphertext can continue to exist on the ledger while becoming undecryptable if a client endpoint loses local state. Moreover, there is a second consequence of ledger finality that appears at odds with the historical treatment of secure messengers: protocols evolved from the OTR lineage are in fact deliberately repudiable, so that a transcript shown to a third party proves nothing about who wrote what [3], also commonly listed as a desirable property [4]. However, for disputes between counterparties communications subject to audit that property is no longer desirable. In such settings, the sender may later need to prove that something was said, by whom and when [5], [6]. Today that need is met by certified e-mail schemes resting on accredited providers, or by subpoenaing the business records of a platform. Conversely, a protocol whose transport is an authenticated public ledger can offer an analogous publication record without any custodian in certain scenarios [7], [8], [9]. While decentralized non-repudiation work has focused on data exchange, certified delivery and usage logging, here we focus on ongoing private conversations. Our contributions, which extend those already in [2], are:

This work is supported by MUSA – Multilayered Urban Sustainability Action – project, funded by the European Union – NextGenerationEU, under the National Recovery and Resilience Plan (NRRP) Mission 4 Component 2 Investment Line 1.5: Strengthening of research structures and creation of R&D "innovation ecosystems", set up of "territorial leaders in R&D"

[1]https://www.europarl.europa.eu/doceo/document/E-10-2025-003249_EN.html.

[2]https://www.theguardian.com/technology/2025/feb/21/apple-removes-advanced-data-protection-tool-uk-government.

[3]https://www.wired.com/story/signal-operating-costs/.

[4]https://cordis.europa.eu/project/id/780477

- We define a blockchain native setting for secure messaging in which EVM logs are used as the transport layer. Prior ledger-based channels show that blockchains can carry hidden data, but they do not provide end-to-end encrypted messaging with non-repudiation and explicit metadata goals.
- Over that transport, we illustrate a post-quantum hybrid key exchange protocol that lets two users establish a session asynchronously (i.e. without relying on pre-keys) where long-term keys are restricted to identity binding and discovery. The response to a handshake request is built so that its public tag supports stealth discovery, while the ratchet key that seeds the first post-handshake messages remains inside the encrypted response.
- We formalize query leakage for outsourced log retrieval. In practice, many clients do not read the chain through their own full node, but through an RPC endpoint that may learn sensitive information. We analyze both query leakage under ordinary filtered retrieval and under a bulk retrieval variant.
- We avoid stable conversation handles, so messages are emitted with ratcheted EVM topics derived from the Double Ratchet key schedule. Among other things, we show that unlinkability of these topics inherits the forward secrecy and post compromise recovery behavior of the ratchet.
- We instantiate and evaluate the protocol with respect to the client side cryptographic workflow and the cost of the main operations. We also compare the two constructions to achieve first contact in terms of on-chain gas, local scanning cost, bandwidth and private signaling overhead.

II. Related work

Decentralized and Wallet-based messengers: A first line of systems binds messaging identities to blockchain accounts while deferring delivery, to varying degrees, to dedicated networks. In XMTP, a user proves control of an address through a wallet signature and then interacts with a network of dedicated nodes that deliver and store messages [10]. DirectCasts, the client-side messaging feature implemented in Farcaster's apps like Warpcast, follows a similar pattern over Farcaster hubs, adding a hierarchy of signed keys and a more complex ratchet structure [11], [12]. Waku (as used by Status) and the Session network instead route messages over peer-to-peer overlays that provide decentralized routing and some metadata protection [13], [14]. Interestingly, Session is the only system here that does rely on a blockchain, but only to maintain a Sybil resistant consensus over the service node network, without a role in transport. Across all of these designs the ledger anchors identity, incentives or membership, whereas message availability still depends on dedicated node networks whose operators observe delivery and retrieval metadata and remain exposed to the jurisdictional pressures recalled in the introduction. Asynchronous first-contact, in particular, inherits the X3DH and PQXDH pattern of pre-key bundles served by that same infrastructure [15], meaning that contacting a stranger requires a key server to be reachable and honest about the bundles it serves.

The ledger as a communication channel: A complementary line of work uses the ledger itself to carry data. Works by Recabarren and Carbunar embed data in Bitcoin transaction scripts to build a censorship-resistant channel [16], [17]. Steganographic approaches hide ciphertexts in transaction fields for censorship-resistant rendezvous [18], while others embed messages in the signature nonces of ordinary transactions [19]. These works establish the property that our transport inherits, namely that ledger publication survives adversaries able to block or coerce dedicated infrastructure. The adversary in these works, however, targets publication so the goal is to hide the existence of a channel from a censor inspecting transactions. Also, keys are assumed to be pre-shared out of band, receivers detect messages by scanning the entire chain from a node they run themselves, and the channel carries independent datagrams with no session state.

Metadata-private messaging systems: Several protocols address metadata leakage in private messaging over the Internet. Vuvuzela [20] routes messages through a fixed chain of mix servers that inject cover traffic, under the assumption that at least one server remains uncompromised, while Pung [21] relies on private information retrieval, allowing clients to fetch messages from an untrusted key-value store without revealing which entries they access. Later systems improve scalability by distributing the workload across many servers [22], [23]. These designs provide strong guarantees even against global traffic analysis, but they anchor availability to server organizations that can be shut down or legally coerced.

Private message detection and retrieval: When a message is posted on a public bulletin board, how can the receiver identify that a specific message is addressed to them? In the literature we find many works that propose a solution to this problem. Fuzzy Message Detection [24] lets the receiver delegate detection to an untrusted server, which can flag candidate messages without learning precisely which ones are addressed to them; then we find many works based on Oblivious Message Retrieval [25], [26], [27], where servers detect and retrieve messages obliviously, without learning any information about which message is addressed to which receiver. A third approach replaces cryptographic obliviousness with hardware assumptions, leveraging TEEs [28], [29]. Crucially, these schemes are not instantiated against the query interface of a concrete ledger, whose indexed filters are themselves observable by whoever serves them, and they leave open how a detected message bootstraps an authenticated session and how retrieval remains private for the conversation that follows.

In summary, prior work covers several aspect although in isolation. No existing system combines ledger transport with serverless asynchronous first-contact, where discovery exposes no recipient selector to either the public or the serving infrastructure, and post-quantum session establishment.

III. Preliminaries

A. End-to-end encryption

End-to-end encryption encrypts messages on the sender's device so that only the intended recipient can decrypt them, keeping plaintext inaccessible to intermediate servers. The symmetric key is established through asymmetric cryptography, either by a key agreement protocol such as Diffie-Hellman

(DH) [30] or a Key Encapsulation Mechanism (KEM). A single static exchange, however, lets the compromise of a long-term key expose past or future communications.

a) Forward Secrecy and Post-Compromise Security: FS guarantees that compromising a long-term key does not expose past communications, since session keys are ephemeral and discarded after use; PCS guarantees that communications recover their security after a compromise of the current state, provided fresh key material is later introduced. A fresh DH exchange per message would achieve both, but is impractical when the recipient is offline. Formally, let K_i be the key of the i-th message: a deterministic derivation $K_{i+1} = \mathsf{KDF}(K_i)$ gives FS but not PCS, since an attacker holding K_i obtains every K_j with $j > i$. PCS therefore requires an ephemeral DH contribution dh_i, unpredictable to that attacker, yielding $K_{i+1} = \mathsf{KDF}(K_i, dh_i)$.

b) Double Ratchet: The Double Ratchet is a cryptographic construction for end-to-end encrypted communication that guarantees both FS and PCS. The term *ratchet* refers to the one-way nature of the key derivation process: given the current cryptographic state, it is computationally infeasible to recover any previous state, just as a mechanical ratchet can only turn in one direction. In the original construction proposed by Signal, the two ratchets are referred to as the Symmetric-Key Ratchet and the Diffie-Hellman Ratchet [31]. Its operation is fully described by four algorithms whose pseudo-code we provide in the supplementary material.

c) Authenticated Encryption with Associated Data: AEAD is the standard approach to symmetric encryption, guaranteeing confidentiality and integrity under a single key so that tampering with the ciphertext is detectable. It additionally authenticates *associated data* without encrypting it, which suits the Double Ratchet, where headers must stay readable yet tamper-evident.

d) Non-repudiation and deniability: The choice between MAC-based and signature-based authentication has consequences beyond integrity. A MAC computed under a shared symmetric key, as in any AEAD scheme, authenticates a message to the peer and to nobody else such that a transcript shown to a third party proves nothing about authorship. With the OTR protocol, this repudiability is a clear goal, hence leaving no cryptographic evidence behind [3], and later such property is inherited, in various forms, by Signal and most of its successors [4]. Instead, with a digital signature a signer cannot deny a message that verifies under their public key. The OSI security architecture defines it as *non-repudiation*, to prevent an entity from denying having participated in all or part of a communication[5]. Classical treatments distinguish non-repudiation of origin, of submission, of delivery and of receipt, and obtain their evidence through an inline trusted third party [32], [6]. We return to these notions in Section V-C, where the ledger itself takes over the role of that third party.

B. Post-quantum encryption

Shor [33] and Grover [34] algorithms represent a concrete threat once quantum computers become available. NIST initiated a competition in 2016 that eventually led to the standardization of ML-KEM as a post-quantum primitive [35].

a) Key privacy: For some applications, the public KEM ciphertext must not reveal which public key was used for encapsulation. We use the standard key-privacy, or recipient-anonymity, notion for public-key encryption [36], which we state here in the analogous KEM form.

Definition 1 (KEM key privacy)**.** *A Key encapsulation method* $\mathsf{KEM} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ *is key-private if for every PPT adversary* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$*, the following experiment outputs* 1 *with probability at most* $1/2 + \mathsf{negl}(\lambda)$*. Sample* $(pk_0, sk_0), (pk_1, sk_1) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and* $b \leftarrow \{0,1\}$*. Let* $s \leftarrow \mathcal{A}_0(pk_0, pk_1)$*, compute* $(ct^\star, K^\star) \leftarrow \mathsf{Encaps}(pk_b)$*, and let* $b' \leftarrow \mathcal{A}_1(ct^\star, s)$*. The experiment outputs* 1 *iff* $b' = b$*.*

b) Hybrid key-exchange method: An hybrid key exchange method is built by combining a post-quantum secure KEM (e.g., ML-KEM) and a classical key exchange (e.g., ECDH), deriving a final shared secret by feeding both outputs into a Key Derivation Function (KDF). This provides security guarantees as long as at least one of the two schemes remains unbroken, protecting against both classical and quantum adversaries. Formally, such a protocol $HybKE = (Init, Respond, Finish)$ is constructed from a classical Diffie-Hellman scheme $DH_{cls} = (KeyGen, ComputeKey)$ and a post-quantum KEM $KEM_{pqc} = (KeyGen, Encaps, Decaps)$. The algorithmic description is available in the supplementary material.

C. Private signaling

Private signaling is a cryptographic technique where a sender posts data on a public bulletin board along with a signal that only the intended receiver can interpret, allowing them to locate and retrieve the data without revealing their identity to the server. This idea was introduced by Madathil et. al [28] proposing two constructions: i) leveraging two non-colluding servers; ii) leveraging a Trusted Execution Environment (TEE). The TEE construction has been later on refined by Jakkamsetti et. al [29] integrating an Oblivious RAM TEE to avoid side channel attacks. For the aim of this paper, we are mostly interested to the TEE-based construction. Refer to the supplementary materials for a primer on TEEs and private signaling with TEEs.

IV. System model

We consider a client-side secure messaging protocol deployed over an append-only ledger $\mathcal{L}$. In this model, the on-chain component is a smart contract $\mathcal{C}$ whose role is limited to event publication, while all cryptographic operations are performed locally by client endpoints. The effective transport medium is not contract state, but the sequence of logs emitted by $\mathcal{C}$ that are embedded in transaction receipts and authenticated by the per-block receipt trie . Each user is treated as an externally owned account (EOA), and we use subscripts s and r to denote sender and recipient accounts. Table I summarizes the notation used next.

[5]https://www.iso.org/standard/76153.html

TABLE I: Core protocol notation

Symbol / Acronym	Meaning
s, r	sender and recipient
a_u	transport account controlled by user u
$\mathcal{L}$	ledger (append-only authenticated log)
$\mathcal{C}$	Verbeth transport contract
ρ_r	recipient reachability selector for r
π_u	identity-binding proof
τ	hybrid response tag carried in $\mathsf{HSR.inResponseTo}$
$(pk^{\mathsf{DH}}, sk^{\mathsf{DH}})$	elliptic-curve Diffie-Hellman keypair
$(pk^{\mathsf{SIG}}, sk^{\mathsf{SIG}})$	signature keypair used for message authentication
$(pk^{\mathsf{KEM}}, sk^{\mathsf{KEM}})$	ML-KEM key-pair
ct	encrypted blob payload published on chain
θ_i	ratcheted message topic at epoch i
$P^{\mathsf{w}}, P^{\mathsf{r}}$	write-side and read-side providers

A user u controls an EOA a_u and local client state. This state includes long-term key-exchange and signature keys, ephemeral handshake material, and ratchet state for ongoing conversations. At a high level, the protocol consists of three publication steps: (i) A sender s initiates first contact by emitting a handshake event HS, (ii) the recipient r answers with a handshake response event HSR, (iii) once a shared session is established, both parties exchange post-handshake messages through MSG events addressed to ratcheted topics.

a) Identity, session establishment and channel evolution: We separate transport identity, messaging identity and session state. While transport identity is simply a_u authorizing publication on $\mathcal{L}$, messaging identity is defined by long-term public keys that are bound to the a_u through an identity proof π_u. This binding allows the counterparty to verify that the messaging keys presented during session establishment are endorsed by the same principal that controls a_u. Reachability at first contact is handled through a recipient selector ρ_r included in HS. In a basic instantiation, ρ_r is derived from the recipient address and acts as a discovery handle for inbound handshake retrieval. After the session is established, reachability no longer depends on address-derived selectors and instead moves to ratcheted message topics known only to the two endpoints. Session establishment is based on a hybrid key exchange that combines an ephemeral EC-DH component with an ML-KEM component. From a resulting hybrid shared secret σ_{hyb}, the parties derive an initial root key RK_0 and bootstrap the session state used by the double ratchet. The responder publishes an HSR event containing a response tag τ and an encrypted payload ct that carries the material needed by the initiator to complete session setup. Subsequent message delivery uses ratcheted topics θ_i, where each epoch is derived from fresh state and produces a new on-chain delivery handle. At the system level, the important point is that first contact and post-handshake reachability are handled by different objects: the former is tied to ρ_r (at least in the baseline setting), while the latter is tied to evolving session state.

b) Client access model: As in most blockchain-based applications, a client submits EOA-authorized transactions to $\mathcal{L}$ through a provider P^{w}. These writes include transport publications and, in protocol variants that use auxiliary contracts, any additional registration call. Moreover, the client retrieves logs either through its own full node or through a read-side provider P^{r}. For later use, we denote the public ledger view by $\mathsf{View}_{\mathsf{pub}} = (e_1, e_2, \ldots, e_n)$, where each e_j is an event emitted by $\mathcal{C}$ and recorded on $\mathcal{L}$. For a read-side provider P^{r}, we denote the observation associated with a client u by $\mathsf{View}^u_{P^{\mathsf{r}}} = (q_1, q_2, \ldots, q_m, \nu_u)$, where each q_i denotes a log-retrieval query issued by the client and ν_u denotes auxiliary network-side metadata available to the provider, such as timing and endpoint-level identifiers.

Notably, read queries towards events and logs are privacy-sensitive, since they reveal the indexed values that the client is attempting to match, such as ρ_r, τ, or a set of active topics $\{\theta_i\}$. Later we show that in the basic setting, privacy-sensitive observations arise from the joint view $\mathsf{View}_{\mathsf{pub}} \cup \mathsf{View}^u_{P^{\mathsf{r}}}$. To minimize cross-channel correlation, a client may use a different provider for transaction submission and for read-side log queries, as it is often the case in modern dApp architectures (e.g., the use of third-party indexers for reads vs. private relays for writes). Our results do not rely on this separation for protocol correctness. However, this separation matters with respect to the amount of metadata that a single operator can correlate across the write and read paths.

c) Remark on ledger finality: Cryptographic protocols whose participants act on unconfirmed on-chain messages face security and delay issues, where an adversary can exploit forks to observe a counterparty's move on one branch and adapt its own on another [37]. In this protocol, no step discloses a value whose early observation gives a cross-branch advantage, and client state transitions are safe to reorg because state advances only locally, at encryption time or upon successful authenticated decryption, and never as a consequence of ledger position (i.e. a dropped transaction is equivalent to a lost message, and a re-included event fails authentication). Only the evidence claims of Definition 2 bind to receipt-trie inclusion, and must therefore be evaluated at finality.

V. Problem statement

The security problem studied in this paper is not limited to message confidentiality and integrity. On a public ledger, ciphertext transport is usually inseparable from metadata exposure, exacerbated by the fact that many clients outsource log retrieval to infrastructure that observes their filters. Our objectives then fall under both *content security* and *metadata minimization*. While the former concerns, among other well-known guarantees, preventing harvested ciphertexts from becoming useful plaintext later, and is thus relevant to much of the latest literature on efficient hybrid post-quantum messaging systems, the latter goal comprises solutions to prevent linkability tests that determine whether a specific party a_u is in communication with another one.

When observing $\mathsf{View}_{\mathsf{pub}}$, one necessarily reveals the a_u that emits an event, the event type, block position, all transaction metadata, gas behavior, indexed event fields and approximate payload sizes. Practically, if a_s posts HS or MSG, an observer sees only that a_s acted, whereas P^{w} may additionally observe transaction-submission metadata. On the reading side, the provider P^{r} sees the user log-retrieval $\mathsf{View}^u_{P^{\mathsf{r}}}$, including queried block ranges, event signatures, indexed-topic filters, request timing, and endpoint metadata such as IP address or

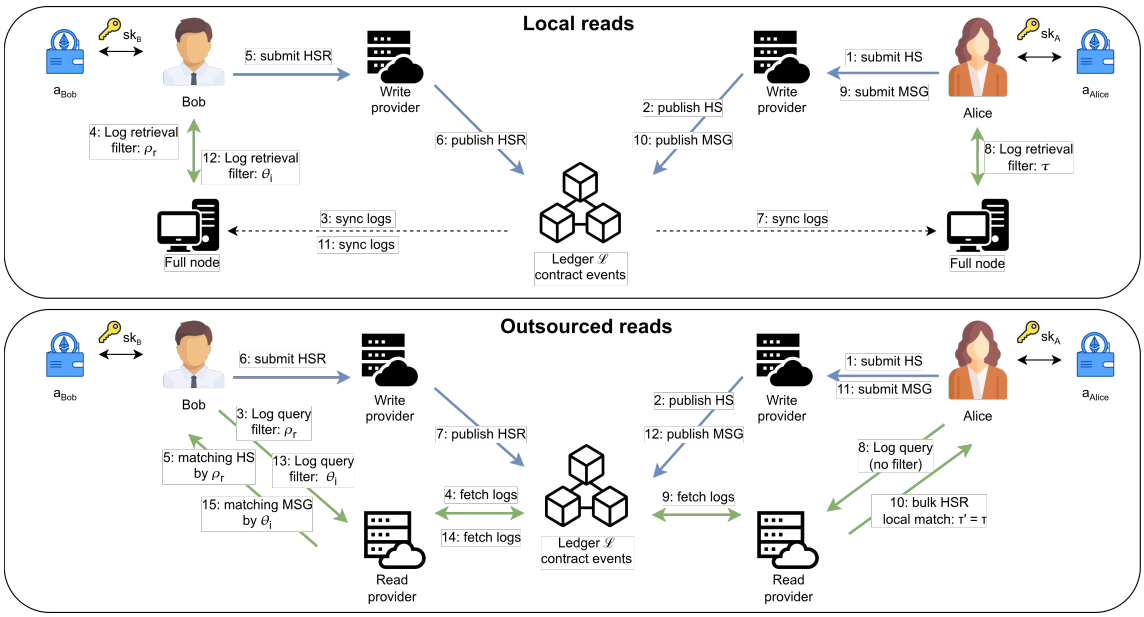

Fig. 1: Read and write paths for two client configurations: clients relying on local full nodes for reads (top image), and clients using third-party read provider (bottom image). Blue arrows indicate write-path steps, green arrows indicate read-path steps.

TLS identifiers. Since the separation of these two providers is a key assumption for our work, we avoid further abstractions such as those of smart contract accounts standards where the presence of bundlers, paymasters, and wallet modules would complicate the analysis and make it more difficult to isolate, operationally, the read and write providers.

a) Direct leakage versus heuristic leakage: We distinguish direct metadata attacks from heuristic or statistical attacks. This separation is close in spirit to models for preserving privacy within blockchain transaction structures, where an adversary observes the transaction structure and attempts to infer hidden relations between objects on the ledger [38]. In that setting, unlinkability concerns the adversary's inability to decide whether two states, and therefore the transactions involving them, are related. For example, let L denote a ground-truth relation over the observables recorded on $\mathcal{L}$, for instance the one grouping observables that belong to the same conversation pair, and let $\widetilde{L}$ denote the relation inferred by an adversary from $\mathsf{View}_{\mathsf{pub}}$ and from $\mathsf{View}^{u}_{P^{\mathsf{r}}}$. A direct metadata attack gives the adversary an efficient predicate that collapses part of this uncertainty, e.g. "this handshake targets B", "this query is looking for B" or "this topic belongs to the same conversation as that handshake". Heuristic attacks instead produce $\widetilde{L}$ from imperfect evidence such as timing, traffic sparsity, transaction graph structure or network observations [39], [40], [41]. Our goal is therefore distinct from that broader literature, as we mostly explore how to reduce direct metadata attacks within the context of transaction log events.

b) Basic linkability inference attack: To illustrate why recipient privacy and query privacy cannot be analyzed independently, we take the case of a basic mechanism for achieving first contact without prior coordination between two accounts. Suppose an initiator a_s sends from their client $\mathsf{HS} = (a_s, \rho_r, \ldots)$, where ρ_r is deterministically obtained as $Hash(a_r)$. Since the candidate address space relevant to an application is often enumerable, an observer can test guesses for a_r offline by recomputing $Hash(a_r)$. If the recipient's client B uses outsourced retrieval to find HS, the read-side provider P^{r} sees a query of the form

$$\nu_B \longrightarrow \mathsf{getLogs}(\mathsf{HS}, \rho_r, \Delta),$$

where ν_B denotes endpoint metadata of B and Δ is the queried block range. P^{r} can therefore associate the endpoint with the recipient selector, and, after offline guessing, with a_r itself. Later, when the same endpoint queries for a ratcheted topic θ and the public log contains $\mathsf{MSG} = (a_s, \theta, \ldots)$, the provider can correlate

$$\nu_B \longrightarrow \theta \quad \text{and} \quad (A, \theta, \ldots) \in \mathsf{View}_{\mathsf{pub}}$$

to recover the conversational relation $a_s \leftrightarrow a_r$ (i.e. that two public addresses are exchanging messages). Intuitively, a way to prevent the above attack is to ensure that neither the public event nor the retrieval query exposes a deterministic recipient selector, e.g. $\rho_r = H(a_r)$.

The above observation is not new in the literature and is closely related to the private signaling problem introduced in section III-C. Here, we specialize this as a contact discovery problem to EVM log transport and propose two different solutions. Furthermore, once a session has been established, we set a different goal from hiding first contact. In fact, even if an adversary learns that a_s exchanged keys - i.e. attempted to contact a_r, they should not obtain a direct test linking HS to HSR, HSR to a later MSG or one ratcheted topic θ_i to the next. Finally, the goal of content security under long-term observation, i.e. resistance to harvest-now-decrypt-later attacks, is orthogonal to the above metadata problems but also necessary for the same deployment model. As introduced in section III-B, hybrid key exchange is the

direction taken by recent secure-messaging work [15], [12], and allows us to derive session keys from single use material. In our setting, because first contact happens asynchronously without a pre-key server, we require a small adaptation so that only ephemeral DH secrets and the KEM shared secret contribute to the initial session key. Thus, we show that harvested HS, HSR, and MSG payloads remain undecryptable even if long-term identity keys are later compromised, within the protocol's FS and PCS assumptions.

To place the two models that follow, recall the adversaries usually considered in secure messaging: a passive network observer, who sees encrypted traffic, and an untrusted server, who stores every envelope and learns which client fetches which mailbox. Our below models involve what an adversary gets once the server's database is public, and how security changes when retrieval is mediated by an untrusted party.

A. Threat Model A

In this threat model (TM-A), honest users retrieve logs from their own full nodes. The adversary, denoted by $\mathcal{A}_{\mathsf{pub}}$, is an external observer of the public ledger and its view is limited to $\mathsf{View}_{\mathsf{pub}}$. Not only they observe every event emitted by $\mathcal{C}$ and the account a_u that publishes each transaction, but can also control arbitrary accounts that submit any transaction accepted by $\mathcal{C}$. In contrast, the adversary does not see client log filters, queried block ranges, endpoint metadata ν_u, local payload decryption outcomes or the local state used to test ρ_r, τ or θ_i. The essence of such model is to ask if the public objects in $\mathsf{View}_{\mathsf{pub}}$ give an efficient direct test for hidden relations such as the intended recipient of HS or the continuity relation between HS, HSR and later MSG events.

B. Threat Model B: outsourced retrieval

TM-B extends TM-A with an adversarial read provider. Honest clients may retrieve logs through P^{r}, such as a RPC endpoint, relay or indexer server. Notably, this adversary $\mathcal{A}_{\mathsf{read}}$ observes both $\mathsf{View}_{\mathsf{pub}}$ and $\mathsf{View}^{u}_{P^{\mathsf{r}}}$ for clients that use the adversarial provider. Event signatures requested by the client, indexed topic filters, queried block ranges, endpoint data and metadata ν_u can all be seen by the P^{r}. Thus, the adversary can also correlate these observations with $\mathsf{View}_{\mathsf{pub}}$. The important additional leakage in this model is query leakage, because even if a value is not linkable from $\mathsf{View}_{\mathsf{pub}}$ alone, it may become linkable if it appears as a filter in some query $q_i \in \mathsf{View}^{u}_{P^{\mathsf{r}}}$. In particular, a query for a recipient selector, or for a value that can be tested offline against likely recipients, can bind endpoint data ν_u to the hidden retrieval target.

a) Remark on non-colluding providers: The metadata claims in this paper rely on provider separation, as P^{w} and P^{r} are assumed to be independent and non-colluding (i.e. they do not share information outside what is already committed to the ledger, hence obtaining a larger joint view), even though correctness of the messaging protocol does not require this assumption.

b) Remark on integrity and availability: A malicious P^{r} may also omit or delay, or otherwise cause denial of service. Still, these attacks do not allow the provider to make an endpoint accept a forged protocol message, assuming the endpoint always checks ciphertext authentication against long-term signing keys and ledger inclusion whenever such inclusion is relied upon. In deployments where clients do not run a full node, this verification can be supplied by a light node which locally verifies consensus and execution against untrusted RPC endpoints [42].

1) TM-B variant with bulk retrieval.: We also consider a restricted variant of TM-B, denoted by $\mathsf{TM\text{-}B}_{\mathsf{bulk}}$, in which clients still rely on an adversarial read provider P^{r}, but do not send privacy-sensitive indexed filters. By contrast, the client requests all relevant events and performs matching locally at the cost of heavier processing and bandwidth usage. We write such a query as $q_i = \mathsf{getLogs}(\mathcal{C}, \mathsf{type}, \Delta)$, where type may identify a public event class such as HS, HSR, or MSG, but q_i does not filter for ρ_r, τ, θ_i, or any value derived from a hidden recipient or session state. The adversary in $\mathsf{TM\text{-}B}_{\mathsf{bulk}}$ therefore still observes $\mathsf{View}_{\mathsf{pub}}$ and the metadata in $\mathsf{View}^{u}_{P^{\mathsf{r}}}$, including ν_u, but no longer receives information that the endpoint is trying to match. Thus, the particular attack step in paragraph V-0b where $\nu_B \longrightarrow \mathsf{getLogs}(\mathsf{HS}, \rho_r, \Delta)$ binds ν_B to ρ_r is removed. However, this does not remove all direct pair leakage. The provider still observes endpoint metadata ν_u which may itself identify the client or restrict it to a small candidate set. For instance, let $\Gamma(\nu_u)$ denote the set of accounts that P^{r} can associate with ν_u. This set may be large, but in the most conservative case it may be a singleton (e.g. when ν_u includes stronger identifiers). If the public event still exposes a deterministic recipient selector, such as $\rho_r = H(a_r)$, then P^{r} can combine endpoint attribution with $\mathsf{View}_{\mathsf{pub}}$. Therefore, for a queried range Δ, the provider can compute: $\mathsf{Cand}(\nu_u, \Delta) = \{(a_s, a_r) : \mathsf{HS} = (a_s, H(a_r), \ldots) \in \mathsf{View}_{\mathsf{pub}}[\Delta] \ \wedge \ a_r \in \Gamma(\nu_u)\}$.

When this set is small, the adversary obtains fewer possible conversation pairs, but when it is a singleton, it directly recovers $a_s \leftrightarrow a_r$. Thus, the privacy advantage of bulk retrieval depends on the residual uncertainty in $\Gamma(\nu_u)$ and on the density of matching public events in the queried range.

C. Security and Privacy goals

a) Security goals: The blockchain makes all published ciphertexts permanently and publicly accessible to anyone, hence exposing all cryptographic material to Harvest Now, Decrypt Later (HNDL) attacks across the three protocol steps. Our content security goals are such that: FS must hold against a passive quantum attacker, i.e. one that harvests the ledger but never compromises a client; while PCS against a classical adversary that additionally obtains a transient snapshot of a client's ratchet state. We do not claim PCS against a quantum adversary holding such a snapshot, since the DH ratchet updates are classical, and closing this gap requires post-quantum ratchet updates [12]. Section VII argues both goals per event type.

b) Forensic goals: Where most secure messengers preserve the deniability recalled in paragraph III-A0d, we deliberately target the opposite property, since part of the value of a public ledger transport is that messages can serve as evidence. Thus, we set out our goals based on the following definition.

Definition 2 (Ledger-anchored non-repudiation)**.** (i) Non-repudiation of publication. *For every event $e \in \mathsf{View_{pub}}$, any party can verify from $\mathcal{L}$ alone that account a_u authorized the publication of e at block b, the evidence being the transaction signature together with the receipt-trie inclusion proof.* (ii) Conditional non-repudiation of origin. *For any successfully decrypted message m, the receiving party can produce a tuple (plaintext, decryption key material, inner signature, identity proof π_s, inclusion proof) convincing any verifier that the principal controlling a_s authored m no later than block b, while no party lacking the disclosed material can run the same test.* (iii) Conditional non-repudiation of receipt. *For a later post-handshake message m' sent by the recipient of an earlier ratchet epoch, if the signed plaintext of m' binds an acknowledged prefix of earlier ciphertexts, a disclosing party can produce the tuple of part (ii), augmented with the ratchet witness and that prefix, convincing any verifier that the sender of m' both possessed the state needed to derive the later topic under Equation 2 and acknowledged receipt of the prefix.*

Specifically for part (ii), every payload must carry an inner signature under sk_s^{SIG}, bound to a_s by π_s and verifiable only after decryption. We note ledger availability alone does not show that a recipient read a message, only that the message was available on the agreed channel, which in the classical typology amounts to evidence of origin and delivery to the channel, but not of receipt [32]. Nevertheless, with a prefix a later signed payload can indeed evidence receipt of the acknowledged messages.

c) Privacy goals: To meet our goals on metadata leakage, we use a single notion of unlinkability between items of interest. The notion borrows from a consolidated terminology [43], in which two or more items of interest are unlinkable if the adversary cannot sufficiently distinguish whether they are related, and from its transaction-level instantiation over ledger objects [38]. In particular, we restrict the adversary's distinguishing capability to efficient direct tests.

Definition 3 (Direct unlinkability)**.** *Let V be the view available to the adversary in a given threat model (under* TM-A*, $\mathsf{View_{pub}}$; under* TM-B *and its bulk variant, $\mathsf{View_{pub}} \cup \mathsf{View}^{u}_{P^{\mathsf{r}}}$), and let L be a hidden relation over items of interest, such as published events, accounts, and retrieval queries. The protocol provides direct unlinkability for L with respect to V if no efficient predicate over the values exposed in V, including equality tests and recomputation of public deterministic functions, decides whether a tuple of items belongs to L.*

Table II instantiates the definition 3 on our four privacy goals, each row naming the hidden relation through its forbidden direct test and the threat models, hence the views V, under which it must be prevented.

TABLE II: Privacy goals as instances of Definition 3

Privacy goal	**Forbidden direct test**	**Relevant threat model**
Recipient first contact	Given HS and a candidate a_r, decide whether HS targets r.	TM-A, TM-B, TM-$\mathsf{B_{bulk}}$
Filtered query	Given q_i and endpoint metadata ν_u, decide that the endpoint is searching for r or for a specific τ or θ_i.	TM-B
Conversation continuity	Given public τ or θ_i, link HS to HSR, HSR to MSG, or MSG_i to MSG_{i+1}.	TM-A, TM-B, TM-$\mathsf{B_{bulk}}$
Conversation pair	Link a_s and a_r by combining first-contact leakage with filtered retrieval or endpoint attribution.	TM-B; also TM-$\mathsf{B_{bulk}}$when ν_u identifies or narrows the recipient set

VI. Proposed Solution

We now present two variants of first contact for key exchange that remove the need of any public handle in the handshake and, after this step, share the same procedure for session establishment and the same use of ratcheted message topics for post-handshake delivery. Algorithmic descriptions of the two constructions' steps and the post-handshake steps are provided in the supplemental material.

A. Construction 1: bulk first-contact retrieval.

In this first construction, we encrypt the entire material of HS under a long-term key controlled by the recipient who later discovers their own inbound handshakes by trial decryption of the relevant log range. Notably, paying a linear cost only on HS (and, symmetrically, on HSR later) is acceptable, as we show in section VIII, unlike a full scan of all events for settings under TM-$\mathsf{B_{bulk}}$. For this we introduce a minimal pre-registration contract $\mathcal{C}_{\mathsf{reg}}$ separate from $\mathcal{C}$, which maintains a mapping $a_u \mapsto pk_u^{\mathsf{KEM\text{-}lt}}$, i.e. the account that controls a long-term ML-KEM public key authorizing the registration via a transaction over $\mathcal{L}$. To prevent rogue key registration of arbitrary public keys under one's own account, $\mathcal{C}_{\mathsf{reg}}$ must accept registration when gated by a proof of possession ϕ_u of the corresponding secret. Importantly, the stored post-quantum encapsulation material, whose rotation mechanism we consider out of scope, only protects the handshake envelope, not the eventual session. Specifically, the envelope wrapping uses ML-KEM alone, while the keys established inside the envelope are the ephemeral hybrid material that initiates the session.

To send a handshake, the initiator s resolves the recipient's long-term key previously registered to $\mathcal{C}_{\mathsf{reg}}$, generates the session material, and, by encapsulating against the registered long-term key, derives an ephemeral envelope key K_{env}. This is used exactly once to encrypt the body carrying $\mathsf{msg_{eph}}$, the signature σ_s, and the identity-binding proof π_s.
Then the recipient r retrieves all HS events over the queried range Δ through the reading provider P^{r} and attempts to decrypt each one under $sk_r^{\mathsf{KEM\text{-}lt}}$. The result is that AEAD verification implicitly filters events not addressed to r, after which σ_s and π_s are checked to authenticate the sender.
With this procedure, we observe that:

- An honest recipient reconstructs associated data locally from their own address and the public sender field of the event, and AEAD verification implicitly enforces that the envelope was constructed for them by the claimed sender.
- The inner signature σ_s is verified after envelope decryption against π_s, this prevents a public observer from validating π_s off chain even when one a_s is suspected.

Overall, the on-chain registration action publicly attributes "willingness to receive" to a_u which is strictly weaker leak than recipient identification per handshake.

B. Construction 2: private-signaling handshake discovery.

Another way to avoid the usage of ρ_r entirely is by leveraging the Scalable Private Signaling construction [29]. At a high level, the initiator posts a handshake message HS omitting ρ_r, and separately submits a private signal bound to that transaction to the SPS infrastructure. The intended recipient obtains the ledger position through SPS and then queries the ledger for the corresponding event. For our construction, the only needed adaptation is how the SPS recipient identifier is chosen. In the single-server presentation of SPS, the TEE may assign an opaque incremental RID during registration [29, Sec. 6], but in our setting publishing a separate mapping (a_r, RID_r) would reintroduce a discovery dependency and an additional public object for senders to resolve. Since the sender already knows the recipient's address a_r, we instead set $\mathsf{RID} = a_r$. If an implementation requires fixed-length ORAM identifiers, a_r denotes its canonical fixed-length encoding. During registration the client includes its address in the encrypted registration payload:

$$R_i\colon ct_{\mathsf{key}} \leftarrow \mathsf{Enc}(epk_{\mathsf{TEE}},\ pk_i \| a_i),$$

and the TEE assigns $\mathsf{RID} \leftarrow a_i$. As a result, any sender who knows a_r can directly use it as the SPS recipient identifier, with no additional lookup required. The handshake initiator can now construct and publish the handshake message $\mathsf{HS}' = \mathsf{HS} \setminus \{\rho_r\}$ while submitting an SPS signal for the same transaction. In the SPS abstraction the signal location is a bulletin-board position loc but in our instantiation this is the transaction hash returned after publishing HS'. The sender therefore waits for the transaction receipt, sets $pos = tx_hash$ and submits the private signal to the SPS service. In the recipient-side discovery, the recipient invokes $\mathsf{SPS.Receive}$ to obtain a vector of pending ledger positions. Values that are not valid positions are treated as SPS padding, as specified by the underlying protocol [29, Sec. 6]. For each valid pos, the recipient can query any RPC provider for the corresponding transaction receipt and extract the HS' event data.

C. Stealth response and topic-ratcheted delivery

The two constructions differ only in how $\mathsf{msg}_{\mathsf{eph}}$ reaches the responder, while everything from the response onward is shared and completes the protocol. Two mechanisms remain to be specified: the responder must answer without publishing any value testable against candidate conversation pairs, and post-handshake delivery must move to handles that admit no continuity test across events (see Table II). Both reuse the primitives of Section III-B and the claims they support are argued in Section VII. Throughout, $\mathsf{HKDF}(\mathsf{ikm}, \mathsf{salt}, \mathsf{info})$ fixes the argument convention and Kec denotes Keccak-256, whose only role is mapping derived keys onto the `bytes32` indexed-topic encoding of EVM logs.

a) Stealth response to contact request.: Upon recovering $\mathsf{msg}_{\mathsf{eph}} = (pk_s^{\mathsf{cls}}, pk_s^{\mathsf{kem}})$ from an inbound handshake, the responder r executes $\mathsf{HybKE.Respond}$, relabeling its output K as σ_{hyb} and retaining the encapsulated share ss^{kem}, and additionally samples a one-time *tag keypair* (sk^{tag}, R) whose sole purpose is deriving the response tag

$$\tau \ = \ \mathsf{Kec}\big(\mathsf{HKDF}(\ ss^{\mathsf{kem}},\ \mathsf{DH}(sk^{\mathsf{tag}}, pk_s^{\mathsf{cls}}),\ \texttt{hsr}\ )\big).$$

The response message $\mathsf{msg}_B = (pk_r^{\mathsf{cls}},\ ct^{\mathsf{kem}})$, the responder's signing key and identity proof π_r travel encrypted under the initiator's ephemeral key, so that the public event carries only τ, a_r and R .

The separation between the tag keypair and the key-exchange material is what removes any public bridge out of HSR because R is consumed by tag verification and never reused, while pk_r^{cls}, which seeds the first DH ratchet epoch and therefore reappears as a MSG header, never occurs in a public field. Notably, the derivation inverts the roles that a classical stealth construction would assign: dual-key stealth addresses derive their detection tag from an ECDH share between a published ephemeral and a scanning key, a pattern recently ported to EVM environments [44], [45]. Since that share would be quantum-recoverable from the published values, we use it only as HKDF salt and let the encapsulated ss^{kem} provide the input key material. Furthermore, while in payment-oriented schemes the recipient scans all outputs, here the burden falls on the initiator, who retrieves HSR events and matches each against their pending handshakes . Since τ depends on the responder's fresh sk^{tag} and on ss^{kem}, the initiator cannot predict it, so no filtered query is possible even in principle. Thus, the cost is $O(P)$ in the initiator's own pending handshakes (see also Table III), not in global traffic.

b) Session bootstrap: Both parties now hold $(\sigma_{\mathsf{hyb}}, \tau)$. The initial root key and the responder's sending chain are derived as $RK_0, CK \leftarrow \mathsf{KDF}(0^{32}, \sigma_{\mathsf{hyb}})$, after which the Double Ratchet of Section III-A0a proceeds unchanged, the initiator performing the first DH step against pk_r^{cls} on its first send. Epoch-0 topics are derived per direction as

$$\theta_0^{\mathsf{dir}} \ = \ \mathsf{Kec}\big(\mathsf{HKDF}(\sigma_{\mathsf{hyb}},\ \tau,\ \text{topic-dir})\big), \mathsf{dir} \in \{\mathsf{out}, \mathsf{in}\} \quad (1)$$

with labels assigned from the initiator's perspective. We note that no long-term key contributes to σ_{hyb}, unlike in PQXDH where asynchronous first contact is achieved by mixing DH terms against the recipient's long-term and signed pre-keys uploaded to a server [15]. We instead let HS itself carry one-shot pre-keys, restricting long-term keys to identity binding and, in Construction 1, envelope wrapping.

c) Topic ratcheting: After the handshake, reachability still needs an efficient retrieval handle, and a static one would reintroduce the continuity predicate that ρ_r posed for first contact. We therefore make the delivery handle an output of the key schedule itself, namely each DH ratchet step derives,

alongside RK_{i+1} and CK, the directional topic pair of the next epoch

$$\theta_{i+1}^{\mathsf{dir}} = \mathsf{Kec}\big(\mathsf{HKDF}(dh_i,\ RK_{i+1},\ \text{topic-dir})\big), \mathsf{dir} \in \{\mathsf{out}, \mathsf{in}\} \tag{2}$$

where the out label is owned by the party whose fresh key produced dh_i and its peer derives the same pair with labels swapped, following the same intuition of role-based send and receive state assignment [46]. Each message is then emitted as $\mathsf{MSG} = (a_u, \theta_i^{\mathsf{out}}, \mathsf{ct})$ with ct obtained using the fresh key material produced by the double ratchet state, where the input plaintext carries the sender's inner signature under sk_u^{SIG}, which keeps authorship provable after decryption as required by Definition 2, without exposing anything testable to outside observers. Also, the previous inbound topic remains monitored for a bounded transition window, and out-of-order delivery within an epoch falls back on the standard skipped-message-key store [31].

Let us further remark the state consumed in Equation 2. Rotating the public label under which messages are posted and retrieved is not new in metadata-private messaging, where retrieval labels are derived per round from a static pairwise secret [20], [21], so a single compromise of that secret links all past and future epochs. Here θ_{i+1} is a one-way image of (dh_i, RK_{i+1}), and selector unlinkability therefore inherits the FS and PCS of the key schedule as well as its post-quantum property, since the root-key chain is rooted in σ_{hyb}, whose KEM share never appears on-chain (Section VII-C). Ratcheting the delivery label also differs from evolving key material alone. For example, the header-encryption variant of the Double Ratchet [31] and MLS-based messengers [10] both refresh their keys over time but keep delivering to the same identifier. In those settings this is a reasonable choice, since the delivery is done via a server that must route to the recipient anyway and whose view is not public. In our setting, what remains linkable is only confined within an epoch because messages of one epoch share θ_i, and the epoch advances whenever the conversation direction alternates.

VII. Security and privacy analysis

We show here that, even without an anonymous channel, both constructions in Section VI expose a minimal leakage. In Construction 1, an adversarial read provider $\mathcal{A}_{\mathsf{read}}$ controlling P^{r} observes a bulk query for all HS events in a range, together with endpoint metadata ν_r. In Construction 2, the analogous observation is made by the SPS server $\mathsf{Srv}_{\mathsf{SPS}} = (\mathsf{Host}, \mathsf{TEE})$, namely that an adversary $\mathcal{A}_{\mathsf{SPS}}$ controlling the untrusted host may learn that the endpoint identified by ν_r invokes SPS.Receive at time t. This reveals that the endpoint is checking for inbound handshakes, but not which recipient identifier or ledger location is being retrieved (i.e., the latter remain protected by the SPS guarantee, assuming correct attestation and private ORAM accesses). Thus, while in Construction 1 the read provider P^{r} is assumed not to collude with P^{w}, in Construction 2 the corresponding assumption is non-collusion between the untrusted SPS host and P^{w} when the sender publishes HS′. A proof sketch for all the following lemmas and propositions are available in the supplementary materials.

A. Construction 1 analysis

Construction 1 is analyzed under TM-A and TM-$\mathsf{B}_{\mathsf{bulk}}$, and where no external component beyond the pre-registration contract $\mathcal{C}_{\mathsf{reg}}$ is required. Importantly, we assume that $\mathcal{C}_{\mathsf{reg}}$ correctly stores the long-term encapsulation key registered by the recipient, bound to the registering address by transaction authorization and proof of possession, and that ledger queries remain available. The recipient-privacy claim additionally relies on key privacy of the envelope KEM (Definition 1).

Lemma 1 (Correctness)**.** *If an honest sender s encapsulates against $pk_r^{\mathsf{KEM\text{-}lt}}$ retrieved from $\mathcal{C}_{\mathsf{reg}}$ and emits* HS *on $\mathcal{C}$, then an honest recipient r executing* ScanHS *with $sk_r^{\mathsf{KEM\text{-}lt}}$ recovers the body for exactly the events addressed to r, assuming correctness of* ML-KEM*, ciphertext integrity of* AEAD*, and availability of ledger queries.*

Proposition 1 (Recipient-address unlinkability of discovery)**.** *Under* TM-A *and* TM-$\mathsf{B}_{\mathsf{bulk}}$*, and assuming the envelope KEM used in* HS *is key-private (Definition 1) with respect to the registered keys, Construction 1 provides direct unlinkability (Definition 3) between an* HS *event and its intended recipient address a_r, i.e. the recipient first-contact goal of Table II: neither the public ledger observer nor $\mathcal{A}_{\mathsf{read}}$ obtains a direct test for a_r.*

We note that in the bulk-retrieval discussion of Section V-B1, $\mathcal{A}_{\mathsf{read}}$ can combine endpoint attribution with the public selector $\rho_r = H(a_r)$, recover candidate recipients by offline recomputation over the enumerable address set, and compute the set of sender-recipient pairs compatible with the public events and with the set of accounts $\Gamma(\nu_u)$ that $\mathcal{A}_{\mathsf{read}}$ can associate with the endpoint metadata ν_u. However, this is no longer possible with Construction 1, where any residual leakage comes only from $\Gamma(\nu_u)$ and statistical heuristics.

B. Construction 2 analysis

Construction 2 is analyzed under TM-B extended with the SPS server $\mathsf{Srv}_{\mathsf{SPS}} = (\mathsf{Host}, \mathsf{TEE})$ for the discovery of handshake events. We use SPS as a black-box primitive for which we rely on the UC security proved in [29]. In our setting, the SPS location is the ledger position pos of the transaction containing HS′.

Lemma 2 (Correctness)**.** *If an honest sender publishes* HS′ *at ledger location pos and submits the SPS signal* $(\mathsf{RID} = a_r, pos)$*, then an honest registered recipient r obtains pos through* SPS.Receive *and can retrieve the corresponding* HS′ *event from $\mathcal{L}$, assuming correctness of SPS and availability of ledger queries.*

Proposition 2 (Recipient-address unlinkability of discovery)**.** *Under* TM-B*, the SPS security assumptions in [29], and non-collusion between the untrusted SPS host and P^{w}, Construction 2 provides direct unlinkability (Definition 3) between an* HS′ *event and its intended recipient address a_r: neither the public ledger observer nor $\mathcal{A}_{\mathsf{SPS}}$ obtains a direct test for a_r.*

As residual leakage, the targeted $\mathsf{getTransactionReceipt}(pos)$ query lets $\mathcal{A}_{\mathsf{read}}$ associate the endpoint metadata ν_r with the

public sender address a_s of that transaction. However, this does not recreate the attack described in Section V-0b, since any residual link between sender and recipient comes only from $\Gamma(\nu_r)$ and from statistical heuristics.

C. Response and Message topic unlinkability

In this section we show that, regardless of which first contact construction is used, the protocol achieves direct unlinkability (Definition 3) for the conversation-continuity goal of Table II.

a) HS-HSR (C1): For the HS-HSR pair in construction 1, unlinkability follows from two observations. First, the receiver r fetches HS events via an unfiltered bulk query $\mathsf{getLogs}(\mathcal{C}, \mathsf{type} = \mathsf{HS}, \Delta)$, so P^{r} observes no value linked to a_r. We remark that the $\mathcal{C}_{\mathsf{reg}}$ lookup performed by the sender s to retrieve $pk_r^{\mathsf{KEM\text{-}lt}}$ does expose (ν_s, a_r) to P^{r}; however, under the non-collusion assumption between P^{w} and P^{r}, the read provider cannot link the observed a_r to the sender address a_s, since the HS transaction is submitted through P^{w} and the two providers do not share information. Second, the initiator s is also forced to perform a bulk scan to discover the HSR addressed to them, since the response tag τ is computed by r using fresh ephemeral material generated upon receiving HS, which s has not yet observed. Therefore s cannot predict τ before fetching the HSR events, and must attempt to recompute it locally for each retrieved event until a match is found. Consequently, s issues only an unfiltered query to P^{r}, which therefore observes no session-derived value and cannot link any HSR event back to the HS that preceded it.

b) HS-HSR (C2): In Construction 2, the HS-HSR unlinkability argument is somewhat different. The sender s posts the HS′ transaction with no recipient field, and separately submits the private signal carrying $pos = tx_hash$ to the SPS infrastructure. This signaling step is entirely decoupled from P^{r}: the TEE receives and processes the signal without involving the read provider in any way. The receiver r, upon obtaining pos from the SPS infrastructure, issues a targeted query $\mathsf{getTransactionReceipt}(pos)$ to P^{r}. This is the only potentially sensitive read-side observation: P^{r} sees the endpoint metadata ν_r of the receiver linked to the sender address a_s of the transaction at pos. However, this does not recover the full sender-recipient pair (a_s, a_r), since P^{r} has no knowledge of the SPS signal and therefore cannot associate ν_r with a_r beyond what is already captured by $\Gamma(\nu_r)$. The unlinkability of the subsequent HSR discovery then follows by the same bulk scan argument given for Construction 1, since the initiator s must retrieve HSR events and attempt to recompute τ locally from the ephemeral material contained therein.

c) HSR-MSG (C1, C2): Once the initiator gets the event with tag τ, only he can compute the root key with the cryptographic material and start the session posting the first encrypted message on topic θ_0 (Equation 1) which is visible to all observers in $\mathsf{View}_{\mathsf{pub}}$, making the link (a_s, θ_0). Now a_r starting from the same hybrid secret can compute the same θ_0 and query P^{r} for that topic. All that P^{r} can correlate is the sender address with the receiver endpoint metadata: (a_s, ν_r).

d) MSG-MSG (C1, C2): Thanks to topic rotation induced by the topic ratchet, while the posting account $a_u \in \{a_s, a_r\}$ is always observable in $\mathsf{View}_{\mathsf{pub}}$ as part of the pair (a_u, θ_i), consecutive topics θ_i and θ_{i+1} are computationally unlinkable to any party that did not participate in the session, since each new topic is derived from fresh ratchet state via the one-way derivation of Equation 2. Consequently, no observer can establish conversation continuity across epochs, and the relation (a_s, a_r) cannot be recovered by composing observations across multiple MSG events. A discussion on the Event logs content security is finally provided in the supplementary material.

TABLE III: Local client processing overhead in the baseline setting.

Operation	Scaling	Median (ms)	p95 (ms)	Local work
HS lookup by indexed ρ_r	$O(1)$	< 0.01	< 0.01	Local indexed lookup
MSG topic re-computation	$O(1)$	0.018	0.028	HKDF/SHA-256 and Keccak
DH ratchet step	$O(1)$	1.96	2.31	Scalar mult., root key update
HSR matching	$O(P)$	7.74	8.68	Decrypt over pending contacts

Measurements are execution times after warm-up with libraries `noble/post-quantum`, `noble/hashes`, `ethers` and `tweetnacl`. $P = 10$ is the number of pending ephemeral secrets that the initiator has in memory.

VIII. Performance and scalability evaluation

All local benchmarks were run on a MacBook Air with an Apple M2 SoC, 8 CPU cores and 24 GB of memory, running macOS 26.2 and Node.js v20.15.1. The results in Table III show that most local client operations are effectively constant with respect to the number of user contacts and global traffic volume, because the protocol uses fixed-size cryptographic inputs. In the basic setting, the only exception is the matching for the received HSR event where initiators scan and attempt to open the response payload under each pending ephemeral secret. Only the decrypting candidate reaches ML-KEM decapsulation and hybrid τ recomputation. Therefore, the scan grows linearly in the number of pending outbound handshakes, but it is likely to remain small as pending entries can be removed once their responses arrive.

The gas used for message transport contract is governed by calldata and log data size as there is no contract logic. Table IV reports the measured costs also for the two first-contact variants introduced in Section VI-A and Section VI-B. Unlike Construction 2, where if anything the cost is a few bytes lower due to the absence of the recipient selector field, Construction 1 pays part of the privacy cost directly in the handshake event as the sender must publish enough encrypted material for every potential recipient to attempt local discovery without a public selector, namely a 1088-byte ML-KEM-768 envelope ciphertext alongside a 1561-byte encrypted envelope. This is in fact why its cost is closer to the measured HSR response, which also carries a post-quantum ciphertext-sized object. As for the two MSG rows, a tenfold increase in ciphertext size produces only a $2.8\times$ increase in gas, because a large part of the transaction cost is already paid before payload bytes become dominant. Over this interval, the observed marginal cost is $\simeq 40.1$ gas per additional byte, matching the post-Pectra calldata floor pricing of EIP-7623

TABLE IV: EVM gas for transport operations.

Call to emit event	Gas	Base USD[a]	L1 USD[a]
Baseline HS request	85,790	0.0069	0.1915
Construction 1 HS request	128,680	0.0104	0.2872
Construction 2 HS′ request	84,630	0.0068	0.1889
HSR response	129,140	0.0104	0.2882
MSG, 145-byte blob	29,850	0.0024	0.0666
MSG, 1450-byte blob	82,150	0.0066	0.1833

[a] These figures were obtained with $gas \cdot \overline{p}_{gas} \cdot p_{ETH}$ with $p_{ETH} = 2{,}264.78$ USD/ETH. The gas-price averages are daily averages over the latest three-month window available in the explorer CSVs at the time of writing: $\overline{p}_{gas} = 0.0356$ gwei on Base and $\overline{p}_{gas} = 0.9854$ gwei on Ethereum L1. The Base column does not include any additional L1 data fee component.

[47] (40 gas per non-zero byte). This is relevant because the current implementation pads plaintext before encryption into power-of-two buckets with bounded random jitter. As a result, this metadata protection has a direct but predictable cost such that larger messages pay an extra step only when they cross the next bucket boundary.

A. Realistic costs for contact discovery comparison

In Table V, we write λ_{HS} for the average number of handshake events per block, assuming 12s blocks, a 5-minute retrieval window and a fetched log size of 6 KiB (i.e., the approximate JSON-RPC transfer size of one fetched Construction 1 handshake log). The last three rows express handshake density as a fraction of the Ethereum L1 target gas per block (i.e., 30M gas units). We also report a normalization based on realistic daily active users to make the traffic easier to interpret.

In our instantiation, using the SPS benchmark of less than 70 ms per send when the service is provisioned for $N = 10^7$ stored signals and $M = 10^6$ registered recipients [29], a single sequential TEE would process about $1/0.07 \approx 14.3$ signals per second, i.e., roughly 171 signals per 12s block. Under this conservative interpretation without parallelism, all regimes up to the 20% L1-target row remain below one such TEE's send-side capacity. This indicates that in Construction 2 recipient discovery bandwidth remains essentially independent of the global handshake density. Instead, the expected number of stored SPS signal nodes is $7200 \cdot days \cdot \lambda_{\mathsf{HS}}$, hence the 20% L1-target regime would produce approximately $5.0 \cdot 10^5$ signal nodes per day. We show in the right panel of Figure 2 that for historical event logs, Construction 1 is in fact limited by the amount of HS data that the recipient must download and scan locally: if we assume that ordinary clients can operate easily up to around 1 GiB, then a 30 days scan is feasible for a traffic scenario of $\lambda_{\mathsf{HS}} \approx 1$. This is much lower than the traffic level supported by Construction 2 for the same retention horizon. The tradeoff is that SPS stores these signal nodes as encrypted state behind ORAM outside the TEE, so the external database size grows proportionally with the global handshake volume and the chosen retention window.

When considering background polling for continuous HS discovery, however, the bottleneck of Construction 2 is no longer the insertion rate alone. Every active recipient may invoke SPS.Receive even when no new handshake is waiting for them. Thus, while Construction 1 pays a cost proportional to the number of HS events in the queried block range, Construction 2 relies on a service whose cost depends on the number of active recipients and their polling timing. By using the same equivalence for daily active usage in Table V, one can translate a given λ_{HS} into an approximate population of users that may be polling the SPS service. Although this is only a normalization, since real deployments would depend on notification strategy and mobile background limits, it still gives a useful upper-bound estimate of the pressure induced by a simple periodic polling strategy. With a representative 5-minute polling interval, each additional HS per block corresponds to roughly 240 SPS receive calls per second. For example, as shown in the left panel of Figure 2, at the same traffic level where Construction 1 local decryption begins to become arguably impractical (i.e. 250 ms per check on our reference laptop), Construction 2 would require about 3.0k receive calls per second, or about 210 sequential worker equivalents at the ∼70 ms per-request receive cost measured in [29] for retrievals $\bar{l} = 1$ at the same provisioning, a non-trivial service cost as this would correspond to roughly seven 32-vCPU confidential compute parent instances, i.e. about $10/hour at current AWS on-demand prices. Specifically, these worker estimates are best thought of as a lower bound, since each SPS.Receive performs exactly $\bar{l}$ ORAM reads regardless of how many signals are pending, thus the per-request cost grows linearly in the provisioned $\bar{l}$, about 6 s at $\bar{l} = 100$ [29].

TABLE V: The C1 columns report per-recipient client bandwidth for Construction 1, which scales linearly in the log size. The SPS column reports the total number of private-signaling notifications processed by the Construction 2 infrastructure, one per handshake.

Traffic regime	DAU-eq.[a]	λ_{HS}	C1 / 5 min (MiB)	C1 / day (GiB)	SPS signals / day
Low	7.2k	0.1	0.01	0.004	720
Moderate	72k	1	0.15	0.04	7,200
Busy	720k	10	1.46	0.41	72,000
5% L1 gas	1.26M	17.5	2.56	0.72	125,889
20% L1 gas	5.04M	69.9	10.24	2.88	503,555
100% L1 gas	25.18M	349.7	51.22	14.41	2,517,776

[a] This equivalance assumes $h = 0.1$ first-contact handshakes per active user per day, i.e. one new handshake event every ten days. For $h = 0.02$, multiply this column by 5, while for $h = 0.5$ divide it by 5.

IX. Limitations and future work

Since we emphasized that the guarantees of Section VII are scoped to the direct attacks, we now state the families of statistical attacks that remain applicable to our protocol:

- *Timing correlation.* An HSR event posted shortly after an HS, or a fresh topic θ_{i+1} becoming active in the blocks where θ_i falls silent, can be associated probabilistically. The read provider can additionally observe when an endpoint polls and how soon after publication a given range is first retrieved. Timing-based profiling and deanonymization have been demonstrated on Ethereum at both the application and the network level [40], [41].
- *Anonymity-set collapse at low traffic.* Every claim in Section VII leaves the adversary with a candidate set, and the operational value of the guarantee is the size of that

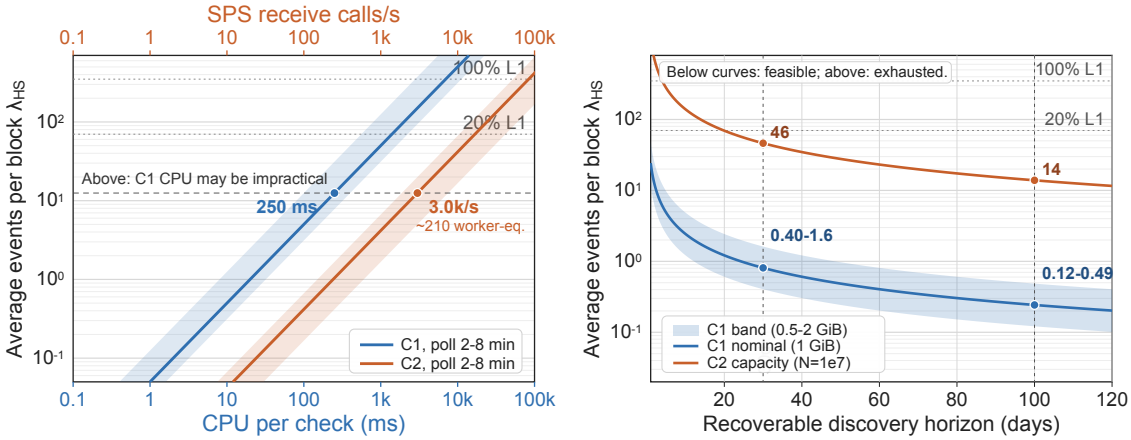

Fig. 2: *Left panel*: In Construction 1, polling cost is based on a measured 0.8 ms per trial decryption. In Construction 2, one worker-equivalent is a sequential SPS receive worker at 70 ms per operation, the cost measured in [29] for single-location retrievals at $N = 10^7$, $M = 10^6$. *Right panel*: Historical recovery with ≈ 1 GiB client bandwidth and SPS capacity $N = 10^7$.

set within the relevant observation window, as in classical anonymity metrics. At the Low regime of Table V ($\lambda_{\mathsf{HS}} = 0.1$), an HSR appearing within a few blocks of a single pending HS has essentially one plausible predecessor. Notably, this differs in Construction 1 where the recipient anonymity set is at most the population registered in $\mathcal{C}_{\mathsf{reg}}$, which is small precisely during early deployment, and in Construction 2 where it is at most the recipients registered with the SPS service.

- *Long-term intersection attacks.* Despite topic ratcheting, an account a_r tends to post within a window after a_s, repeatedly, across epochs. An observer accumulating $\mathsf{View}_{\mathsf{pub}}$ over weeks can thus recover persistent conversation pairs via statistical disclosure attacks on round-based anonymity systems [48].

Our design already includes measures that blunt these types of attack, for example random padded plaintexts before encryption or the bounded topic-transition window of Section VI-C limiting the period in which a client's read path simultaneously monitors two adjacent epochs. Moreover, for first-contact discovery, a Construction 1 recipient issues the same bulk query whether or not any event is addressed to them, and Construction 2 responses are padded to $\bar{l}$ locations with dummy ORAM reads regardless of pending volume. Further standard countermeasures are not taken into account for this paper, e.g. those on randomized polling or publication delays, or even decoy events that manufacture a synthetic anonymity set. None of these is conceptually incompatible with Verbeth, but their monetary and usability cost would arguably be unsustainable. Cost aside, there is a more fundamental reason to regard the residual leakage acceptable: since the claims of Section VII degrade with the size of the active population, a countermeasure that creates more friction ends up reducing the very anonymity sets it was meant to protect [49].

Finally, how does non-repudiation hold against potential traffic-analysis countermeasures? With respect to our Definition 2, we argue that randomized publication delays would loosen the temporal bound of part (ii), and relayed submission would move the provable publisher of part (i) away from the sender. Future work should characterize these tradeoffs precisely, and in particular ask whether users can choose between a submission procedure optimized for attribution and one optimized for metadata privacy.

X. Conclusion

We demonstrate that public blockchains can be transformed into robust, private communication channels without requiring additional messaging infrastructure or assuming clients run full nodes. Our system reconciles the transparency of event logs with the strict confidentiality and unlikability requirements of modern end-to-end messaging. Owing to its dual discovery design, Verbeth can accommodate different privacy and deployment assumptions while retaining its core security properties. This opens a promising path toward deployable, truly censorship-resistant communication networks operating natively on global decentralized infrastructure.

Acknowledgment

The authors thank I. Visconti for helpful discussions.

References

[1] G. Almashaqbeh and R. Solomon, "Sok: Privacy-preserving computing in the blockchain era," in *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2022, pp. 124–139.

[2] M. Esposito, A. Rizzini, F. Bruschi, and D. Sciuto, "Verbeth: Secure messaging over the blockchain event logs," 2026, bRAIN 2026: Seventh Workshop on Blockchain theoRy and ApplicatIoNs.

[3] N. Borisov, I. Goldberg, and E. Brewer, "Off-the-record communication, or, why not to use pgp," in *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society*, 2004, pp. 77–84.

[4] N. Unger, S. Dechand, J. Bonneau, S. Fahl, H. Perl, I. Goldberg, and M. Smith, "Sok: secure messaging," in *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 232–249.

[5] J. Zhou and D. Gollmann, "Evidence and non-repudiation," *Journal of Network and Computer Applications*, vol. 20, no. 3, pp. 267–281, 1997.

[6] S. Kremer, O. Markowitch, and J. Zhou, "An intensive survey of fair non-repudiation protocols," *Computer communications*, vol. 25, no. 17, pp. 1606–1621, 2002.

[7] P. Abla, T. Li, D. He, H. Huang, S. Yu, and Y. Zhang, "Fair and privacy-preserved data trading protocol by exploiting blockchain," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 9012–9025, 2024.

[8] C. Zhang, H. Xuan, T. Wu, X. Liu, G. Yang, and L. Zhu, "Blockchain-based dynamic time-encapsulated data auditing for outsourcing storage," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 1979–1993, 2023.

[9] V. Zieglmeier, G. Loyola Daiqui, and A. Pretschner, "Decentralized inverse transparency with blockchain," *Distributed Ledger Technologies: Research and Practice*, vol. 2, no. 3, pp. 1–28, 2023.

[10] M. Galligan, P. Anchuri, S. Carlin, J. Hasegawa, M. Kobetic, N. Molnar, and J. Turner-Baggs, "The XMTP litepaper," XMTP Labs, Tech. Rep., Jul. 2025. [Online]. Available: https://github.com/xmtp/litepaper

[11] f. CassOnMars (maintainer, "Fip: Direct casts #99," https://github.com/farcasterxyz/protocol/discussions/99, Jun. 2023, gitHub Discussion.

[12] Y. Dodis, D. Jost, S. Katsumata, T. Prest, and R. Schmidt, "Triple ratchet: A bandwidth efficient hybrid-secure signal protocol," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2025, pp. 302–331.

[13] H. Cornelius, S. Tikhomirov, A. Revuelta, S. P. Vivier, and A. Challani, "The waku network as infrastructure for dapps," in *2024 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, Jul. 2024, pp. 31–32. [Online]. Available: https://ieeexplore.ieee.org/document/10646404

[14] K. Jefferys, M. Shishmarev, and S. Harman, "Session: End-to-end encrypted conversations with minimal metadata leakage," https://arxiv.org/abs/2002.04609, Feb. 2020.

[15] K. Bhargavan, C. Jacomme, F. Kiefer, and R. Schmidt, "Formal verification of the PQXDH Post-Quantum key agreement protocol for end-to-end secure messaging," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 469–486.

[16] R. Recabarren and B. Carbunar, "Tithonus: A bitcoin based censorship resilient system," *Proceedings on Privacy Enhancing Technologies*, 2019. [Online]. Available: https://doi.org/10.2478/popets-2019-0005

[17] ——, "Toward uncensorable, anonymous and private access over satoshi blockchains," *Proceedings on Privacy Enhancing Technologies*, 2022. [Online]. Available: https://doi.org/10.2478/popets-2022-0011

[18] M. Minaei, P. Moreno-Sanchez, and A. Kate, "Moneymorph: Censorship resistant rendezvous using permissionless cryptocurrencies," *Proceedings on Privacy Enhancing Technologies*, vol. 2020, no. 3, 2020.

[19] T. Tiemann, S. Berndt, T. Eisenbarth, and M. Liśkiewicz, ""act natural!": Exchanging private messages on public blockchains," in *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2023, pp. 292–308.

[20] J. Van Den Hooff, D. Lazar, M. Zaharia, and N. Zeldovich, "Vuvuzela: Scalable private messaging resistant to traffic analysis," in *Proceedings of the 25th Symposium on Operating Systems Principles*, 2015, pp. 137–152.

[21] S. Angel and S. Setty, "Unobservable communication over fully untrusted infrastructure," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 551–569.

[22] N. Tyagi, Y. Gilad, D. Leung, M. Zaharia, and N. Zeldovich, "Stadium: A distributed metadata-private messaging system," in *Proceedings of the 26th Symposium on Operating Systems Principles*, 2017, pp. 423–440.

[23] D. Lazar, Y. Gilad, and N. Zeldovich, "Karaoke: Distributed private messaging immune to passive traffic analysis," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 2018, pp. 711–725.

[24] G. Beck, J. Len, I. Miers, and M. Green, "Fuzzy message detection," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 1507–1528.

[25] Z. Liu and E. Tromer, "Oblivious message retrieval," in *Annual International Cryptology Conference*. Springer, 2022, pp. 753–783.

[26] Z. Liu, E. Tromer, and Y. Wang, "{PerfOMR}: Oblivious message retrieval with reduced communication and computation," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 3169–3186.

[27] ——, "Group oblivious message retrieval," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024, pp. 4367–4385.

[28] V. Madathil, A. Scafuro, I. A. Seres, O. Shlomovits, and D. Varlakov, "Private signaling," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 3309–3326.

[29] S. Jakkamsetti, Z. Liu, and V. Madathil, "Scalable private signaling," in *2025 IEEE 38th Computer Security Foundations Symposium (CSF)*, 2025, pp. 143–158.

[30] W. Diffie and M. E. Hellman, "New directions in cryptography," in *Democratizing cryptography: the work of Whitfield Diffie and Martin Hellman*, 2022, pp. 365–390.

[31] T. Perrin and M. Marlinspike, "The double ratchet algorithm," https://signal.org/docs/specifications/doubleratchet/, 2016, signal Messenger.

[32] J. Zhou and D. Gollman, "A fair non-repudiation protocol," in *Proceedings 1996 IEEE symposium on security and privacy*. IEEE, 1996, pp. 55–61.

[33] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 1994, pp. 124–134.

[34] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 212–219.

[35] National Institute of Standards and Technology, "Module-lattice-based key-encapsulation mechanism standard," NIST, Tech. Rep. FIPS 203, 2024, https://doi.org/10.6028/NIST.FIPS.203.

[36] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval, "Key-privacy in public-key encryption," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2001, pp. 566–582.

[37] V. Botta, D. Friolo, D. Venturi, and I. Visconti, "Shielded computations in smart contracts overcoming forks," in *International Conference on Financial Cryptography and Data Security*. Springer, 2021, pp. 73–92.

[38] F.-X. Wicht, Z. Wang, D. V. Le, and C. Cachin, "A transaction-level model for blockchain privacy," in *International Conference on Financial Cryptography and Data Security*. Springer, 2024, pp. 293–310.

[39] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage, "A fistful of bitcoins: characterizing payments among men with no names," in *Proceedings of the 2013 conference on Internet measurement conference*, 2013, pp. 127–140.

[40] F. Béres, I. A. Seres, A. A. Benczúr, and M. Quintyne-Collins, "Blockchain is watching you: Profiling and deanonymizing ethereum users," in *2021 IEEE international conference on decentralized applications and infrastructures (DAPPS)*. IEEE, 2021, pp. 69–78.

[41] S. Wang, M. Yang, Y. Liu, Y. Zhang, S. Zhang, Z. Ling, J. Cao, and X. Fu, "Time tells all: Deanonymization of blockchain rpc users with zero transaction fee," in *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, 2025, pp. 3490–3504.

[42] P. Chatzigiannis, F. Baldimtsi, and K. Chalkias, "Sok: Blockchain light clients," in *International Conference on Financial Cryptography and Data Security*. Springer, 2022, pp. 615–641.

[43] A. Pfitzmann and M. Hansen, "A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management," 2010.

[44] A. Wahrstätter, M. Solomon, B. DiFrancesco, V. Buterin, and D. Svetinovic, "Basesap: Modular stealth address protocol for programmable blockchains," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 3539–3553, 2024.

[45] T. Wahrstätter, M. Solomon, B. DiFrancesco, and V. Buterin, "ERC-5564: Stealth addresses," Ethereum Improvement Proposals, Standards Track 5564, Aug. 2022. [Online]. Available: https://eips.ethereum.org/EIPS/eip-5564

[46] T. Perrin, "The noise protocol framework," 2018.

[47] T. Wahrstätter and V. Buterin, "EIP-7623: Increase calldata cost," 2024, https://eips.ethereum.org/EIPS/eip-7623.

[48] N. Mathewson and R. Dingledine, "Practical traffic analysis: Extending and resisting statistical disclosure," in *International workshop on privacy enhancing technologies*. Springer, 2004, pp. 17–34.

[49] R. Dingledine and N. Mathewson, "Anonymity loves company: Usability and the network effect." in *WEIS*, 2006.

Supplemental Material

Marco Esposito, Andrea Rizzini, Francesco Bruschi, and Donatella Sciuto

SUPPLEMENT MATERIALS FOR PRELIMINARIES

Blockchain and Smart Contract

Blockchain is a technology used for the creation of distributed ledgers primarily recording transactions. Formally, a blockchain $\mathcal{B}$ is an ordered set of blocks $\mathcal{B} = (b_0, b_1, \ldots, b_n)$ and each block is defined by the tuple (h_{i-1}, D_i, t_i, n_i), where h_{i-1} is the hash of the previous block, D_i is the set of all the transactions included in that specific block, t_i is the timestamp of creation and n_i is a nonce. Each new block is added in a distributed manner leveraging a consensus mechanism that guarantees the append-only property and immutability of the chain itself; among the most well-known consensus algorithms we find Proof of Work (PoW) adopted e.g. by Bitcoin [10], and Proof of Stake (PoS) adopted e.g. by Ethereum [11]. Furthermore, we can distinguish between public (permissionless) and private (permissioned) blockchains; in the case of the former, all the blocks and transactions are publicly viewable through the so called explorer, providing an important property called transparency. A smart contract is a computer program deployed on a blockchain which runs automatically upon user calls, according to the terms and logic encoded in the contract itself [12]. This concept gained popularity thanks to the Ethereum network and its native support, but it is worth noting that not all blockchains have native support for them. A smart contract once deployed will be immutable and transparent in its execution, thanks to the intrinsic properties of the blockchain.

World state database

The blockchain's infrastructure is composed of a set of distributed machines, also called nodes; anyone with a machine supporting the blockchain software can participate and be part of the distributed protocol. Among the different types of nodes we find the full nodes, entities that download and verify all blocks and transactions since the genesis block, maintaining the current state of the blockchain. Focusing on the Ethereum blockchain, when a function is called on a smart contract, it is common for logs (also called events) to be emitted on the blockchain during execution. Referring to Solidity, the language used to write smart contracts for Ethereum, the standard syntax is `emit EventName(args[])`. In each Ethereum block's header we find, among other fields, the receipt trie root, i.e. the root of a Merkle Patricia Trie encoding the receipt of each transaction in the block. Formally, denoting with $\mathcal{R}_b = \{r_1, \ldots, r_n\}$ the set of transaction receipts for a block b, each $r_i \in \mathcal{R}_b$ contains an array of the emitted logs, and is stored as a leaf of the receipt trie.

Manuscript received Month Day, Year; revised Month Day, Year.

The receipt trie, maintained by full nodes, allows the retrieval of the logs emitted by smart contracts during transaction execution. However, for efficient querying of logs across a large range of blocks, indexing services such as The Graph are commonly adopted in practice.

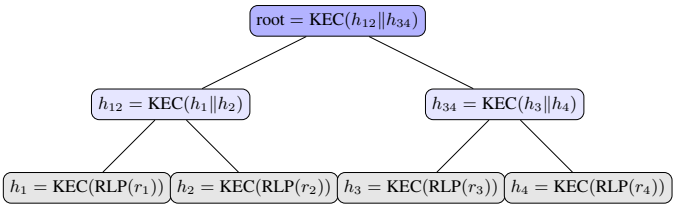

Fig. 1: Receipt Trie in the case of four transactions, where RLP is a specific format for serialization before hashing, and KEC is the Keccak hash function used in Ethereum.

Double Ratchet

Algorithm 1 SymRatchet(CK)

Require: Chain Key CK_i
Ensure: Message Key MK_i, updated Chain Key CK_{i+1}
1: $MK_i \leftarrow \text{KDF}(CK_i, \texttt{0x01})$
2: $CK_{i+1} \leftarrow \text{KDF}(CK_i, \texttt{0x02})$
3: **delete** CK_i
4: **return** $MK_i,\ CK_{i+1}$

Algorithm 2 DHRatchet($RK_i, sk_i, \hat{pk}$)

Require: Root Key RK_i, own ephemeral secret key sk_i, remote ephemeral public key $\hat{pk}$
Ensure: Updated Root Key RK_{i+1}, new Chain Key CK, new ephemeral key pair (sk_{i+1}, pk_{i+1})
1: $dh_i \leftarrow \text{DH}(sk_i, \hat{pk})$
2: $RK_{i+1},\ CK \leftarrow \text{KDF}(RK_i, dh_i)$
3: $(sk_{i+1}, pk_{i+1}) \leftarrow$ GenerateKeyPair()
4: **delete** $sk_i,\ dh_i$
5: **return** $RK_{i+1},\ CK,\ (sk_{i+1}, pk_{i+1})$

Algorithm 3 SendMessage($state, plaintext$)

Require: Protocol state $state$, plaintext m
Ensure: Ciphertext c, updated $state$
1: **if** $state.CK_s = \emptyset$ **then** ▷ First message after receiving: step DH ratchet
2: $state.RK,\ state.CK_s,\ (state.sk, state.pk) \leftarrow \text{DHRatchet}(state.RK,\ state.sk,\ state.\hat{pk})$
3: **end if**
4: $MK,\ state.CK_s \leftarrow \text{SymRatchet}(state.CK_s)$
5: $c \leftarrow \text{Sym.Encrypt}(MK,\ m,\ \text{header}(state.pk))$
6: **delete** MK
7: **return** $c,\ state$

TABLE I: Representative prior cecentralized and wallet-based messengers along the axes relevant to this work. "Infrastructure view" summarizes what the party serving a recipient's retrieval learns about who is being contacted; "PQ" indicates whether session establishment resists a quantum adversary.

System	Ledger role	Delivery infrastructure	Asynch. first contact	Infrastructure view	PQ
XMTP [1]	identity anchor	dedicated node network	key packages served by nodes	nodes learn mailbox and fetch pattern	hybrid
DirectCasts [2]	identity registry	Farcaster hubs	pre-key bundles on hubs	hubs learn conversation participants	no
Waku [3]	none	libp2p relay and store nodes	X3DH pre-key bundles	store nodes learn content-topic filters	no
Session [4]	Sybil resistance for nodes	onion-routed service-node swarms	sent to swarm derived from recipient ID	swarm learns mailbox; onion routing hides requester	no
Vuvuzela [5]	none	mix cascade with cover traffic	out-of-band dialing secret	hidden if one mix server is honest	no
Pung [6]	none	untrusted store with PIR	out-of-band shared secret	hidden by PIR	no
MoneyMorph [7], [8]	covert transport	none	out-of-band pre-shared keys	full-chain scan from own node	no
Verbeth	native log events	none (optional TEE for discovery only [9])	on-chain hybrid handshake	no recipient selector in events or queries	hybrid

Algorithm 4 ReceiveMessage$(state, c, header)$

Require: Protocol state $state$, ciphertext c, message header $header$
Ensure: Plaintext m, updated $state$
1: **if** $header.pk \neq state.\hat{pk}$ **then** ▷ New DH public key: step DH ratchet
2: $state.\hat{pk} \leftarrow header.pk$
3: $state.RK, \quad state.CK_r, \quad (state.sk, state.pk) \leftarrow$ DHRatchet$(state.RK,\ state.sk,\ state.\hat{pk})$
4: **end if**
5: $MK,\ state.CK_r \leftarrow$ SymRatchet$(state.CK_r)$
6: $m \leftarrow$ Sym.Decrypt$(MK,\ c,\ header)$
7: **delete** MK
8: **return** $m,\ state$

Trusted Execution Environment

A TEE, or enclave, is a hardware-backed isolated execution environment that guarantees confidentiality and integrity of the code and data being executed, even in the presence of a malicious host. Among the most common and well-known solutions we can find Intel SGX/TDX, ARM TrustZone, AMD SEV and AWS Nitro [13]. For our purposes, the relevant abstraction is remote attestation where clients can verify that the expected private signaling code is running inside the enclave before relying on it to process private signals.

Private Signaling with TEE

In the following we provide the algorithm showing how it is possible to realize PS leveraging a TEE. The original TEE construction proposed in [14] stores a table of size $M \times \bar{l}$ where M is the number of registered recipients and $\bar{l}$ is the maximum number of pending signals stored per recipient in one row, resulting in a runtime cost per signal of $\mathcal{O}(M \times \bar{l})$. The Scalable Private Signaling (SPS) construction proposed in [9] improves upon this by replacing the fixed table with ORAM-backed linked lists where each signal is stored as an encrypted node pointing to the previous signal for the same recipient, and the latest node for each recipient is itself stored through ORAM. Concretely, SPS uses an ORAM database DB for encrypted list nodes, $\mathsf{DB}_{\mathsf{recip}}$ for the latest signal of each recipient identifier RID, and $\mathsf{DB}_{\mathsf{keys}}$ for recipient keys. Its send cost is $\mathcal{O}(\log M + \log N)$, where N is the number of stored signals, while each receive request returns up to $\bar{l}$ locations at cost $\mathcal{O}(\bar{l} \log N + \log M)$. Since a detailed ORAM overview appears in [9, Sec. 2] and the TEE protocol in [9, Sec. 6, Figs. 3–4], in the rest of this paper, SPS is used as a black-box service exposing the operations $\mathsf{SPS.Setup}$, $\mathsf{SPS.Register}$, $\mathsf{SPS.Send}$, and $\mathsf{SPS.Receive}$. A sender submits a private signal containing a recipient identifier RID and a bulletin-board location loc; the service privately updates the recipient's linked list; and the recipient later receives a vector of locations, together with the continuation behavior defined by SPS when fewer or more than $\bar{l}$ pending signals exist.

Hybrid key-exchange method

Algorithm 5 $\mathsf{HybKE.Init}()$ - Run by Initiator

Require: Security parameter 1^λ
Ensure: Public message msg_A to send to Responder, private state state_A
1: $(sk_A^{\mathsf{cls}}, pk_A^{\mathsf{cls}}) \leftarrow \mathsf{DH}_{\mathsf{cls}}.\mathsf{KeyGen}()$
2: $(sk_A^{\mathsf{kem}}, pk_A^{\mathsf{kem}}) \leftarrow \mathsf{KEM}_{\mathsf{pqc}}.\mathsf{KeyGen}()$
3: **return** $\mathsf{msg}_A = (pk_A^{\mathsf{cls}}, pk_A^{\mathsf{kem}}), \quad \mathsf{state}_A = (sk_A^{\mathsf{cls}}, sk_A^{\mathsf{kem}})$

Algorithm 6 $\mathsf{HybKE.Respond}(\mathsf{msg}_A)$ - Run by Responder upon receiving msg_A from Initiator

Require: Public message $\mathsf{msg}_A = (pk_A^{\mathsf{cls}}, pk_A^{\mathsf{kem}})$ from Initiator
Ensure: Public message msg_B to send to Initiator, shared session key K
1: Parse $\mathsf{msg}_A = (pk_A^{\mathsf{cls}}, pk_A^{\mathsf{kem}})$
2: $(sk_B^{\mathsf{cls}}, pk_B^{\mathsf{cls}}) \leftarrow \mathsf{DH}_{\mathsf{cls}}.\mathsf{KeyGen}()$
3: $(ct^{\mathsf{kem}}, ss^{\mathsf{kem}}) \leftarrow \mathsf{KEM}_{\mathsf{pqc}}.\mathsf{Encaps}(pk_A^{\mathsf{kem}})$
4: $ss^{\mathsf{dh}} \leftarrow \mathsf{DH}_{\mathsf{cls}}.\mathsf{ComputeKey}(sk_B^{\mathsf{cls}}, pk_A^{\mathsf{cls}})$
5: $K \leftarrow \mathsf{KDF}(ss^{\mathsf{dh}} \| ss^{\mathsf{kem}})$
6: **return** $\mathsf{msg}_B = (pk_B^{\mathsf{cls}}, ct^{\mathsf{kem}}), \quad K$

Algorithm 7 $\mathsf{HybKE.Finish}(\mathsf{state}_A, \mathsf{msg}_B)$ - Run by Initiator upon receiving msg_B from Responder

Require: Private $\mathsf{state}_A = (sk_A^{\mathsf{cls}}, sk_A^{\mathsf{kem}})$, public $\mathsf{msg}_B = (pk_B^{\mathsf{cls}}, ct^{\mathsf{kem}})$ from Responder
Ensure: Shared session key K matching the one derived by Responder
1: Parse $\mathsf{state}_A = (sk_A^{\mathsf{cls}}, sk_A^{\mathsf{kem}})$
2: Parse $\mathsf{msg}_B = (pk_B^{\mathsf{cls}}, ct^{\mathsf{kem}})$
3: $ss^{\mathsf{kem}} \leftarrow \mathsf{KEM}_{\mathsf{pqc}}.\mathsf{Decaps}(sk_A^{\mathsf{kem}}, ct^{\mathsf{kem}})$
4: $ss^{\mathsf{dh}} \leftarrow \mathsf{DH}_{\mathsf{cls}}.\mathsf{ComputeKey}(sk_A^{\mathsf{cls}}, pk_B^{\mathsf{cls}})$
5: $K \leftarrow \mathsf{KDF}(ss^{\mathsf{dh}} \| ss^{\mathsf{kem}})$
6: **return** K

SECURITY AND PRIVACY ANALYSIS

Definition 1 (KEM key privacy)**.** *A Key encapsulation method* $\mathsf{KEM} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ *is key-private if for every PPT adversary* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$*, the following experiment outputs* 1 *with probability at most* $1/2 + \mathsf{negl}(\lambda)$*. Sample* $(pk_0, sk_0), (pk_1, sk_1) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and* $b \leftarrow \{0,1\}$*. Let* $s \leftarrow \mathcal{A}_0(pk_0, pk_1)$*, compute* $(ct^\star, K^\star) \leftarrow \mathsf{Encaps}(pk_b)$*, and let* $b' \leftarrow \mathcal{A}_1(ct^\star, s)$*. The experiment outputs* 1 *iff* $b' = b$*.*

Proof sketch for Lemma 1

For an event addressed to r, decapsulation correctness of ML-KEM [15] yields the K_{env} derived by s, and AEAD.Dec succeeds under the locally reconstructed $\mathsf{ad} = a_s \,\|\, a_r$. For an event addressed to some $r' \neq r$, decapsulation does not fail explicitly since ML-KEM uses the variant with implicit rejection of the Fujisaki–Okamoto transform [15], so $\mathsf{Decaps}(sk_r^{\mathsf{KEM\text{-}lt}}, \cdot)$ on a ciphertext encapsulated to another key returns a key that is pseudorandom and independent of the encapsulated K_{env}. AEAD decryption under such a key then fails except with negligible probability by ciphertext integrity. Such trial-decryption behavior is indeed the robustness property analyzed for KEM–DEM hybrids built from implicitly rejecting KEMs in [16]. Hence our Algorithm 10 accepts exactly the events addressed to r, up to negligible error.

For an event addressed to r, decapsulation correctness of ML-KEM [15] yields the K_{env} derived by s, and AEAD.Dec succeeds under the locally reconstructed $\mathsf{ad} = a_s \| a_r$. For an event addressed to $r' \neq r$, cross-key robustness of the chosen KEM–AEAD hybrid ensures that trial decryption under $sk_r^{\mathsf{KEM\text{-}lt}}$ is rejected except with negligible probability. This property must be required separately: implicit rejection of the KEM alone does not generically imply robustness of the hybrid PKE [16]. Hence Algorithm 10 accepts exactly the events addressed to r, up to negligible error.

Proof sketch for Proposition 1

We enumerate the values exposed in each view. The public event carries $(a_s, \mathsf{ct}_{\mathsf{kem}}^{\mathsf{env}}, \mathsf{ct}^{\mathsf{env}})$ and no indexed recipient field. A direct test mapping $\mathsf{ct}_{\mathsf{kem}}^{\mathsf{env}}$ to a candidate a_r would distinguish an encapsulation under $pk_r^{\mathsf{KEM\text{-}lt}}$ from one under another registered key, contradicting Definition 1. This notion has been established for implicitly rejecting FO-transformed KEMs [16] and specifically for the Kyber scheme [17]. The body $\mathsf{ct}^{\mathsf{env}}$ is an AEAD ciphertext under the single-use K_{env} and the associated data $a_s \,\|\, a_r$ is authenticated but never published. Also, ciphertext lengths do not depend on the recipient, since the ML-KEM-768 ciphertext is fixed-size and the body has a fixed format. On the read side, $\mathcal{A}_{\mathsf{read}}$ additionally observes only bulk queries $\mathsf{getLogs}(\mathcal{C}, \mathsf{type} = \mathsf{HS}, \Delta)$ together with endpoint metadata ν_r, neither of which contains a value derived from a_r.

Proof sketch for Lemma 2

Correctness of the SPS functionality [9, Sec. 6] guarantees that a signal (RID, loc) accepted by the TEE is appended to the linked list of the recipient registered under RID, and that SPS.Receive returns the pending locations of that list, up to the $\bar{l}$-bounded batching behavior recalled above in *Private Signaling with TEE*. Our instantiation changes only the identifier assignment, with $\mathsf{RID} = a_i$ supplied inside the encrypted registration payload, and since accounts are unique on $\mathcal{L}$, the sender's $\mathsf{RID} = a_r$ addresses exactly r's list. Finally, $pos = tx_hash$ identifies the transaction receipt containing HS', which any provider can serve and whose inclusion the recipient can verify against the receipt trie (Figure 1).

Proof sketch for Proposition 2

The signal $(\mathsf{RID} = a_r, pos)$ reaches the TEE encrypted under pk_{TEE}, and the host's view of signal processing and retrieval consists of attested code execution and ORAM accesses to DB, $\mathsf{DB}_{\mathsf{recip}}$ and $\mathsf{DB}_{\mathsf{keys}}$, which the security proof of [9, Sec. 6] renders independent of RID and pos. The only recipient-side value that leaves the SPS path is pos itself, later queried to an ordinary read provider. This binds ν_r to the public sender address a_s of that transaction, not to a_r, and joining this observation with the sender's signal submission would require the collusion excluded by assumption above.

Event log content security

In the following, every classical DH public value published on $\mathcal{L}$ is treated as known to a future quantum adversary, since Shor's algorithm [18] recovers the corresponding secret from harvested events. Each claim below therefore reduces to identifying, per event type, the secret input that never appears on-chain in quantum-recoverable form.

a) HS-C1: In Construction 1, the HS event carries the ML-KEM ciphertext $\mathsf{ct}_{\mathsf{kem}}^{\mathsf{env}}$ and the AEAD-encrypted body $\mathsf{ct}^{\mathsf{env}}$. Since K_{env} comes from a single-use ML-KEM encapsulation against the recipient's long-term post-quantum key $pk_r^{\mathsf{KEM\text{-}lt}}$, a harvesting adversary cannot recover the body, which contains the ephemeral key exchange material $\mathsf{msg}_{\mathsf{eph}}$. The long-term ML-KEM key protects only the handshake envelope; the session keys are derived exclusively from the ephemeral material inside it, so $\mathsf{ct}^{\mathsf{env}}$ yields no useful plaintext under HNDL assumptions.

b) HS-C2: In Construction 2, $\mathsf{HS}' = \mathsf{HS} \setminus \{\rho_r\}$ removes only the recipient selector, so the initiator's ephemeral classical public key pk_s^{cls} is exposed on-chain and its DH contribution ss^{dh} is quantum-recoverable. The session key $K = \mathsf{KDF}(ss^{\mathsf{dh}} \| ss^{\mathsf{kem}})$ nevertheless remains hidden, since ss^{kem} is protected by ML-KEM.

c) HSR: The HSR event exposes in cleartext the response tag τ and the responder's one-time tag public key R. The tag is computed via the hybrid derivation of Algorithm 14, which takes ss^{kem} as input key material and the classical DH output involving R only as salt: a quantum adversary recovers the salt, but ss^{kem} remains computationally hidden, so τ can be neither reproduced nor used to extract session material.

d) MSG: Each MSG event exposes in cleartext the ratcheted topic θ_i and the sender's current ratchet DH public key, alongside the encrypted message payload. We consider two adversarial scenarios. (i) A quantum adversary harvests MSG events without having compromised the local ratchet state: the DH contribution of every ratchet step is then quantum-recoverable, but advancing the ratchet also requires the current root key RK_i, which is local client state, never published on-chain, and ultimately anchored in the hybrid secret σ_{hyb} of the handshake. FS is thus preserved even against a quantum adversary. (ii) A classical adversary has obtained a transient snapshot of one party's ratchet state, including its current ratchet DH secret key. A ratchet step by the non-compromised party alone does not heal the session, since its fresh public key is on-chain and the stolen secret completes the DH. PCS is instead satisfied once the compromised endpoint contributes its next ratchet key. Since neither secret of that DH is known to the adversary, the resulting root key RK_{i+1} lies outside its reach.

Algorithms for the Proposed Solutions

Algorithm 8 $\mathsf{Register}(a_u)$

Require: Account a_u
Ensure: $\mathcal{C}_{\mathsf{reg}}$ updated with $(a_u, pk_u^{\mathsf{KEM\text{-}lt}})$; $sk_u^{\mathsf{KEM\text{-}lt}}$
1: $(sk_u^{\mathsf{KEM\text{-}lt}}, pk_u^{\mathsf{KEM\text{-}lt}}) \leftarrow \mathsf{KEM}_{\mathsf{pqc}}.\mathsf{KeyGen}()$
2: $\phi_u \leftarrow \mathsf{PoP}(sk_u^{\mathsf{KEM\text{-}lt}}, pk_u^{\mathsf{KEM\text{-}lt}}, a_u)$
3: submit transaction $\mathcal{C}_{\mathsf{reg}}.\mathsf{register}(pk_u^{\mathsf{KEM\text{-}lt}}, \phi_u)$ from a_u
4: **store** locally $sk_u^{\mathsf{KEM\text{-}lt}}$

Algorithm 9 $\mathsf{SendHS}(s, r)$

Require: Sender s with identity a_s; recipient address a_r
Ensure: HS emitted on $\mathcal{C}$; initiator state state_s
1: $pk_r^{\mathsf{KEM\text{-}lt}} \leftarrow \mathcal{C}_{\mathsf{reg}}.\mathsf{lookup}(a_r)$
2: $(\mathsf{msg}_{\mathsf{eph}}, \mathsf{state}_s) \leftarrow \mathsf{HybKE.Init}()$ ▷ see Algorithm 5
3: $(\mathsf{ct}_{\mathsf{kem}}^{\mathsf{env}}, K_{\mathsf{env}}) \leftarrow \mathsf{KEM}_{\mathsf{pqc}}.\mathsf{Encaps}(pk_r^{\mathsf{KEM\text{-}lt}})$
4: $\sigma_s \leftarrow \mathsf{Sig.Sign}(sk_s^{\mathsf{SIG}},\ a_s \,\|\, a_r \,\|\, \mathsf{msg}_{\mathsf{eph}})$
5: $\mathsf{body} \leftarrow (\mathsf{msg}_{\mathsf{eph}}, pk_s^{\mathsf{SIG}}, \pi_s, \sigma_s)$
6: $\mathsf{ad} \leftarrow a_s \,\|\, a_r$
7: $\mathsf{ct}^{\mathsf{env}} \leftarrow \mathsf{AEAD.Enc}(K_{\mathsf{env}}, \mathsf{body}, \mathsf{ad})$
8: **erase** K_{env}
9: emit $\mathsf{HS} = (a_s, \mathsf{ct}_{\mathsf{kem}}^{\mathsf{env}}, \mathsf{ct}^{\mathsf{env}})$ on $\mathcal{C}$ ▷ no indexed recipient field
10: **return** state_s

Algorithm 10 $\mathsf{ScanHS}(r, \Delta)$

Require: Recipient r; block range Δ
Ensure: Set inbox of handshakes addressed to r
1: $Q \leftarrow P^r.\mathsf{getLogs}(\mathcal{C},\ \mathsf{type} = \mathsf{HS},\ \Delta)$
2: $\mathsf{inbox} \leftarrow \emptyset$
3: **for** each $(a_s, \mathsf{ct}_{\mathsf{kem}}^{\mathsf{env}}, \mathsf{ct}^{\mathsf{env}}) \in Q$ **do**
4: $K_{\mathsf{env}} \leftarrow \mathsf{KEM}_{\mathsf{pqc}}.\mathsf{Decaps}(sk_r^{\mathsf{KEM\text{-}lt}}, \mathsf{ct}_{\mathsf{kem}}^{\mathsf{env}})$
5: $\mathsf{ad} \leftarrow a_s \,\|\, a_r$
6: $\mathsf{body} \leftarrow \mathsf{AEAD.Dec}(K_{\mathsf{env}}, \mathsf{ct}^{\mathsf{env}}, \mathsf{ad})$
7: **if** $\mathsf{body} \neq \perp$ **and** $\mathsf{Sig.Verify}(\mathsf{body}.pk_s^{\mathsf{SIG}},\ \mathsf{body}.\sigma_s,\ \cdot)$ **and** $\mathsf{IDVerify}(a_s,\ \mathsf{body}.\pi_s)$ **then**
8: $\mathsf{inbox} \leftarrow \mathsf{inbox} \cup \{(a_s, \mathsf{body}.\mathsf{msg}_{\mathsf{eph}})\}$
9: **end if**
10: **end for**
11: **return** inbox

Algorithm 11 $\mathsf{V2Setup}(R_i)$

Require: Security parameter 1^λ, upper bounds N, M, $\bar{l}$, recipient R_i with address a_i
Ensure: SPS infrastructure initialized, R_i registered with $\mathsf{RID} = a_i$
SPS infrastructure, run once:
1: $\mathsf{SPS.Setup}(1^\lambda, N, M, \bar{l})$
Recipient registration, run by each R_i:
2: $\mathsf{SPS.Register}(pk_i, sk_i;\ \mathsf{RID} = a_i)$

Algorithm 12 $\mathsf{V2SendHS}(s,\ r)$

Require: Sender s with address a_s, recipient r with address a_r
Ensure: HS′ published and signal submitted to SPS infrastructure
1: $\mathsf{HS}' \leftarrow \mathsf{HS} \setminus \{\rho_r\}$
2: $(tx, pos) \leftarrow$ submit transaction $\mathcal{C}.\mathsf{sendHS}(\mathsf{HS}')$ from a_s
3: $\mathsf{SPS.Send}(\mathsf{RID} = a_r,\ loc = pos)$

Algorithm 13 V2ScanHS(r)

Require: Recipient r registered as $\mathsf{RID} = a_r$ with SPS keys (pk_r, sk_r)
Ensure: All valid pending HS' events retrieved from $\mathcal{C}$

$\vec{P} \leftarrow \mathsf{SPS.Receive}(\mathsf{RID} = a_r,\ pk_r,\ sk_r)$
$\mathsf{inbox} \leftarrow \emptyset$
for each $pos \in \vec{P}$ **do**
  if pos is a valid ledger position **then**
    $\mathsf{receipt} \leftarrow P^{\mathsf{r}}.\mathsf{getTransactionReceipt}(pos)$
    add every HS' event in receipt to inbox
  end if
end for
return inbox

Algorithm 14 RespondHS(r, $\mathsf{msg}_{\mathsf{eph}}$)

Require: $\mathsf{msg}_{\mathsf{eph}} = (pk_s^{\mathsf{cls}}, pk_s^{\mathsf{kem}})$ recovered from an inbound HS, identity proof π_r
Ensure: HSR emitted on $\mathcal{C}$, responder bootstrap state

$(sk^{\mathsf{tag}},\ R) \leftarrow \mathsf{DH}_{\mathsf{cls}}.\mathsf{KeyGen}()$ ▷ one-time tag keypair
$(\mathsf{msg}_B,\ \sigma_{\mathsf{hyb}}) \leftarrow \mathsf{HybKE.Respond}(\mathsf{msg}_{\mathsf{eph}})$ ▷ retain ss^{kem} and sk_r^{cls}
$\tau \leftarrow \mathsf{Kec}\big(\mathsf{HKDF}(ss^{\mathsf{kem}},\ \mathsf{DH}_{\mathsf{cls}}.\mathsf{ComputeKey}(sk^{\mathsf{tag}}, pk_s^{\mathsf{cls}}),\ \mathtt{hsr})\big)$
$\mathsf{body} \leftarrow (\mathsf{msg}_B,\ pk_r^{\mathsf{SIG}},\ \pi_r)$
$\mathsf{ct} \leftarrow \mathsf{Enc}(pk_s^{\mathsf{cls}},\ \mathsf{body})$
erase sk^{tag}
emit $\mathsf{HSR} = (\tau,\ a_r,\ R,\ \mathsf{ct})$ on $\mathcal{C}$
return $(\sigma_{\mathsf{hyb}},\ sk_r^{\mathsf{cls}},\ \tau)$

Algorithm 15 MatchHSR(s, Δ)

$Q \leftarrow P^{\mathsf{r}}.\mathsf{getLogs}(\mathcal{C},\ \mathsf{type} = \mathsf{HSR},\ \Delta)$ ▷ no indexed filter
for each $(\tau,\ a_r,\ R,\ \mathsf{ct}) \in Q$ and each $(a_p, \mathsf{state}_p) \in \mathcal{P}$ **do**
  $\mathsf{body} \leftarrow \mathsf{Dec}(sk_p^{\mathsf{cls}},\ \mathsf{ct})$
  if $\mathsf{body} \neq \perp$ **then**
    $ss^{\mathsf{kem}} \leftarrow \mathsf{KEM}_{\mathsf{pqc}}.\mathsf{Decaps}(sk_p^{\mathsf{kem}},\ \mathsf{body}.ct^{\mathsf{kem}})$
    $ss^{\mathsf{dh}} \leftarrow \mathsf{DH}_{\mathsf{cls}}.\mathsf{ComputeKey}(sk_p^{\mathsf{cls}},\ R)$
    $\tau' \leftarrow \mathsf{Kec}\big(\mathsf{HKDF}(ss^{\mathsf{kem}},\ ss^{\mathsf{dh}},\ \mathtt{hsr})\big)$
    if $\tau' = \tau$ **and** $\mathsf{IDVerify}(a_r,\ \mathsf{body}.\pi_r)$ **then**
      $\sigma_{\mathsf{hyb}} \leftarrow \mathsf{HybKE.Finish}(\mathsf{state}_p,\ \mathsf{body}.\mathsf{msg}_B)$
      bootstrap session from $(\sigma_{\mathsf{hyb}}, \tau)$; $\mathcal{P} \leftarrow \mathcal{P} \setminus \{(a_p, \mathsf{state}_p)\}$
    end if
  end if
end for

REFERENCES

[1] M. Galligan, P. Anchuri, S. Carlin, J. Hasegawa, M. Kobetic, N. Molnar, and J. Turner-Baggs, "The XMTP litepaper," XMTP Labs, Tech. Rep., Jul. 2025. [Online]. Available: https://github.com/xmtp/litepaper

[2] f. CassOnMars (maintainer, "Fip: Direct casts #99," https://github.com/farcasterxyz/protocol/discussions/99, Jun. 2023, gitHub Discussion.

[3] H. Cornelius, S. Tikhomirov, A. Revuelta, S. P. Vivier, and A. Challani, "The waku network as infrastructure for dapps," in *2024 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, Jul. 2024, pp. 31–32. [Online]. Available: https://ieeexplore.ieee.org/document/10646404

[4] K. Jefferys, M. Shishmarev, and S. Harman, "Session: End-to-end encrypted conversations with minimal metadata leakage," https://arxiv.org/abs/2002.04609, Feb. 2024.

[5] J. Van Den Hooff, D. Lazar, M. Zaharia, and N. Zeldovich, "Vuvuzela: Scalable private messaging resistant to traffic analysis," in *Proceedings of the 25th Symposium on Operating Systems Principles*, 2015, pp. 137–152.

[6] S. Angel and S. Setty, "Unobservable communication over fully untrusted infrastructure," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 551–569.

[7] R. Recabarren and B. Carbunar, "Tithonus: A bitcoin based censorship resilient system," *Proceedings on Privacy Enhancing Technologies*, 2019. [Online]. Available: https://doi.org/10.2478/popets-2019-0005

[8] M. Minaei, P. Moreno-Sanchez, and A. Kate, "Moneymorph: Censorship resistant rendezvous using permissionless cryptocurrencies," *Proceedings on Privacy Enhancing Technologies*, vol. 2020, no. 3, 2020.

[9] S. Jakkamsetti, Z. Liu, and V. Madathil, "Scalable private signaling," in *2025 IEEE 38th Computer Security Foundations Symposium (CSF)*, 2025, pp. 143–158.

[10] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," May 2009. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[11] V. Buterin, "A next generation smart contract & decentralized application platform," 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:19568665

[12] N. Szabo, "Formalizing and securing relationships on public networks," *First Monday*, vol. 2, 1997. [Online]. Available: https://api.semanticscholar.org/CorpusID:33773111

[13] X. Li, B. Zhao, G. Yang, T. Xiang, J. Weng, and R. H. Deng, "A survey of secure computation using trusted execution environments," *arXiv preprint arXiv:2302.12150*, 2023.

[14] V. Madathil, A. Scafuro, I. A. Seres, O. Shlomovits, and D. Varlakov, "Private signaling," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 3309–3326.

[15] National Institute of Standards and Technology, "Module-lattice-based key-encapsulation mechanism standard," NIST, Tech. Rep. FIPS 203, 2024, https://doi.org/10.6028/NIST.FIPS.203.

[16] P. Grubbs, V. Maram, and K. G. Paterson, "Anonymous, robust post-quantum public key encryption," in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2022, pp. 402–432.

[17] V. Maram and K. Xagawa, "Post-quantum anonymity of kyber," in *IACR International Conference on Public-Key Cryptography*. Springer, 2023, pp. 3–35.

[18] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 1994, pp. 124–134.