SoK: Metadata-Protecting Communication Systems

Sajin Sasy
University of Waterloo
Waterloo, ON, Canada
ssasy@uwaterloo.ca

Ian Goldberg
University of Waterloo
Waterloo, ON, Canada
iang@uwaterloo.ca

ABSTRACT

Protecting metadata of communications has been an area of active research since the dining cryptographers problem was introduced by David Chaum in 1988. The Snowden revelations from 2013 resparked research in this direction. Consequently over the last decade we have witnessed a flurry of novel systems designed to protect metadata of users' communications online. However, such systems leverage different assumptions and design choices to achieve their goal; resulting in a scattered view of the desirable properties, potential vulnerabilities, and limitations of existing metadata-protecting communication systems (MPCS).

In this work we survey 31 systems targeting metadata-protected communications, and present a unified view of the current state of affairs. We provide two different taxonomies for existing MPCS, first into four different categories by the precise type of metadata protections they offer, and next into six families based on the core techniques that underlie them. By contrasting these systems we identify potential vulnerabilities, as well as subtle privacy implications of design choices of existing MPCS. Furthermore, we identify promising avenues for future research for MPCS, and desirable properties that merit more attention.

KEYWORDS

privacy, metadata privacy, anonymous communications

1 INTRODUCTION

Individuals from opposite ends of the world communicate seamlessly over the Internet using several choices of popular messengers like Signal, Whatsapp, or Telegram. We innately expect such conversations we have over the Internet to have the same privacy as us conversing with our correspondent in person in a private space. The Snowden revelations from 2013 [68], however, revealed that this expectation was fallacious, resulting in a widespread call to arms within the security and privacy community to proactively resist mass surveillance [1, 60]. In practice, this eventually culminated with the adoption of end-to-end encryption as a standard for the vast majority of communications over the Internet [35, 66, 75]. End-to-end encryption effectively hides the contents of any communication between two parties from network adversaries. Nonetheless, by virtue of how these communications tools operate, they inherently leak the *existence* of a conversation and the *metadata* of who conversed with whom, when, and how much did they converse. Unfortunately, in the era of global surveillance that we live in, this metadata might be sufficient for terrible consequences [3, 52].

Free and democratic societies depend on an informed public, which depends on whistleblowers shedding light on misdeeds and corruption [13, 68]. Imagine the plight of a whistleblower, faced with the daunting task of publishing confidential documents that would expose acts of incompetence, fraud, or inhumane practices carried out by their government or employer that thrives on deceiving the public. Any action in such a direction often results in incarceration or risks to their lives [9, 74]. The same is true of individuals in oppressive regimes whose lives are endangered by their mere sexual orientation or political stance [26, 59]. Protecting individuals' communications requires a communication system with strong security and privacy guarantees for both data and metadata.

Related Work. In order to tackle the problem of metadata leakage, various forms of anonymous communication networks (ACNs) with varying degrees of metadata protections and design goals have been proposed. In practice users are limited to a few choices like Tor [27] or I2P [78]. These systems are designed to enable their users to protect their communications from a local eavesdropping network adversary like the user's Internet Service Provider (ISP). These systems have low latencies, but they are innately susceptible to traffic analysis attacks by a global passive network adversary. Moreover, recent attacks have surfaced that make a Tor user's Internet usage susceptible to privacy breaches by even local network adversaries [37, 54, 73]. Cwtch [43] internally leverages Tor's v3 onion services protocol to facilitate peer-to-peer communications, and consequently inherits the privacy limitations of the Tor network, and similarly I2P's threat model too is limited to local adversaries.

Academically, peer-to-peer (P2P) based anonymous communication networks with varying privacy properties have been designed and studied [48, 50, 55, 58]. These systems do not provide the desired metadata-protecting guarantees since they too are limited in their privacy guarantees in the face of a strong global adversary. Furthermore, several attacks have been demonstrated against such P2P designs [49, 62, 67, 72]. Most of such early literature focuses on either designing P2P-based anonymous communication networks that circumvents some of Tor's shortcomings (such as its ability to scale, and the central points introduced in its design), or on improving DC-net and mixnet based constructions to be more efficient. Earlier surveys on anonymous communication [25, 65] well cover these aforementioned approaches; these surveys focus on the routing-level challenges of anonymous communication networks. However over the last few years there has been a paradigm shift. Several recent metadata-protecting communication schemes use radically different approaches than those covered in prior surveys, to the extent that prior routing hurdles encountered do not apply anymore.

In this work we investigate 31 systems that aim to facilitate metadata-protecting communications even in the presence of global network adversaries. To contrast existing works and extract meaningful insights of the status quo, we provide two broad classifications of these works. Our contributions are:

(1) We classify existing works into four different categories by the type of metadata privacy guarantees they offer, and

into six different families based on the core techniques that underpin these works. We then leverage these classifications to identify tradeoffs induced on such systems by their privacy guarantees, as well as the inherent limitations and benefits of the different families of constructions.
(2) Existing works often target different goals. Consequently, they aspire to different properties and defend against different forms of attacks, resulting in a scattered view of the desirable properties and potential vulnerabilities of such systems. In this work, we contrast existing systems under a unified view of these different facets.
(3) This unified view helps us identify potential vulnerabilities of existing works, as well as identify promising avenues for future research of metadata-protecting communication systems. Additionally, we also highlight subtle but impactful differences in the privacy implications of systems that leverage the same underlying assumption.
(4) Deployable metadata-protecting messaging systems have been the aspirational goal of this line of research. We consider this application and highlight factors that hinder such systems today.

Paper Layout: We first detail our methodology to gather all the systems we survey in this work in Section 2. In Section 3, we establish what we mean by a Metadata-Protecting Communication System (MPCS), and categorize MPCS based on their target functionality and the metadata privacy guarantees they offer. We outline the properties various MPCS aim to provide in Section 4. We then provide a high-level overview of these existing MPCS in Section 5, divided into six families based on the core techniques used; here we present works within a family together, incrementally exhibiting the contribution of each system over its predecessor. We then succinctly systematize all these systems in Table 1, partitioned by the metadata-privacy guarantee they support. In Section 6, we discuss the tradeoffs introduced by the different privacy goals, as well as the tradeoffs inherent to the underlying families. Based on these discussions, in Section 7 we identify promising directions for future research for MPCS, and conclude in Section 8.

2 METHODOLOGY

To gather all the works relevant to this SoK, we started with Express [30] as the seed paper. We then gathered all the works that were cited by Express, and those that cite Express using Google Scholar during November 2022. Among them, works that introduce a new system for metadata-protecting communications (as defined below) were selected while others where filtered out. This same process was then repeated transitively for all selected papers, to finally arrive at the 31 works. Each of these works details a novel approach towards metadata-protecting communications.

3 METADATA-PROTECTING COMMUNICATION SYSTEMS (MPCS)

In this work, we define a Metadata-Protecting Communication System (MPCS) to be any communication system designed with its threat model accounting for a global network adversary observing all communications on the network, and protecting (at least one form of) metadata of its users' communications. This definition of an MPCS is intentionally broad, and allows us to capture several different forms of MPCS designs. We focus on the global adversary threat model as that is the real-world adversary to combat today in the context of MPCS. We include systems that are secure even when a client's correspondent is adversarial and those that are not [6]. In systematizing existing MPCS, we identify two dimensions that help categorize existing MPCS, namely functionality and privacy goals. We first list the different types of functionality and privacy goals the systems we surveyed target.

Functionality. Existing MPCS are designed to attain either *anonymous broadcast* or *end-to-end messaging (E2E)*. Anonymous broadcast protects the sender of a broadcasted message from network adversaries. End-to-end MPCS protect the conversation relationships of who is communicating with whom on the network, and possibly even the existence of a conversation.

Privacy Goals. Existing works overload the terms 'private' or 'anonymous' with different meanings. Kuhn et al. [36] provide a taxonomy of the myriad privacy notions attainable in a communication network. We use their terminology to categorize privacy goals of existing systems into four, namely: (i) *Sender-Message Unlinkability* ($(SM)\bar{L}$: *S*ender-*M*essage Un$\bar{L}$inkability) and (ii) *Receiver-Message Unlinkability* ($(RM)\bar{L}$) which aim to hide the metadata of who sent or received some message in a system, iii) *Relationship Unobservability* ($M\bar{O}[M\bar{L}]$ short for $M\bar{O}(SM\bar{L}, RM\bar{L})$: *M*essage Un$\bar{O}$bservability with *S*ender-*M*essage Un$\bar{L}$inkability and *R*eceiver-*M*essage Un$\bar{L}$inkability) aims to hide the metadata of conversation relationships of users in the system, and iv) *Communication Unobservability* ($C\bar{O}$) hides the metadata of even the existence of a conversation.

Contrasting the 31 systems we surveyed fairly poses a challenge as they have different privacy and functionality goals. Hence we classify MPCS into four different categories based on their combined privacy and functionality goals,[1] such that systems within a category can be fairly compared. For example, comparing a broadcast scheme against an end-to-end scheme offers no insights. Similarly systems that align in functionality but differ in their privacy goals cannot be fairly compared either. The four categories are:

(1) **Sender Unlinkable Messaging Systems (SUMS)** [$(SM)\bar{L}$, E2E]: These systems protect the sender and the message they submit. Typically, clients have a dedicated 'mailbox', and the sender's message is delivered to its destination mailbox without leaking which mailbox it was.
(2) **Sender Unlinkable Broadcast Systems (SUBS)** [$(SM)\bar{L}$, Broadcast]: These are systems designed for metadata-protected broadcast, and typically operate over epochs. Messages are gathered during the course of an epoch. At the end of an epoch, the messages are published to a public bulletin board while hiding the mapping between senders and messages.
(3) **Relationship Unobservable Systems (RUS)** [$M\bar{O}[M\bar{L}]$, E2E]: In a RUS, the global adversary can infer the set of senders and recipients, as well as the number of ongoing conversations, but they cannot identify any sender-receiver pair from this pool of active correspondents. We identify

[1] While there are eight permutations of functionality and privacy goals, only the four pairs we detail (plus one more that only appears in the RUS subdesigns) are logically sound. For instance, broadcast does not pair with relationship unobservability as there are no relationships to be observed for a broadcast.

three broad designs for a RUS that start with a system with a weaker privacy goal, and then boost it to a RUS by enforcing communication epochs where first each client in the system sends a message followed by all clients retrieving a message.

(a) **Mix Systems (Mix)** [$(SM)\bar{L}$, $(RM)\bar{L}$, E2E]: In these systems all incoming messages are mixed to break any sender-message correlations, before they arrive at their destinatio. From an adversary's perspective, the senders and receivers that participate in a round of communication are observable, but the sender-receiver pairs remain hidden.

(b) **Sender Unlinkable Messaging Systems (SUMS)** [$(SM)\bar{L}$, E2E]: SUMS (discussed earlier) can be converted into a RUS with the above generic transformation.

(c) **Receiver Unlinkable Messaging Systems (RUMS)** [$(RM)\bar{L}$, E2E]: In these systems, the sender and the (encrypted) message they submit in a round are linkable. However recipients retrieve messages without revealing which messages in this round were addressed to them.

(4) **Communication Unobservable Systems (CUS)** [$C\bar{O}$]: These systems correspond to the highest level of metadata privacy attainable by Kuhn et al.'s taxonomy, as the global adversary is unaware of even the existence of a conversation in such systems. Typically CUS designs attain this high level of privacy by enforcing rigid requirements on clients. Existing CUS designs require all clients participating in such a system to be online at all times and perform predetermined conversation actions (like sending and receiving dummy messages) even if they are not currently in a conversation.

4 MPCS PROPERTIES

Informed by the surveyed works, we present several system-level properties for MPCS, and later we use these properties to contrast existing schemes. We partition these properties into three categories, namely *Protections*, *Usability*, and *Performance*.

4.1 Protections

(1) **(Server) Robust**: In this work we consider the property of server robustness as the ability of an MPCS to function correctly while tolerating server churn.[2]

(2) **Anonymity Set (AS) Protection**: The anonymity set of honest clients participating in the system must not be reducible by malicious servers. This property ensures that an adversarial server cannot perform deanonymization attacks by controlling the anonymity set. Being able to manipulate the inclusion or exclusion of clients using an MPCS can place adversarial servers at a strong vantage point. For instance, in the extreme case an adversarial server could perform an $n-1$ attack [63] to deanonymize a user and their message in a sender-anonymous broadcast system. This ability to manipulate participation can also be used by servers for targeted blocking of clients, as captured by the notion of censorship resistance [2], or the 'audit attack' [51].

(3) **DoS Resistance**: There are broadly two types of DoS attacks that MPCS aim to protect against, namely:

(a) **Resource Exhaustion (RE)**: Like traditional DoS attacks, clients can attempt to overwhelm servers through Sybil messages, precluding honest clients from communicating.

(b) **Disruption**: In some synchronous systems a client can potentially submit malformed messages that disrupt a communication round for all users. Alternatively in asynchronous systems, clients can craft malformed messages that disrupt the logical database that stores all client messages.

(4) **Disconnection Impact**: Several MPCS impose liveness assumptions on its users; i.e., they require the set of clients be the same throughout the operation of the system (until a reset at which point new clients can be enrolled), and that these clients never go offline. Alternatively, some constructions do not impose a liveness assumption, but participants incur privacy penalties when users go offline during a round of communication. We identify four possible outcomes for user disconnections (and present the symbols we later use to denote them): (i) system fails altogether (⃠), (ii) correspondent leaked (♉), (iii) anonymity set reduction (▼), and (iv) *temporary* anonymity set preservation (≈), i.e., even when a client disconnects for a small window of time they can still appear to send and receive messages by offloading this work to an untrusted server on their behalf.

4.2 Usability

(1) **Setup**: The MPCS we cover do not require clients to have any IT knowledge or host servers to support metadata-protected communications, however some systems do have other prerequisites of clients. Namely, some systems require (i) clients exchange information (public keys, pseudorandom addresses, etc.) out of band (🤝), or (ii) out-of-band exchange, plus clients must participate in a dialing protocol (see Section 4.4) before they can send actual conversation data (📞).

(2) **Parallel Conversations**: The ability of an E2E MPCS to support multiple simultaneous conversations for a user.

(3) **Asynchronous**: Users in an asynchronous system can send messages to others even if the recipients are not currently online; such messages can then be retrieved the next time the recipients connect to the system.

(4) **Low Latency**: To use an MPCS for messaging, it needs to have an *acceptable messaging latency*; i.e., when a user sends a message it should be delivered to their correspondent (or published) within a reasonable time frame. We set this acceptable latency to 10 s for a system serving more than 10^5 clients, about two orders of magnitude greater than messaging latencies enjoyed by users today over standard non-metadata-protecting messengers (supporting billions of users). It typically takes few seconds for a user to type a message, so the idea of a 10 s acceptable latency is that sending a message should not take too much longer than creating one for a seamless messaging experience.

4.3 Performance

(1) **Horizontal Scalability**: The ability of a system to scale as more users join by adding more servers to the system. In the context of MPCS, the system needs to be able to scale while

[2]Within our definition of robustness, we do not include fairness or guaranteed output notions of robustness from the secure multi-party computation (SMC) literature.

maintaining metadata protections without partitioning the anonymity set. Later, we will quantify the multiplicative factor of servers required to scale horizontally such that the work done per server remains the same when the system's user base grows by a factor of k.

(2) **Client Overhead**: The asymptotic computational or communication overheads induced by the MPCS on its clients.

(3) **Messages and Message/Compute Complexity**: The number of messages the MPCS can process in a logical round, and the corresponding message (or compute) complexity induced on its servers.

4.4 Dialing and Conversation Protocols

Several existing designs for E2E MPCS divide communication into two subprotocols, namely dialing and conversation protocols. Dialing protocols enable an online user Alice to express their desire to communicate with another user Bob. If Bob is online, and reciprocates this interest, such protocols result in Alice and Bob establishing a shared secret that they can leverage to facilitate their metadata-protected conversation. The dialing protocol itself must also be designed to hide metadata.

Once the shared secret is established by a dialing protocol, Alice and Bob converse using a conversation protocol. Conversation protocols typically require the correspondents to have a pre-established shared secret. Different systems from the literature suggest different frequencies of dialing to conversation rounds in the network; this ratio is typically system- and application-dependent.

The requirement of a dialing protocol is an artifact of designs that assume a synchronous communication model between users. In such works a pair of users can enjoy metadata-protected conversations only when they are both online and actively conversing, but users cannot send messages to an offline user for later retrieval. In several cases [29, 40, 42, 69, 71] it even hurts the privacy guarantees of a user, if their correspondent goes offline during a conversation round or vice versa, as we detail later in Section 7.2. If a metadata-protecting communication network is asynchronous by design then it removes the requirement of a dialing protocol to check if their participant is currently online, as we see later in Section 5.4.

5 EXISTING WORK

In this section, we provide a high-level overview of the 31 different MPCS that we survey in this work. We broadly categorize these metadata-protecting communications systems into six different families based on the core cryptographic technique that underpins their security and privacy properties, namely DC-nets, mixnets, Private Information Retrieval (PIR), Reverse PIR, differential privacy (DP), and secure multi-party computation (SMC). For ease of exposition we discuss the various system designs by family here, but later in Table 1 we present these systems partitioned by the category of metadata privacy guarantees they offer.

5.1 DC-net Based Systems

The *dining cryptographers problem* introduced by David Chaum [14] poses the question of designing communication networks ingrained with sender and recipient unobservability, and introduced **DC-nets** (1988) as a solution. This original DC-net solution envisions a network with all clients connected to each other without any server involvement in the protocol; we hence call this an all-client DC-net. Participants communicate lockstep in a series of rounds, with each round split into as many slots as there are participants. Only one participant sends a real message in a given slot of a round, while the rest of the participants provide "cover" ciphertexts carefully crafted such that when any participant XORs all the ciphertexts received in a given slot, they end up with the message of that slot. While DC-nets guarantee perfect sender and recipient unobservability, they induce significant communication overheads to do so, as well as have several constraints that make them impractical to deploy.

For instance, DC-nets are prone to Denial-of-Service or disruptions, when more than one participant attempts to send a real message in a round. Disruptions are only feasible in certain designs of metadata-protecting schemes. In particular DC-net based and reverse PIR based schemes (that we discuss later in Section 5.5) are susceptible to such attacks, as these underlying techniques enable clients to influence the contents of other messages in the system. **Dissent** [22] (2010) addresses the disruption problem by scheduling slots for its participants while preserving the anonymity of senders. They do so by introducing an accountable group shuffle protocol, the output of the shuffle determines the slot ordering of the round.[3]

Another fundamental problem with DC-nets is their inability to scale as they require an all-to-all communication network among all participating clients. To address this, Wolinsky et al. [77] leverage the *anytrust* model in proposing **D3** (2012), which converts the classical DC-net from a fully peer-to-peer network to a client-server model with n clients and m servers. In the anytrust model the clients need only trust that at least one server from the set of m servers behaves honestly.[4] The privacy guarantees are upheld even if the honest server is not the client's immediate upstream server. The anytrust model reduces the overheads imposed on a client, as the client outsources the bulk of its cover ciphertext generation to its upstream server. **Dissent in Numbers** [76] (2012) (Dissent v2) leverages the anytrust assumption from D3 to enable the original Dissent protocol to support more clients.

While these systems that leverage the anytrust assumption are efficient, they are still susceptible to disruptions, where malicious clients deviate from the standard protocol and transmit arbitrary ciphertexts, which prevents revealing the plaintext for that round. Hence, D3 and Dissent v2 additionally introduce a retroactive blame protocol to detect disruptors in a group. However, they cannot prevent a disruption from happening in the first place. **Verdict** [23] (2013) attempts to prevent disruptions by introducing verifiable DC-nets. Clients in Verdict submit a proof of correctness to prove their contribution is well formed. This defence incurs significant overheads of verifying correctness proofs of all client messages.

5.2 Mixnet Based Systems

Riffle [39] (2016) uses the anytrust assumption as well, but instead of the servers facilitating a DC-net, they act as a mixnet. The servers form an ordered chain, and each server performs a verifiable shuffle

[3] In Dissent, the disruption resistance only applies to the shuffle protocol, but not the communication protocol itself; i.e., clients misbehaving in the shuffle protocol can be detected and removed but Dissent cannot protect against clients disrupting the following communication rounds.

[4] This is the same trust model as Chaum's mix networks [15].

on the set of messages it receives in a given round and passes it on to the next server; hence as long as one of the servers in this mixnet is honest no information about the ordering of messages is revealed to any malicious nodes after the honest node in the chain. At the end of a round the last server publishes all the shuffled messages thus providing sender anonymity, hence making Riffle a SUBS.

cMix [16] (2016) proposes a fixed cascade of mix servers like Riffle with the same trust assumption. The key innovation in cMix is the separation of all the (expensive) public key operations into a precomputation phase, which allows for lighter computation during the online phase of the protocol. cMix can serve as a SUBS or a RUS as these shuffled messages can then be published, or forwarded to their intended recipient if they have a recipient identifier.

Leibowitz et al. introduced the Anonymous Post-Office Protocol, **AnonPOP** (2016), a synchronous mix-cascade network with Post Office (PO) and mix servers. In AnonPOP, each client has a mailbox that is maintained by one of the PO servers. Messages in AnonPOP are onion routed with authenticated encryption through a set of random mix servers. To retrieve messages, clients send a pull request (which contains a proof of ownership of the mailbox) to their mailbox PO through a mixchain, and the PO responds back with the first message in the client's mailbox. Their main contributions are techniques for bad-server isolation, and the notions of request pool and per-epoch mailboxes that enable limited disconnection of a client, without complete loss of their sender and recipient anonymity. However, the latter two mechanisms, while efficient defences, provide only heuristic security.[5]

Kwon et al.'s **Atom** [38] (2017) is the first system in this line of work that has the explicit design goal of horizontal scalability. Atom is an anonymous broadcast protocol designed for 'short latency-tolerant' messages, enabling anonymous microblogging. The high-level idea of Atom is to have its servers arranged as a random permutation network and shuffle clients' messages as they pass through the network. Atom uses the square network by Håstad [34], which can permute m elements using $\sqrt{m}$ nodes with each shuffling $\sqrt{m}$ ciphertexts and connects to $\sqrt{m}$ nodes in the subsequent layer.[6]

In Atom each logical server in this network is realized by a group of physical servers such that the anytrust assumption holds over the group. The number of servers in each group is tuned to ensure that given the total number of servers, and knowledge that up to a bounded fraction of them can be malicious, at least one server in each group should be honest with very high probability. To send a message, a user picks an entry group and encrypts their message to the public keys of all servers in that group, and then sends it to all servers in that entry group. Once enough ciphertexts are available to an entry group, each server in a group shuffles the ciphertexts and passes them on to the next server. The last server after shuffling divides the permuted ciphertexts into $\sqrt{m}$ batches, and passes all the batches back to the first server. Each server then leverages the out-of-order reencryption property of El Gamal to partially decrypt their portion from each batch and reencrypt the ciphertexts to the next group that this batch is destined for. The last group in the network simply makes the plaintexts available, completing the anonymous broadcast functionality.

In order to protect against malicious servers that could tamper with client messages in Atom, Kwon et al. propose two techniques i) using non-interactive zero knowledge (NIZK) proofs and ii) using client-inserted trap messages (which turns out to be significantly faster than using NIZK proofs in practice, but at the expense of doubling the number of messages in the system). Finally, in order to prevent the system from being blocked due to server churn they also propose a fault-tolerant version of their protocol by leveraging 'manytrust' groups, which are anytrust groups with at least h honest servers in each group, such that each group can tolerate up to h failures by replacing the group's public key with the key of a threshold cryptosystem.

XRD [40] (2019) (short for Crossroads) further reduced the overheads of shuffling in such mixnet based protocols. XRD aims to support relationship-unobservable communications and hence is by design a RUS, unlike the other mixnets schemes discussed so far that are SUBS that can be extended to a RUS. Each client in XRD has a unique mailbox associated with them. Servers in the system are organized into mix chains, and in each round each user sends a message along ℓ chains. The system is designed such that all pairs of users have at least one chain in common, and the choice of chains themselves are publicly computable; in order for this to operate correctly given q chains, clients have to choose $\ell \approx \sqrt{q}$ chains to ensure that they have an intersecting chain with every other user. The high-level idea is that users not communicating send ℓ 'loopback messages' in a round which return to their mailbox, while if two users Alice and Bob are communicating in a given round they replace their common chain with messages for each other instead of the loopback message. This results in an adversary always observing each active user's mailbox receiving ℓ messages in each round, and they cannot determine whether any two individuals are actively communicating or not.

The two immediate downsides that arise from this network design based on chain intersections is that clients effectively have to send $O(\sqrt{n})$ messages for n users in the system, and given m servers each server also consequently has to process $O(n/\sqrt{m})$ messages as opposed to $O(n/m)$ by other horizontally scalable designs. Additionally, XRD does not provide (or recommend) a dialing protocol for their system, and none of the currently existing dialing protocols are appropriate for XRD. However, XRD introduces a novel technique 'aggregate hybrid shuffle' (AHS) to circumvent having to perform expensive verifiable shuffles, and hence has lower latencies for each round of communication.

Loopix [56] (2017) takes a significantly different approach. Other than mix servers and users, Loopix includes special service provider nodes. The service provider nodes act as entry and exit points of the network (bookending the mix servers), and clients interface with the Loopix system through these nodes. The network itself consists of mix nodes that are organized in a stratified network topology with shallow depth, with each node connected to all the other nodes from its next and previous layer.[7] The high-level idea is that users send each other messages by routing their messages through this network with source-picked delays from a Poisson distribution

[5] These defences are fragile. They assume honesty of a particular server (as opposed to any one server of a pool like anytrust), and the defence fails if that server is adversarial. This leads us to use the term heuristic security.

[6] This network only requires constant iterations of mixing to produce a near-uniform random permutation, and in their work they use 10 iterations of mixing.

[7] In its experiments, Loopix uses a network depth of three.

at each of the mix nodes. Mixes and users also generate cover traffic using a Poisson distribution to thwart passive and active attacks; consequently the entire system can be viewed as a large Poisson mixing process, providing it with formal guarantees on its message delivery and privacy. The service provider plays a semi-trusted role and maintains clients' mailboxes. If a client is offline, their service provider stores the messages destined for that client, enabling clients to retrieve messages even after they go offline, making it the only mixnet design with support for asynchrony.

Unfortunately, Loopix relies on trusting its clients' service provider nodes. A malicious service provider node can violate receiver anonymity, as they can observe receivers' interactions with their mailbox, and can infer metadata about the messages received. Finally, while Loopix implicitly leverages the manytrust assumption, its design has concerning privacy implications when contrasted with other systems that leverage the same assumption, as we detail later in Section 6.3.1.

Trellis [41] (2022) introduces an anonymous broadcast system that is robust to server churn. Their work, however, assumes a synchronous communication network and user liveness (i.e., all users are online for all rounds of communication). Similar in flavour to cMix, the high-level idea of Trellis is to lift all the heavyweight public key cryptography operations into a one-time path establishment phase, followed by lightweight message transmission rounds.

In Trellis, Langowski et al. introduce two novel cryptographic building blocks, namely Anonymous Routing Tokens (ART) and Boomerang Encryption (BE). Similar to Atom, Trellis leverages a permutation network to instantiate the mix network. ART enables a user to generate a publicly verifiable random next server for routing their message through the permutation network. Given a chain of servers to onion encrypt a message for (dictated by the ART at each layer), BE onion encrypts a message in the reverse path (of the onion encryption chain), and then onion encrypts this in the forward direction. The resulting ciphertext serves as a proof of message delivery in onion routing schemes, when the message in the reverse direction is set to be a random nonce, as this guarantees that every server along the path decrypted onion layers correctly and forwarded it to the next server. The interplay of ART and BE enables honest servers to blame deviant servers if messages for an ART it is responsible for is missing or faulty, during both path establishment as well as message broadcast rounds. In order to recover from such faults, similar to Atom, Trellis leverages proactive secret sharing to distribute the secret key of each server in the permutation network across an anytrust group, thus enabling a substitute server to recover the state of a faulty (or malicious) server, and resume operations in the event of a blame protocol.

5.3 Differential Privacy Based Systems

These systems start with a mixnet design and further enhance them by leveraging differential privacy (DP) [28]. However, using DP implies their privacy guarantees are significantly different from those of mixnets (as we detail later in Section 6.1), so we treat them as their own family.

Vuvuzela [71] (2015) introduced this approach of leveraging DP to noise the adversary-observable variables of a communication network. This allows them to claim via DP that any set of observations made given a user's real action is almost as likely as what would be observed given any other plausible cover story. They apply this DP principle to build a metadata-protected messaging system using two protocols: i) a point-to-point conversation protocol, and ii) a dialing protocol to initiate the conversation.

Vuvuzela consists of a single chain of servers under an anytrust assumption, which clients communicate through. In Vuvuzela, users communicate through virtual locations called *dead drops*. Two conversing users send their messages to the same dead drop. Conversing clients are assumed to know each other's long-term public key a priori, and the dead drop location is computed for a given round via a shared secret derived from their public keys and the current round number. Clients send onion-encrypted messages through the server chain, to have their message delivered to their desired dead drop at the last server in the chain. In order to achieve privacy, each server in the chain after peeling its layer of encryption shuffles all the messages it receives and forwards it to the next. After all the messages have arrived at the dead drop (if two arrive at the same dead drop, they get exchanged), the messages make their way back through the system in reverse order with each server reversing the shuffle they performed.

In order to hide the true number of conversations happening in the system, all users irrespective of if they are actually conversing or not send a message. Users not participating in a conversation send messages to a random dead drop. Additionally each server in the chain also injects two forms of cover traffic: single-message and pair-message dead drop cover traffic, which hides the true count of conversing and non-conversing users in the system.

Since Vuvuzela is throttled in scalability by the number of messages each server has to process as the number of users in the system grow, **Stadium** [69] (2017) was designed with the goal of horizontal scalability. All servers add noise messages collaboratively at the start of a round into the system, and throughout the duration of the round they get processed along with users' messages by verifiable processing techniques. In Stadium, the system comprises two types of mix chains, input chains and output chains, with the same anytrust assumption held true for each individual chain. Two users conversing with each other choose a random input chain, but the same output chain as their messages are destined for the same dead drop. In order to reduce the overheads of the verifiable shuffle they extend the hybrid verifiable shuffling techniques from Riffle [39], and separate verifiable shuffling of large messages onto shuffle proofs over smaller AEAD keys.

The deviation from Vuvuzela's design then arises from Stadium's phase of distributing messages from input to output chains. In order to prevent malicious servers from tampering with users' messages, servers in an input chain collaborate to compute a hash of the messages intended for each output chain and sends the hash to every server of that output chain. If every server in the input chain acts honestly, these hashes will be equivalent, and the distribution phase would have been correctly executed.

Karaoke [42] (2018) further refines the horizontal scalability by introducing two key techniques, namely efficient noise verification and optimistic indistinguishability. In Karaoke, Lazar et al. point out that instead of performing verifiable operations, it is sufficient to verify that all noise messages inserted into the system still exist in the system at the end of round. Towards this end they propose

using Bloom filters [10]; each server at each layer in Karaoke's topology computes a Bloom filter of all the messages it has received and sends this to all other servers. The other servers then verify whether the noise messages they generated appear in the Bloom filter. If any server indicates that their noise has been lost, the round is stopped. This circumvents the overheads of verifiable shuffle in exchange for efficient hash operations.

Instead of each user sending and receiving one message like in Vuvuzela and Stadium, Lazar et al. observe that if users send and receive two messages in a round they can appear to be in a conversation. In this setting, when the adversary is passive or there are no network outages, the number of dead drop accesses reveals no metadata about the communication of any pair of users, since for a pair of users either idling or chatting, there will be two dead drops both of which are accessed twice. If messages are lost, however, the adversary may observe a dead drop with a single access, which reveals some information. Karaoke addresses this through the addition of noise messages, as in the prior systems. Interestingly, message loss is detectable by the user in Karaoke since they can see the messages received back from the server. Hence, in Karaoke, Lazar et al. make the distinction of "leakage-free" rounds for rounds where no message loss is incurred. Karaoke's leakage-free rounds allow them to improve performance by tuning the amount of noise messages injected depending on whether the system is currently experiencing message loss or not.

Barman et al. introduced **Groove** [8] (2022) which further refines DP-based systems with three main user flexibility options. The first is oblivious delegations, which enables a user to leverage an untrusted service provider to participate in the typically rigid mechanisms of a CUS; i.e., users can preemptively provide servers with their messages for some number of future rounds of communications, which enables these service providers to receive messages from the users' correspondents while they go offline. Upon reconnection, users can retrieve just the messages of interest by a shuffle-and-select protocol that hides which of the messages were retrieved. Second, Groove enables users to converse with multiple correspondents (a system constant of 50) in parallel, and finally it enables multi-device usage for clients so they can use Groove over multiple devices, without any privacy leakages that may arise from lack of synchronization between these devices.

5.4 Private-Read Based Systems

We classify PIR-based systems into private-read and private-write based systems. Private-read systems leverage PIR in the standard fashion. Private-write systems use 'PIR in reverse' [21] instead, and hence we call this technique Reverse PIR (RPIR). The high-level operation of a PIR-based MPCS is that arriving messages are placed into a data structure, and receivers use PIR to privately read that data structure without revealing which messages were read.

Pynchon Gate [61] (2005) is one of the earliest works that used PIR to design private communications; their work proposes a practical pseudonymous mail retrieval system. Their system consists of three types of nodes: a nym server, a collator, and distributors. The nym server is the public-facing end of Pynchon Gate that sends and receives pseudonymous email. On its own the nym server does not provide sender anonymity but clients can leverage mix networks [15] for sender anonymity. The nym server collects mails for all the pseudonyms and at the end of every 'cycle' passes these messages to the collator. The collator organizes the mails into a structure that allows clients to query for their mails. The collator then distributes this structure across the distributor nodes, and the clients use Chor's ℓ-server PIR protocol [19] to retrieve their mail.

Pung [7] (2016) proposed the first PIR-based MPCS and indeed the first MPCS at all since the original (disruption-prone) all-clients DC-net that does not rely on an assumption about non-colluding servers. They treat the problem as a metadata-protected key-value store, and enable clients to privately retrieve messages from this key-value store by leveraging the computational PIR (CPIR) scheme, XPIR [47]. In each round, clients send and retrieve exactly one message. The key for each message is a random label generated by a shared secret (established via sharing public keys through an out-of-band channel), and the value is the encrypted message. At the end of the round, the Pung server organizes the received messages into a binary search tree, and clients perform an oblivious search by labels to retrieve messages of interest to them.

Angel et al. [5] improved the efficiency of CPIR schemes with **SealPIR** (2018), by i) reducing the network bandwidth overheads of CPIR schemes by a novel query compression technique which reduces the query size from $O(n)$ ciphertexts to $O(1)$ for an n-record PIR database and ii) a data-encoding scheme for multi-query PIR schemes that allows it to amortize the cost of processing a batch of requests from a single client. They port these techniques to Pung to further reduce the network costs and throughput of Pung.

Talek [17] (2020) introduces a private *group messaging* system that aims to provide privacy for its users via access sequence indistinguishability. Talek's core abstraction is a private log, that enables a single writer to share messages with many readers. In order to convert this to a group messaging protocol, each member subscribes to the logs of all other group members.

The log itself is hosted by ℓ non-colluding ITPIR servers, and leverages a blocked cuckoo hash table [53] to improve efficiency. In Talek, clients issue writes to pseudorandom locations on each server, where the locations are determined by applying a PRF to a secret log handle that is shared between the log's writer and readers. The cuckoo hash table is parameterized to have b buckets, each with d messages in them, which dictates the maximum capacity of messages n in the system to be $b \times d$. When new messages enter the system after this maximum capacity, the oldest entries are evicted from the bucket they get assigned to. Hence, unlike other metadata-protecting communication systems Talek has to be parameterized first to set this value of n, b, and d to trade off the time-to-live (TTL) for each message in the system with the latency of each PIR request.

Their other core contribution is a private notification system via Bloom filters [10], which allow users to check which of their subscribed logs have updates with significantly lower overheads than performing a PIR request. This allows users to poll their subscribed logs periodically and perform PIR requests efficiently.

5.5 Private-Write Based Systems

In contrast to systems from the previous section, instead of leveraging PIR to retrieve a message from a collection of messages privately,

RPIR systems use PIR to *insert* a message into a collection privately. Corrigan-Gibbs et al. introduced this idea with **Riposte** [21] (2015), a novel design for anonymous broadcast, allowing clients to anonymously post messages to a shared 'bulletin board', maintained by a small set of non-colluding servers. Riposte effectively implements a write-private and disruption-resistant database scheme.

In Riposte's efficient variant, two of the three servers act as ITPIR servers that the client issues requests to, while the third is an 'audit' server that participates in an MPC protocol to ensure clients submit well-formed requests. The system operates in epochs, and at the end of every epoch the Riposte servers publish all the write requests they received during an epoch, providing each honest client an anonymity set of honest clients that participated in that epoch. The two servers together hold a logical database, and each client aims to anonymously drop messages into a slot in this database.

Since clients drop messages into a random slot via private writes, collisions could render messages unreadable. Corrigan-Gibbs et al. address this issue in two ways: i) tuning the size of the database table to be large enough to accommodate the expected number of write requests for a given success rate, and ii) introducing a novel technique to recover from collisions at the expense of storage. Given the expected number of messages in an epoch, these two techniques can be tuned to set a database size that imposes the least overheads for the PIR scheme. Finally, in order to improve the bandwidth efficiency of the PIR scheme, they propose using Distributed Point Functions [32] (DPFs) to reduce the asymptotic overhead of client requests from $O(L)$ to $O(\sqrt{L})$, where L is the number of entries in the database (set by the tuning described earlier).

Express [30] (2021) details a metadata-protecting messaging system that only incurs constant-factor overheads on the client side regardless of the number of users. Express is a natural extension of Riposte, and similar to Riposte leverages a non-collusion assumption among the two servers of the system. Unlike Riposte, in Express all participating recipients have their own individual mailbox. Hence Express does not operate in epochs, since messages can only be read by their recipient, making it the first asynchronous metadata-protecting communication system that leverages PIR.

The key developments in this work are the utilization of a more efficient DPF scheme[12], and using Secret-shared Non-Interactive Proofs (SNIP) [11, 20] instead of the more expensive zero knowledge proof that was used in Riposte for auditing the client-submitted DPFs. The former reduces their client-side overhead to logarithmic in the number of mailboxes, instead of square root. However, in addition they propose using a virtual address space of 2^λ for mailboxes, and senders need to know their recipient's virtual and physical mailbox address in order to send messages, making the client overhead $O(\lambda)$. This virtual address trick enables Express to prevent targeted disruption attacks.

More recently, Vadapalli et al.'s **Sabre** [70] (2022) illustrated how both Riposte and Express are susceptible to resource exhaustion (RE) attacks. Although they both defend against ill-formed DPFs, an RE attacker can still submit well-formed DPFs that write to a non-existent mailbox resulting in the server having to perform expensive $O(n)$ operations for an effective no operation, yielding a DoS attack with an exponential amplification factor as the client work is $O(\lambda \cdot \log n)$ lightweight operations. In the process of defending against this form of resource exhaustion DoS attack, Sabre also succeeds at fundamentally speeding up the message delivery protocol.

The improvement arises from Vadapalli et al.'s observation that mailbox address verification (i.e., verifying that a client is writing a message to a valid mailbox in the system) can be performed separately from message delivery. This allows, for n users, the use of smaller $O(\log n)$ instead of $O(\lambda)$ sized DPF keys that Express uses for message delivery (this is better since in practice $\log n \ll \lambda$ typically). However, one still needs to perform mailbox address verification, and to this end Vadapalli et al. provide two novel protocols, the more efficient one of which is an $O(\lambda)$ protocol. Only if that check passes, the audit protocol then ensures the well-formedness of the DPF, for which Sabre has another innovation enabling servers to only perform $O(\lambda \cdot \log n)$ work, thus removing the DoS amplification factor, unlike Express where the server has to perform $O(\lambda \cdot n)$ work. In concrete performance their experiments show that their optimizations result in a significant multiplicative factor improvement under "good" write conditions (i.e., no malicious DoS attacker writes), with this performance gap widening up to one to two orders of magnitude as the ratio of bad to good writes increases.

Newman et al.'s **Spectrum** [51] (2021) solves a closely related yet slightly different problem of high-bandwidth anonymous broadcast. Their setting considers a small number of broadcasters that wish to share documents or large files with many subscribers by leveraging two or more non-colluding servers. Unlike the previous systems discussed so far, by limiting the number of broadcasters to L out of n participants (broadcasters + subscribers) they succeed at reducing the server-side computations to $O(L)$ per message write, while still providing these L broadcasters an anonymity set of all n participants. The trick lies in allocating one 'channel' (synonymous to mailboxes from Riposte and Express) to each of these L broadcasters, with the broadcaster having a credential that allows them to publish to this channel. The messages to the channels leverage the same DPF-based technique discussed for Riposte and Express, with the only difference being the share of one's credential being sent along with the DPF key share to each server. The subscribers too can submit message writes which are intended for any of the channels, however these writes are strictly zero entries.

Additionally, Newman et al. also point out that Riposte, Express, (and Sabre) are susceptible to malicious servers selectively dropping valid client messages to perform selective deanonymization attacks. To address this they design BlameGame, an audit protocol that helps the honest servers detect if any of the other servers are deviating from their protocol, by having clients send their message shares encrypted under a verifiable encryption scheme to each server and having all servers commit to the message share they used, allowing the servers to implicate a malicious client or server.

5.6 Secure Multiparty Computation (SMC) Based Systems

MCMix [4] (2017) recasts the problem of metadata-protected communication as a SMC consisting of dialing and conversation functionalities. The system operates in rounds; at the end of each conversation round conversing users have the messages they sent into the system swapped. To do so, users interested in conversing establish their intent via the dialing protocol, which allows them to

establish a dead drop location for the next (possibly several) communication rounds. This dead drop location is a 64-bit address that can be generated by both participants by a hash function via their shared secret and the current conversation round. Once all messages for a communication round have been gathered, the servers perform an SMC protocol that obliviously sorts all the messages by the dead drop addresses and swaps adjacent messages that are destined for the same address. This oblivious sort is in fact the crux and bottleneck of their protocol, and they leverage the "shuffle before sort" paradigm from Hamada et al. [33], which results in an overarching $O(n \log n)$ complexity for both their conversation and dialing protocols when there are n participants.

Clarion [29] (2021) designs an improved three-party malicious secure shuffle protocol, and shows how it can serve as an anonymous broadcast channel by having the servers publish the secret-shared shuffled messages at the end of the shuffle. Clarion details a simple mechanism to use such an anonymous broadcast channel as a drop-in replacement for the conversation protocol in MCMix. Instead of having conversation participants send messages to the same dead drop address, they now send messages addressed with two unique dead drop locations derived from the same shared secret, one for each direction of the conversation. At the end of the broadcast, each client retrieves the location their correspondent would have written to, and clients that are not in an active conversation retrieve the same random location they wrote to. Since every client still always sends and retrieves a single message identified by a random address tag, the behaviour of clients having a conversation remains indistinguishable from the other clients. Clarion's improvement stems from the fact that the new secret-shared shuffle protocols are $O(n)$ protocols, saving a $\log n$ factor over the oblivious sort underlying MCMix.

Lu et al. present **Asynchromix** and **PowerMix** [45] (2019). They propose techniques to provide robustness guarantees to SMC protocols, without incurring the additional overheads of malicious-secure SMC protocols. They aim to support fairness and guaranteed output, in addition to the notion of robustness we consider in Section 4. The high-level idea for both designs is to have each server in the system obtain a share of each client's message, mix the message shares, and reconstruct the messages once the mixing has disassociated the clients and their messages. To receive valid message shares from clients without the overheads induced by Verifiable Secret Sharing [57], Lu et al. propose clients blind their message before broadcasting it to the servers. The blinding value is provided by the servers; each server provides a share of a random blinding value to the client, and clients reconstruct and blind their messages with it. Once the client broadcasts their blinded message, the servers reproduce shares of the original message by removing their share of the blind from it. Asynchromix [45] has $O(\log^2 n)$ round complexity as it leverages $O(\log n)$ rounds of iterated butterfly networks (which have depth $O(\log n)$ towards mixing the messages. This results in a protocol with $O(mn \log^2 n)$ communication and computation costs. Alternatively, PowerMix [45] only has 2 rounds of communication, but incurs significantly more computation. PowerMix leverages Newton's sums instead of a shuffle network. In order to mix a batch of n messages, the servers compute the powers of each share of the message from 1 to n and then compute the sums of each power. This can be computed with a single round of communication by using precomputed powers of random secret-shared values. The servers then reconstruct the sums of each power publicly, and then solve for the set of messages.

Blinder [2] (2020) leverages SMC towards enabling robust anonymous broadcast. Blinder aims to limit the ability of malicious servers to block honest clients from submitting their messages, and refers to this property as censorship resistance, effectively preventing the system from reducing the anonymity set offered by honest clients. Blinder is based on Shamir secret sharing [64], and leverages the recent observation of an efficient 'sum of products' gate [18]. Blinder reuses the server-side storage layout and collision handling techniques from Riposte. Clients compress their queries in a fashion similar to recursive PIR [19] to reduce the communication overheads, and the servers perform a format verification on the compressed query. Censorship resistance is achieved via a robust input sharing protocol that prevents malicious servers from discarding (more than a small number of) honest client messages.

RPM [44] (2022) proposes three schemes for anonymous broadcast. Their idea is to generate a permutation matrix to shuffle the client messages in a round. Each server picks its own permutation matrix, and secret shares it with the rest. All servers multiply these shared matrices to get the final permutation matrix. The first scheme requires an inner product SMC computation in the online phase, and the second removes this by shifting it into an offline phase. Both variants require constructing and multiplying n-by-n matrices for n client messages in the offline phase, which is prohibitive for large n. The third variant repurposes the first two schemes as building blocks, by using them as the nodes of Håstad's permutation network [34], resulting in offline creation and multiplication of a constant number of permutation matrices of size $\sqrt{n}$-by-$\sqrt{n}$. The first two variants are limited in scalability as they incur complexity cubic in the number of clients, while the third variant brings down the online computation complexity to $O(n^{1.5})$.

6 DISCUSSION

We succinctly summarize all the works we detailed in Section 5 into Table 1, partitioned by the type of metadata privacy guarantee these systems achieve, and highlighting the properties from Section 4. The table also contrasts each scheme's asymptotic communication and computation complexities, and any dialing or offline computation overheads induced as well. We refrain from comparing concrete performance numbers here as each proposed system uses their own experimental setup for evaluating their prototype implementations. Appendix A provides details on the performance numbers reported by these works for a better sense on practicality of these schemes, which is not immediate from the underlying complexities.

6.1 Tradeoffs Between Families

DC-nets are the oldest line of work towards metadata-protecting communications, and can easily be adapted for use towards E2E or broadcast applications. Interestingly, DC-nets innately achieve a CUS by design. Unlike the other systems that attain a CUS by padding up with cover traffic as we discuss later in Section 6.2.4, the notion of cover traffic in these systems is implicit in the design.

System	Year	Type	Family	Assumption	Robust	AS Protection	RE Resistant	Disruption Resistant	Disconnection Impact	Setup	Parallel Conversations Asynchronous	Low Latency	Horizontally Scalable	Audit Complexity Client Overhead	Message [Compute if different] Complexity	Messages
					Protections					**Usability**				**Performance**		
Vuvuzela [71]	2015	CUS (Mix)	DP	$(1:m)$-S: DP	○	○	●	●	▼	📞	○	○	○	×	m	$n \cdot m + n \cdot m$	n
Pung [7]	2016	CUS (RUMS)	PIR	RLWE	○	○	●	●	▼	📞	●	◐	●	k^2	×	n	$n^2 + n$	1
AnonPOP [31]	2016	CUS (Mix)	M	f-S: DP	○	○	●	●	≈	🤝	●	○	○	k	×	ℓ	$n \cdot \ell$	n
Stadium [69]	2017	CUS (Mix)	DP	f-S: DP	○	○	●	●	▼	📞	○	○	○	k	×	ℓ	$(n + m_d) \cdot m_d + n \cdot \ell + m^3$	n
MCMix [4]	2017	CUS	SMC	$(1:3)$-S	○	○	●	●	▼	📞	○	○	○	○	×	1	$n \cdot \log n + n \cdot \log n$	n
SealPIR [5]	2018	CUS (RUMS)	PIR	RLWE	○	○	●	●	▼	📞	●	◐	●	k^2	×	n	$n^2 + 1$ $[n + n]$	1
Karaoke [42]	2018	CUS (Mix)	DP	f-S: DP	○	○	●	●	▼	📞	○	○	●	k	×	ℓ	$(n + m_d) \cdot m_d + n \cdot \ell + m^3$	n
XRD [40]	2019	CUS	M	f-S	○	○	●	●	▼	📞	◐	○	○	k^2	×	$\sqrt{m}$	$n^2 + n \cdot \sqrt{m}$	1
Clarion [29]	2021	CUS	SMC	$(2:3)$-MA	○	○	●	●	☿	📞	○	○	○	○	×	1	$n \cdot \log n + n$	n
Groove [8]	2022	CUS (Mix)	DP	f-S: DP	○	○	●	●	≈	📞	●	◐	●	k	×	ℓ	$(n + m_d) \cdot m_d + n \cdot \ell + m^3$	n
DC-nets [14]	1988	CUS/SUBS	DC	None	○	●	●	○	⊘	🤝	○	○	○	○	×	n	$n^2[n]$	1
Dissent [22]	2010	CUS/SUBS	DC	None	○	○	●	○	⊘	🤝	○	○	○	○	n^3	n	$n^2 + n^2$ $[n^2 + n]$	1
D3/Dissentv2 [76, 77]	2012	CUS/SUBS	DC	$(1:m)$-S	○	○	●	◐	▼	🤝	○	○	○	○	$n^2 \cdot m$	m	$n \cdot m + n \cdot m$	1
Verdict [23]	2013	CUS/SUBS	DC	$(1:m)$-S	○	○	●	●	▼	🤝	○	○	○	○	$n^2 \cdot m$	m	$n \cdot m + n \cdot m$	1
Riposte [21]	2015	SUBS	RPIR	$(2:3)$-S	○	○	○	●	▼	×	⊘	●	●	k^2	n	$\sqrt{n}$	$\sqrt{n}$ $[n]$	1
Riffle [39]	2016	SUBS	M	$(1:m)$-S	○	○	●	●	▼	×	⊘	●	○	○	×	m	$n \cdot m$	n
cMix [16]	2016	SUBS	M	$(1:m)$-S	○	○	●	●	▼	×	○	○	○	○	×	m	$n \cdot m$	n
Atom [38]	2017	SUBS	M	f-S	●	◐	●	●	▼	×	⊘	●	○	k	×	1	$n \cdot m$	n
AsynchroMix [45]	2019	SUBS	SMC	$(2n/3:n)$-M	◐	●	●	●	▼	×	⊘	●	○	○	×	m	$m \cdot n^2 \cdot \log^2 n + m \cdot n \cdot \log^2 n$	n
PowerMix [45]	2019	SUBS	SMC	$(2n/3:n)$-M	◐	●	●	●	▼	×	⊘	●	○	○	×	m	$n^2 + m \cdot n$ $[n^2 + n^3 + m]$	n
Blinder [2]	2020	SUBS	SMC	$(3n/4:n)$-M	●	●	●	●	▼	×	⊘	●	○	○	$m \cdot \sqrt{n}$	$m \cdot \sqrt{n}$ $m \cdot n \cdot \log n + m \cdot n$ $[m \cdot n \cdot \log n + n^2]$	n
Spectrum [51]	2021	SUBS	RPIR	$(1:m)$-S	○	●	●	●	▼	×	⊘	●	○	○	1	m	m $[m \cdot b]$	1
Trellis [41]	2022	SUBS	M	f-M	●	●	●	●	⊘	×	⊘	○	○	k	×	$\log m$	$n \cdot \log^2 m + n \cdot \log m$ $[n \cdot m]$	n
RPM [44]	2022	SUBS	SMC	$(3n/4:n)$-M	●	●	●	●	▼	×	⊘	●	○	○	×	m	$m \cdot n^{1.5} + n$ $[m \cdot n^2 + n^{1.5}]$	n
Sabre [70]	2022	SUMS/SUBS	RPIR	$(1:2)$-S	○	○	●	●	▼	🤝	●	●	●	k^2	$\log n$	$\log n$	1 $[n]$	1
Express [30]	2021	SUMS	RPIR	$(1:2)$-S	○	○	○	●	▼	🤝	●	●	●	k^2	n	λ	1 $[n]$	1
Mixnets [15]	2003	RUS (Mix)	M	$(1:m)$-S	○	○	●	●	⊘	×	○	○	○	○	×	m	$n \cdot m$	n
Pynchon Gate [61]	2005	RUS (RUMS)	PIR	$(1:m)$-S: TTP	○	○	●	●	▼	×	●	●	○	○	×	$n \cdot m$	$n \cdot m$	1
Loopix [56]	2017	RUS (Mix)	M	f-S: TTP	○	○	●	●	▼	🤝	●	◐	◐	k	×	1	ℓ	1
Talek [17]	2020	RUS (RUMS)	PIR	$(1:m)$-S	○	○	●	●	▼	🤝	●	◐	○	○	×	m	$n \cdot m$	1

Table 1: (I) ***Notation:*** **(i)** ***Family*****: DC = DC-nets, M = Mixnets, RPIR = Reverse-PIR, DP = Differential Privacy, SMC = Secure Multiparty Computation. (ii)** ***Variables*****: n=#clients, m=#servers, m_d=#servers in dialing protocol, f=fraction of malicious servers, ℓ=#servers on an anytrust chain (ℓ=$g(f, m)$ where g depends on the system design to ensure that at least one server in every anytrust chain is honest), b=#broadcasters. (iii)** ***Assumptions*****: TTP: Trusted Third Party. RLWE: Ring Learning With Errors. (x, y)-T: At least x of y servers should be honest, with servers of type T: S - Semihonest, M - Malicious, MA - Malicious but Available. f-T: At most fraction f of m servers can be malicious, with servers of type T. (iv)** ***Disconnection Impact*****: ⊘: system fails, ☿: correspondent leaked, ▼: anonymity set reduction, ≈: temporary anonymity set preservation. (v)** ***Setup*****: 🤝: clients exchange information out of band, 📞: 🤝 + dialing protocol. (vi)** ***Performance*****: Dialing or shuffling (for slot scheduling) costs in blue. Preprocessing/offline costs in gray.**

(II) ***Meanings of ◐:*** **(i) Robust: non-robust offline phase. (ii) Disruption resistant: disruption resistant scheduling, but not conversations. (iii) Asynchronous: The RUS underlying Pung and SealPIR is asynchronous, but as a CUS it is not asynchronous. Groove allows users to disconnect for a small constant number of rounds. Loopix's asynchrony has a privacy expense (Section 5.2). Talek asynchronous messages can be lost by its design (Section 6.5).**

(III) The blue-dashed box highlights the three crucial properties required for a metadata-protecting messaging system as we discuss in Section 6.4.

The primary weakness of DC-nets are its susceptibility to disruptions, and the works we detail in Section 5.1 aim to address that. Furthermore, DC-nets are inherently synchronous, and have poor scalability in spite of three decades of research scrutiny.

Meanwhile mixnets have shown incredible promise over the years. While earlier designs like Riffle and cMix lacked the ability to scale, more recent designs like Atom, Loopix, and Trellis provide horizontally scalable constructions. Moreover, some of the recent constructions like Atom and Trellis are even robust. The main shortcomings of this family are its inability to support asynchronous messaging. While Loopix supports asynchrony, it does so at the expense of a stronger trust assumption, namely trusting the service provider node that clients attach to.

The DP-based family can scale horizontally as well as support significantly lower latency than most prior metadata-protecting communication systems. However, this is achieved at the expense of shifting the underlying privacy guarantee; namely, these works provide differentially private metadata-protections rather than cryptographic ones. This shift in privacy guarantee has two significant drawbacks. First, DP guarantees are based on the probability gap between two possible scenarios (the user sending a real message or a dummy message), and users gain their metadata protections by being able to provide a 'plausible cover story' to deny their actual action. However, this gap may be sufficient for certain adversaries to act upon. Second, the degree of privacy attained by a user is dependent on the number of rounds in which they participate. The more rounds a user participates in, the more their privacy degrades. Eventually with enough number of rounds they exhaust the allocated privacy budget with which the system was designed, at which point the privacy guarantees are unclear.

On the other hand, the PIR-based (private read) and RPIR-based (private write) families to their merit can support asynchrony, but have poor horizontal scalability and no support for robustness. Their design and experiments are limited to the 2 or 3 server settings, and as the number of users in these system increases they are computation bound by each of these servers. Designs from the SMC-based family lack the ability to scale horizontally. However, recent works from the SMC-based family [2, 44, 45] address anonymity set protection, a property that has received hardly any attention.

6.2 Tradeoffs Within Categories

6.2.1 SUBS. Existing SUBS constructions stem from all but DP and PIR families as observed in Table 1; the merits of DP and PIR simply do not align with the goal of SUBS and hence this is unsurprising. Amongst existing SUBS, RPIR based constructions (in particular Sabre) outperform other works by orders of magnitude in terms of concrete latency as detailed in Appendix A, but unfortunately do not support robustness, anonymity set protections, or horizontal scalability. Fortunately, recent SUBS constructions from the SMC family (like Blinder) have focused on robustness and anonymity set protection, and achieve these goal albeit at the cost of performance. Unfortunately, almost all existing SUBS as seen from Table 1 fall short of being able to scale horizontally, except the mixnet designs Atom and Trellis. However, both these designs achieve horizontal scalability at the expense of significant latency overheads in Atom,[8] and in Trellis at the expense of a rigid immutable set of users who are assumed to be online at all times.

6.2.2 CUS. As evidenced by the table, almost all CUS designs from the literature require either a shuffle (DC-net schemes) or a dialing protocol from its participants prior to any actual communication, which lower bounds the latencies they can achieve.[9] Furthermore, the only horizontally scalable constructions for CUS arise from the DP family, with the exception of XRD, a mixnet CUS design that is horizontally scalable. However, XRD has weaker scalability than that of the DP family as described earlier in Section 5.2. The other immediate observation from the table is the lack of asynchronous designs for CUS. However, this lack of asynchrony is in fact a conscious design choice for CUS for two reasons: i) it is easier to analyze metadata protections under a synchronous communication assumption, and ii) having all users synchronously communicate in all rounds of communication circumvents statistical inference and intersection attacks [24, 46] that prior ACNs fall prey to.

6.2.3 RUS. As we mention in Section 3, there are three broad directions one can employ to achieve a relationship unobservable system, namely SUMS, RUMS, and mixnets. A RUS based on mixnets is synchronous and in fact provides the exact metadata privacy purported by the definition of a RUS, as messages from all senders during a round get mixed before they are distributed to the intended recipients; this allows an adversary to infer the exact set of senders and recipients in a round of communication. This is permissible by definition of a RUS; recall that the goal is to hide the pairwise sender-recipient relationships.

On the other hand, SUMS and RUMS constructions leverage PIR (in the form of PIR-writes and PIR-reads respectively), and hence innately support asynchrony. In SUMS, senders send their message into one mailbox amongst the pool of all possible registered mailboxes within the system. These systems do not have a notion of communication rounds; a recipient checks their mailbox periodically to retrieve any messages they may have received.[10] Hence this style of RUS innately benefits from a larger anonymity set (of all possible clients registered in the system) for the recipients, instead of just the true recipient set as in mixnets.

While RUMS has a notion of rounds, recipients in RUMS retrieve messages from the pool of all submitted messages within a round in a fashion that hides the message that they retrieved. Provided all clients in the system attempt to retrieve a message in every round (irrespective of whether they have real messages to retrieve or not), like SUMS, RUMS systems too can support a larger anonymity set for its recipients. This suggests that realizing a RUS via SUMS or RUMS in fact provides better metadata privacy guarantees than the definition of a RUS, since the recipient anonymity set is always the set of all possible clients (that have mailboxes) in the system, and is agnostic of the true recipient set for any batch of messages.

[8] For instance, with 2 million users transmitting a tweet-length message takes about 30 minutes.

[9] The exception is AnonPOP, for which each client has a designated mailbox as we detailed in Section 5.2.

[10] The guidance provided by existing works consequently is that recipients do so in a predictable fashion, agnostic of the underlying real communication pattern.

6.2.4 RUS + Cover Traffic = CUS. CUS provides the highest level of metadata protections feasible in a communication network. However, attaining this in practice is non-trivial, and as mentioned earlier typically requires that all participants of the system send and receive messages at all times so as to essentially provide cover traffic for any actual ongoing conversations. RUS on the other hand can be viewed as a relaxation of CUS without the stringent requirement of all users being online (and pretending to converse) at all times, at the expense of weaker metadata-privacy guarantees and susceptibility to intersection attacks. The gap between these two categories can be bridged by some trivial impositions on the clients of the system, and boils down to having sufficient cover traffic at all times. Any RUS design can trivially be converted into a CUS by requiring that all clients in the system send and receive a (dummy, if they do not have a real) message within an allocated time frame. This is in fact the case with several existing works [4, 5, 7, 29, 40], for which we mention the underlying RUS type in Table 1. We note that the two partially asynchronous statuses in the table, for Pung and SealPIR, stem from the fact that the underlying RUS that is converted into a CUS does support asynchrony.

6.3 Limitations of Security Assumptions

Amongst the pool of assumptions that have been leveraged towards constructing MPCS, traditional all-clients DC-nets assume the least, as the protocol requires no servers (nor any server assumptions consequently); while clients could be malicious and could DoS the system, they cannot subvert metadata privacy. However, scaling DC-nets requires moving to the anytrust assumption. Assumptions that are based on splitting trust across multiple entities, either anytrust, or non-collusion amongst servers, are hard to realize in practice and moreover dangerous as these trust assumptions can be silently subverted, endangering the privacy of all users of such systems. This makes systems designed on top of CPIR (with assumptions like RLWE) significantly more desirable in practice from a privacy standpoint. Unfortunately such systems are computationally expensive and consequently incur prohibitively high latencies for message delivery; additionally, currently proposed schemes from these families lack horizontal scalability.

6.3.1 The Many Manytrusts. There is a silent but concerning difference in the metadata privacy guaranties attained by the different manytrust systems we detailed in Section 5. While on the surface they all require f out of m servers being honest, and seem equivalent in their trust and privacy implications, the way in which these servers are used dictates if metadata privacy guarantees are upheld or not. For instance, Atom's manytrust assumption hopes to distribute at least one of these f honest servers into each group that realizes a logical node in the permutation network used to route messages. Consequently to subvert even a single user's message, the adversary would have to subvert at least one entire path on the entire permutation network; since Atom reuses the nodes of the permutation network, this means the adversary would have to effectively subvert all the anytrust groups in the system to deanonymize any traffic.[11] Hence, this is the strongest privacy guarantee one can extract under the manytrust model, as unless all manytrust groups are subverted, clients' metadata privacy guarantees are upheld.

A weaker variant of manytrust is achieved by designs like XRD, Stadium, and Karaoke, that leverage multiple anytrust groups in isolation for manytrust. Recall in XRD, the manytrust assumption arises from leveraging anytrust assumption over each chain of servers. Hence subverting even one chain of servers that depends on anytrust for its metadata privacy results in the adversary being able to undermine metadata privacy of all communications over that particular chain. Similarly for Stadium and Karaoke, the system creates several input and output chains each of which leverage the anytrust assumption, resulting in the manytrust assumption. In this setup, any adversarially controlled input and output chain pair, leaks the metadata of all communications that use these two chains.

The manytrust assumption is at its weakest in Loopix due its stratified network structure, which implies messages get mixed over a few hops of mixing relays. Hence leveraging the same trust assumption of f servers out of all the n servers in the system be honest, results in an uneven privacy distribution as some subset of conversations will have all servers on their hop path be adversarially controlled, resulting in no metadata privacy guarantees.

6.4 Application: Metadata-Protected Messaging

The MPCS we cover in this work have been designed with two broad application goals, namely anonymous broadcast and metadata-protected messaging. While most of the existing CUS schemes have been proposed towards messaging, we observe that the vast majority of these schemes do not achieve low enough latencies to be a practical messaging system. In order to be usefully deployable, a messaging system requires at least three fundamental properties: (i) low latency, (ii) support for asynchrony, and (iii) horizontal scalability. Table 1 highlights systems that have low latencies (even with a generous definition of 'low latency'), and we observe that only eight systems achieve this.[12]

Out of these eight systems, Pung, SealPIR, Express, Riposte, and Sabre (the RPIR and PIR based schemes) all have a fundamental problem that messages are processed sequentially; i,e, this low latency is only true if one user submits a message and others do not. If all users submit their messages at the same time, each message has to be processed sequentially and users would experience significant delays in delivering their messages. Furthermore, their horizontal scalability is quadratic in the amount of user growth ($k^2\times$ as seen in the table for an increase of $k\times$ users). So while these five systems support asynchrony, their low latency has the drawback of poor throughput, and subpar horizontal scalability. Loopix on the other hand has low latency and high throughput, but their asynchrony is at the expense of a trust assumption as detailed in Section 5.2, and additionally as we mentioned in Section 6.3.1 their design has an uneven privacy distribution. Finally, Karaoke and Groove support messaging with low latency and high throughput, and horizontal scalability, but fall short of supporting asynchrony. Groove proposes novel directions for temporary disconnection of users (a form of partial asynchrony), permitting users to go offline for a small

[11] Although if they accomplish this, they get to deanonymize all traffic in the system.

[12] Loopix has a partial marker for messaging latency, as their implementation only demonstrates a < 2 s latency with 500 clients, but the design of the system suggests that even as users grow the latency should not increase significantly more, as the route length remains same with more servers to distribute traffic over.

constant number of rounds after which they can no longer receive messages. We see from our systematization that there is a clear dearth of MPCS designs that can simultaneously achieve these three fundamental properties, and we hope that this prompts future research to focus on such constructions.

6.5 Asynchronous Message Storage

In the RUS and CUS designs that do not support asynchrony, there is no notion of message storage for later retrieval. However in the asynchronous paradigm, this poses an interesting challenge that has not been well addressed yet. Systems like Pung and SealPIR that leverage CPIR in fact can handle this type of asynchronous storage in the best fashion possible, at the expense of server storage and computation. The server simply stores all the messages received in each round, and clients that reconnect to the system after several rounds can poll each of the rounds they missed to retrieve any messages they missed. This action leaks no additional information as both the CPIR server as well as any global adversary is already aware of when the client last interacted with the system.

Talek introduces a novel middle ground for asynchronous message storage. Recall from Section 5.5 that Talek is initiated with system parameters that dictate the TTL for any message in the system. Older messages eventually get evicted to make space for newer messages. However interestingly their private notification system that uses a Bloom filter, retains the memory of existence of messages in a conversation. While the message itself may have been evicted, this Bloom filter system can notify a user that has gone offline for a long duration of time of the existence of messages they missed and can no longer retrieve. Such users can then request missed messages from their correspondents.

Loopix sidesteps this problem altogether as the design achieves asynchrony by trusting the service provider node, and reveals the number of real messages a client receives and the timing metadata of received messages to this trusted node.

7 DIRECTIONS FOR FUTURE RESEARCH

7.1 Robustness and Availability

Robustness is an important property of any system to ensure liveness and availability. The notion of robustness or handling server churn was discussed by Atom, but did not receive any further attention in most other mixnet designs until Trellis recently. Robustness has been extensively studied in the SMC literature, and unsurprisingly recent SMC-based MPCS schemes address robustness. More research needs to be done to investigate how to efficiently enable robustness within the other families of MPCS. In fact, Blinder shows promise in this regard, since it merges ideas from both the SMC (for robustness and censorship resistance) and RPIR literature (to reduce bandwidth and computation overheads).

7.2 Anonymity Set Protections

Most existing systems towards MPCS do not provide strong protections against anonymity set manipulation attacks as seen from Table 1. The ability to manipulate if a particular client participates in a communication system or not enables adversaries to perform several deanonymization attacks. Such an adversary, along with some Sybil clients, can easily subvert the victim's purported metadata privacy in a RUS or SUBS. The adversary simply manipulates the anonymity set or participants in a round to be composed of just the victim (padded up with Sybil client submissions) to deanonymize the recipient of their communication in the case of RUS, or the message they broadcasted in SUBS. While the definition of a CUS may seem to protect its clients from such attacks, that is not the case. An adversary with the ability to manipulate client participation can exploit the underlying scheme's structure to launch deanonymization attacks. For instance, recall from Section 5.2 that in XRD clients send and receive ℓ messages in each round. Excluding a client currently in an active conversation from participating in the system can result in their correspondent receiving one fewer message back to their mailbox in that round. Similarly, in Clarion if an active client is excluded from a round, it will result in their correspondent attempting to collect a message from an index that does not have a submission, again revealing the communication relationship. Hence an important avenue for future research is designing MPCS with strong anonymity set protection guarantees for honest clients participating in the system even in the face of malicious servers, lest they open up to such deanonymization attacks. Techniques towards this end have surfaced in the SUBS literature for the SMC (Asynchromix, Blinder, RPM) and RPIR (Spectrum) families and these ideas need to percolate back into CUS and RUS designs.

7.3 Asynchronous E2E Metadata-Protected Communications

As we point out in Section 6.2.2, there is a dearth of E2E MPCS that can support asynchronicty. There are several components that merit further research and clarity here. An important area of research for bringing MPCS closer to practice would be a more thorough study on the guarantees one can have in an asynchronous metadata-protecting E2E communication design. Further, techniques to handle asynchronous metadata-protected mailbox storage is another important avenue for further research. As detailed in Section 6.5, this area has received scant attention as asynchronous E2E systems themselves have not received sufficient attention.

8 CONCLUSION

In this work, we survey 31 recent systems proposed for metadata-protecting communications. Since each of these systems make different design choices and assumptions, they have their own advantages and disadvantages. In this work we group them into four categories based on the functionality and metadata-privacy guarantees they achieve, and into six families based on the core underlying technique that underpins the metadata protection. We examine the innate benefits and shortcomings of these categories and families of works towards this goal. We point out subtleties in the way some security assumptions are used to design MPCS that make certain systems provide a false sense of privacy, as well as identify promising directions for future research to further metadata-protected communications.

ACKNOWLEDGMENTS

We thank Sebastian Angel, Debajyoti Das, and our PoPETs reviewers for their feedback and discussions that helped improve this paper. We thank the Ontario Graduate Scholarships program, NSERC

(CRDPJ-534381), and the Royal Bank of Canada for supporting this work. This research was undertaken, in part, thanks to funding from the Canada Research Chairs program.

REFERENCES

[1] 2014. An Open Letter from US Researchers in Cryptography and Information Security. http://masssurveillance.info/. Accessed February 2023.

[2] Ittai Abraham, Benny Pinkas, and Avishay Yanai. 2020. Blinder – Scalable, Robust Anonymous Committed Broadcast. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[3] ACLU of Northern California. 2014. Metadata: Piecing Together a Privacy Solution. https://www.aclunc.org/sites/default/files/Metadata%20report%20FINAL%202%2021%2014%20cover%20%2B%20inside%20for%20web%20%283%29.pdf. Accessed February 2023.

[4] Nikolaos Alexopoulos, Aggelos Kiayias, Riivo Talviste, and Thomas Zacharias. 2017. MCMix: Anonymous Messaging via Secure Multiparty Computation. In *26th USENIX Security Symposium*.

[5] Sebastian Angel, Hao Chen, Kim Laine, and Srinath Setty. 2018. PIR with Compressed Queries and Amortized Query Processing. In *2018 IEEE Symposium on Security and Privacy (S&P)*.

[6] Sebastian Angel, David Lazar, and Ioanna Tzialla. 2018. What's a Little Leakage Between Friends?. In *Proceedings of the 2018 Workshop on Privacy in the Electronic Society (WPES)*.

[7] Sebastian Angel and Srinath Setty. 2016. Unobservable Communication Over Fully Untrusted Infrastructure. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*.

[8] Ludovic Barman, Moshe Kol, David Lazar, Yossi Gilad, and Nickolai Zeldovich. 2022. Groove: Flexible Metadata-Private Messaging. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*.

[9] BBC News. 2021. Putin critic Navalny jailed in Russia despite protests. https://www.bbc.com/news/world-europe-55910974. Accessed February 2023.

[10] Burton H. Bloom. 1970. Space/Time Trade-Offs in Hash Coding with Allowable Errors. (1970).

[11] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. 2019. Zero-Knowledge Proofs on Secret-Shared Data via Fully Linear PCPs. In *39th Annual International Cryptology Conference, Advances in Cryptology - CRYPTO*.

[12] Elette Boyle, Niv Gilboa, and Yuval Ishai. 2016. Function Secret Sharing: Improvements and Extensions. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[13] Buzzfeed News. 2020. The FinCEN Files. https://www.buzzfeednews.com/article/jasonleopold/fincen-files-financial-scandal-criminal-networks. Accessed February 2023.

[14] David Chaum. 1988. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptology* (1988).

[15] David Chaum. 2003. Untraceable Electronic mail, Return Addresses and Digital Pseudonyms. In *Secure Electronic Voting*.

[16] David Chaum, Debajyoti Das, Farid Javani, Aniket Kate, Anna Krasnova, Joeri De Ruiter, and Alan T Sherman. 2017. cMix: Mixing with Minimal Real-Time Asymmetric Cryptographic Operations. In *International Conference on Applied Cryptography and Network Security (ACNS)*.

[17] Raymond Cheng, William Scott, Elisaweta Masserova, Irene Zhang, Vipul Goyal, Thomas Anderson, Arvind Krishnamurthy, and Bryan Parno. 2020. Talek: Private Group Messaging with Hidden Access Patterns. In *Annual Computer Security Applications Conference (ACSAC)*.

[18] Koji Chida, Daniel Genkin, Koki Hamada, Dai Ikarashi, Ryo Kikuchi, Yehuda Lindell, and Ariel Nof. 2018. Fast Large-Scale Honest-Majority MPC for Malicious Adversaries. In *Advances in Cryptology - CRYPTO*.

[19] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. 1995. Private Information Retrieval. In *IEEE Foundations of Computer Science (FOCS)*.

[20] Henry Corrigan-Gibbs and Dan Boneh. 2017. Prio: Private, Robust, and Scalable Computation of Aggregate Statistics. In *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation (NSDI)*.

[21] Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. 2015. Riposte: An Anonymous Messaging System Handling Millions of Users. In *2015 IEEE Symposium on Security and Privacy (S&P)*.

[22] Henry Corrigan-Gibbs and Bryan Ford. 2010. Dissent: Accountable Anonymous Group Messaging. In *Proceedings of the 17th ACM conference on Computer and Communications Security (CCS)*. ACM.

[23] Henry Corrigan-Gibbs, David Isaac Wolinsky, and Bryan Ford. 2013. Proactively Accountable Anonymous Messaging in Verdict. In *22th USENIX Security Symposium*.

[24] George Danezis. 2003. Statistical Disclosure Attacks. In *IFIP International Information Security Conference*.

[25] George Danezis, Claudia Diaz, and Paul Syverson. 2009. Systems for Anonymous Communication. *CRC Handbook of Financial Cryptography and Security* (2009).

[26] Deutsche Welle. 2019. Iran defends execution of gay people. https://www.dw.com/en/iran-defends-execution-of-gay-people/a-49144899. Accessed February 2023.

[27] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. Tor: The Second-Generation Onion Router. In *13th USENIX Security Symposium*.

[28] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography Conference (TCC)*.

[29] Saba Eskandarian and Dan Boneh. 2022. Clarion: Anonymous Communication from Multiparty Shuffling Protocols. In *29th Network and Distributed System Security Symposium (NDSS)*.

[30] Saba Eskandarian, Henry Corrigan-Gibbs, Matei Zaharia, and Dan Boneh. 2021. Express: Lowering the Cost of Metadata-hiding Communication with Cryptographic Privacy. In *30th USENIX Security Symposium*.

[31] Nethanel Gelernter, Amir Herzberg, and Hemi Leibowitz. 2018. Two Cents for Strong Anonymity: The Anonymous Post-office Protocol. In *Cryptology and Network Security (CANS)*.

[32] Niv Gilboa and Yuval Ishai. 2014. Distributed Point Functions and Their Applications. In *Advances in Cryptology - EUROCRYPT*.

[33] Koki Hamada, Ryo Kikuchi, Dai Ikarashi, Koji Chida, and Katsumi Takahashi. 2013. Practically Efficient Multi-party Sorting Protocols from Comparison Sort Algorithms. In *Information Security and Cryptology (ICISC) 2012*.

[34] Johan Håstad. 2006. The square lattice shuffle. *Random Struct. Algorithms* (2006).

[35] IETF. 2018. TLS v1.3. https://www.rfc-editor.org/rfc/rfc8446.txt. Accessed February 2023.

[36] Christiane Kuhn, Martin Beck, Stefan Schiffner, Eduard Jorswieck, and Thorsten Strufe. 2019. On Privacy Notions in Anonymous Communication. *Proceedings on Privacy Enhancing Technologies (PoPETs)* (2019).

[37] Albert Kwon, Mashael AlSabah, David Lazar, Marc Dacier, and Srinivas Devadas. 2015. Circuit Fingerprinting Attacks: Passive Deanonymization of Tor Hidden Services. In *24th USENIX Security Symposium*.

[38] Albert Kwon, Henry Corrigan-Gibbs, Srinivas Devadas, and Bryan Ford. 2017. Atom: Horizontally Scaling Strong Anonymity. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP)*.

[39] Albert Kwon, David Lazar, Srinivas Devadas, and Bryan Ford. 2016. Riffle. *Proceedings on Privacy Enhancing Technologies (PoPETs)* (2016).

[40] Albert Kwon, David Lu, and Srinivas Devadas. 2020. XRD: Scalable Messaging System with Cryptographic Privacy. In *Proceedings of the 17th Usenix Conference on Networked Systems Design and Implementation (NSDI)*.

[41] Simon Langowski, Sacha Servan-Schreiber, and Srinivas Devadas. 2022. Trellis: Robust and Scalable Metadata-private Anonymous Broadcast. Cryptology ePrint Archive, Paper 2022/1548. https://eprint.iacr.org/2022/1548 https://eprint.iacr.org/2022/1548.

[42] David Lazar, Yossi Gilad, and Nickolai Zeldovich. 2018. Karaoke: Distributed Private Messaging Immune to Passive Traffic Analysis. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*.

[43] Sarah Jamie Lewis. 2018. Cwtch: Privacy Preserving Infrastructure for Asynchronous, Decentralized, Multi-Party and Metadata Resistant Applications. https://cwtch.im/cwtch.pdf.

[44] Donghang Lu and Aniket Kate. 2022. RPM: Robust Anonymity at Scale. Cryptology ePrint Archive, Paper 2022/1037. https://eprint.iacr.org/2022/1037 https://eprint.iacr.org/2022/1037.

[45] Donghang Lu, Thomas Yurek, Samarth Kulshreshtha, Rahul Govind, Aniket Kate, and Andrew Miller. 2019. HoneyBadgerMPC and AsynchroMix: Practical Asynchronous MPC and Its Application to Anonymous Communication. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[46] Nick Mathewson and Roger Dingledine. 2004. Practical Traffic Analysis: Extending and Resisting Statistical Disclosure. In *International Workshop on Privacy Enhancing Technologies (WPES)*.

[47] Carlos Aguilar Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. 2016. XPIR: Private Information Retrieval for Everyone. *Proceedings on Privacy Enhancing Technologies (PoPETs)* (2016).

[48] Prateek Mittal and Nikita Borisov. 2009. ShadowWalker: Peer-to-peer Anonymous Communication using Redundant Structured Topologies. In *16th ACM Conference on Computer and Communications Security (CCS)*.

[49] Prateek Mittal and Nikita Borisov. 2012. Information Leaks in Structured Peer-to-Peer Anonymous Communication Systems. *ACM Transactions on Information and System Security (TISS)* (2012).

[50] Arjun Nambiar and Matthew Wright. 2006. Salsa: A Structured Approach to Large-Scale Anonymity. In *Proceedings of the 13th ACM conference on Computer and Communications Security (CCS)*.

[51] Zachary Newman, Sacha Servan-Schreiber, and Srinivas Devadas. 2022. Spectrum: High-Bandwidth Anonymous Broadcast with Malicious Security. In *Proceedings of the 19th USENIX Conference on Networked Systems Design and Implementation (NSDI)*.

[52] NYR Daily. 2014. We Kill People Based on Metadata. https://www.nybooks.com/daily/2014/05/10/we-kill-people-based-metadata/. Accessed February 2023.

[53] Rasmus Pagh and Flemming Friche Rodler. 2004. Cuckoo Hashing. (2004). https://doi.org/10.1016/j.jalgor.2003.12.002
[54] Andriy Panchenko, Fabian Lanze, Jan Pennekamp, Thomas Engel, Andreas Zinnen, Martin Henze, and Klaus Wehrle. 2016. Website Fingerprinting at Internet Scale. In *23rd Network and Distributed System Security Symposium (NDSS)*.
[55] Andriy Panchenko, Stefan Richter, and Arne Rache. 2009. NISAN: Network Information Service for Anonymization Networks. In *16th ACM Conference on Computer and Communications Security (CCS)*.
[56] Ania M. Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. 2017. The Loopix Anonymity System. In *26th USENIX Security Symposium*.
[57] T. Rabin and M. Ben-Or. 1989. Verifiable Secret Sharing and Multiparty Protocols with Honest Majority. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing (STOC)*.
[58] Marc Rennhard and Bernhard Plattner. 2002. Introducing MorphMix: peer-to-peer based anonymous Internet usage with collusion detection. In *Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society (WPES)*.
[59] Reuters. 2018. UN Experts says Egypt systematically targets rights activists. https://www.dw.com/en/iran-defends-execution-of-gay-people/a-49144899. Accessed February 2022.
[60] Phillip Rogaway. 2015. The Moral Character of Cryptographic Work. (2015).
[61] Len Sassaman, Bram Cohen, and Nick Mathewson. 2005. The Pynchon Gate: A Secure Method of Pseudonymous Mail Retrieval. In *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society (WPES)*.
[62] Max Schuchard, Alexander W. Dean, Victor Heorhiadi, Nicholas Hopper, and Yongdae Kim. 2010. Balancing the Shadows. In *Proceedings of the 9th Annual ACM Workshop on Privacy in the Electronic Society (WPES)*.
[63] Andrei Serjantov, Roger Dingledine, and Paul Syverson. 2002. From a Trickle to a Flood: Active Attacks on Several Mix Types. In *International Workshop on Information Hiding*.
[64] Adi Shamir. 1979. How to Share a Secret. *Communicatios of the ACM* (1979).
[65] Fatemeh Shirazi, Milivoj Simeonovski, Muhammad Rizwan Asghar, Michael Backes, and Claudia Diaz. 2018. A Survey on Routing in Anonymous Communication Protocols. *Comput. Surveys* (2018).
[66] Signal Foundation. 2013. Advanced Cryptographic Ratcheting. https://signal.org/blog/advanced-ratcheting/. Accessed February 2023.
[67] Parisa Tabriz and Nikita Borisov. 2006. Breaking the Collusion Detection Mechanism of MorphMix. In *Privacy Enhancing Technologies (PET)*.
[68] The Guardian. 2013. NSA Files Decoded: What the revelations mean for you. https://www.theguardian.com/world/interactive/2013/nov/01/snowden-nsa-files-surveillance-revelations-decoded#section/1. Accessed February 2023.
[69] Nirvan Tyagi, Yossi Gilad, Derek Leung, Matei Zaharia, and Nickolai Zeldovich. 2017. Stadium: A Distributed Metadata-Private Messaging System. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP)*.
[70] A. Vadapalli, K. Storrier, and R. Henry. 2022. Sabre: Sender-Anonymous Messaging with Fast Audits. In *IEEE Symposium on Security and Privacy (S&P)* .
[71] Jelle Van Den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. 2015. Vuvuzela: Scalable Private Messaging Resistant to Traffic Analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*. ACM.
[72] Qiyan Wang, Prateek Mittal, and Nikita Borisov. 2010. In Search of an Anonymous and Secure Lookup: Attacks on Structured Peer-to-Peer Anonymous Communication Systems. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS)*.
[73] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. 2014. Effective Attacks and Provable Defenses for Website Fingerprinting. In *23rd USENIX Security Symposium*.
[74] Washington Post. 2018. Leak charges against Treasury official show encrypted apps only as secure as you make them. https://www.washingtonpost.com/news/powerpost/paloma/the-cybersecurity-202/2018/10/18/the-cybersecurity-202-leak-charges-against-treasury-official-show-encrypted-apps-only-as-secure-as-you-make-them/5bc74eaa1b326b7c8a8d1a3a/. Accessed February 2023.
[75] WIRED. 2014. Whatsapp Switches to End-to-End Encryption. https://www.wired.com/2014/11/whatsapp-encrypted-messaging/. Accessed February 2023.
[76] David Isaac Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. 2012. Dissent in Numbers: Making Strong Anonymity Scale. In *10th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*.
[77] David I Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. 2012. Scalable Anonymous Group Communication in the Anytrust Model. In *Proceedings of the Fifth European Workshop on System Security (EuroSec 2012)*.
[78] Bassam Zantout and Ramzi Haraty. 2011. I2P Data Communication System. In *10th International Conference on Networks (ICN)*.

A LARGEST EXPERIMENTS REPORTED

Table 2 reports the details of the largest experiment conducted by each of the systems we discussed in Section 5. This provides a concrete timing benchmark for each system albeit constrained by their choices of experimental setup. This prevents us from an apples-to-apples comparison across designs since each of these works i) leverage different underlying machines for their experiments, ii) choose experiment setups that highlight their system's advantages, and iii) have nuanced goals and assumptions. For instance, timing benchmarks of asynchronous and synchronous systems typically have very different implications. Systems from the DP family like Karaoke transmit one real message from each client in a round implying a high throughput. On the flip side, private-write based systems like Sabre perform low-latency delivery of a single message from a single client, and their throughput is fundamentally limited by n, the number of clients in the system and the time taken to process a single request, which scales with n.

Nonetheless these timing benchmarks give us a concrete sense of the state of affairs of current MPCS designs, and the stark gap in performance with popular messenger choices today that can accommodate billions of users with latencies in the order of milliseconds. We also note that the timings reported in Table 2 are a lower bound since the timing benchmark quoted for several of the designs in the table do not account for the overheads of setup phases such as the shuffling of keys in the DC-net based schemes, or the cost of the dialing protocol in synchronous schemes. The cost of these phases would further increase the timing benchmark, although such costs are typically amortized over several rounds of communication. Next we contrast benchmarks of representative candidates from the different families of metadata-protecting communication systems we have discussed.

Synchronous Constructions. Systems from the DC-net, mixnet, DP, and SMC based families are typically synchronous in nature. The exception is Loopix, which supports asynchrony at the expense of trusting special service provider nodes in the system. DC-net based schemes scale very poorly as one would expect. The most efficient construction we have seen so far is Verdict [23] with 1000 clients, and 24 servers, takes more than 10 s to transmit one message of 128 bytes, and note that this is in fact assuming no client is actively attempting to disrupt the network, nor accounting for the shuffle protocol that is required to slot all the participants in the network for its communication round, both of which further exacerbates the time to transmit a message. Furthermore as detailed in Section 6, none of these construction can scale horizontally.

SMC-based constructions can serve more clients than the DC-net based ones, but note that more servers do not translate to reduced latency or higher throughput in these SMC-based systems; i.e., these schemes too do not scale horizontally. In terms of performance, Clarion in their prototype demonstrate that they can transmit 1 message of 160 bytes from each of the 10^6 clients to its intended recipient in 80 s with a three-server instantiation of their protocol.

In contrast, recent constructions from the mixnet based and DP based families have been designed with the explicit goal of horizontal scalability. For instance, the mixnet based scheme XRD's experiments demonstrates that with 2×10^7 clients all sending a 100-byte message to another client in the system takes 251 s. While Karaoke the most recent work in the DP based family in their experiments demonstrates that with 1.6×10^7 clients (a similar number of clients as XRD) and each sending a message to another

Table 2: Details of the largest experiments reported (if experimental results were provided) by the systems detailed in Section 5.

System	Year	Type	Family	Largest Experiment Reported ($n\|s\|m\|b\|t$)				
				Clients (n)	Messages Sent (s)	Servers (m)	Message Size (b)	Time (t)
Vuvuzela [71]	2015	CUS (Mix)	DP	2×10^6	2×10^6	3	256	55 s
Pung [7]	2016	CUS (RUMS)	PIR	10^6	1	1	288	1.3 s
AnonPOP [31]	2016	CUS (Mix)	M	5×10^5	5×10^5	4	1000	30 s
Stadium [69]	2017	CUS (Mix)	DP	5×10^7	5×10^7	100	136	200 s
MCMix [4]	2017	CUS	SMC	10^5	10^5	3	256	100 s
SealPIR [5]	2018	CUS (RUMS)	PIR	2.56×10^5	1	1	288	0.51 s
Karaoke [42]	2018	CUS (Mix)	DP	1.6×10^7	1.6×10^7	100	256	28 s
XRD [40]	2019	CUS	M	8×10^6	8×10^6	100	256	> 16 m
Clarion [29]	2021	CUS	SMC	10^6	10^6	3	160	80 s
Groove [8]	2022	CUS (Mix)	DP	1.5×10^8	1.5×10^8	100	56	80 s
Dissent [22]	2010	CUS/SUBS	DC	40	1	0	10^6	> 14 m
D3 [77]	2012	CUS/SUBS	DC	576	1	10	256	> 2 m
Dissent v2 [76]	2012	CUS/SUBS	DC	1000	1	24	128	> 10 s
Verdict [23]	2013	CUS/SUBS	DC	1000	1	8	128	> 10 s
Riposte [21]	2015	SUBS	RPIR	3.8×10^5	1	3	160	0.35 s
Riffle [39]	2016	SUBS	M	1.2×10^6	10^6	3	160	> 10 s
Atom [38]	2017	SUBS	M	10^7	10^7	1024	32	28 m
AsynchroMix [45]	2019	SUBS	SMC	4096	4096	100	32	120 s
PowerMix [45]	2019	SUBS	SMC	1024	1024	100	32	140 s
Blinder [2]	2020	SUBS	SMC	10^6	10^6	20	160	8 m
Spectrum [51]	2021	SUBS	RPIR	10^5	1	2	10^6	> 500 s
RPM [44]	2022	SUBS	SMC	1.6×10^5	1.6×10^5	4	16	48.14 s
Sabre [70]	2022	SUMS/SUBS	RPIR	2^{18}	1	2	1000	0.05 s
Express [30]	2021	SUMS	RPIR	2^{18}	1	2	1000	0.5 s
cMix [16]	2016	RUS (Mix)	M	500	1000	5	256	4.6 s
Loopix [56]	2017	RUS (Mix)	M	500	3×10^4	10	224	1.9 s
Talek [17]	2020	RUS (RUMS)	PIR	3.2×10^4	1	3	1000	1.7 s

client in the system, can transmit all of these messages in a 100-server instantiation of their protocol in 28 s; i.e., almost an order of magnitude faster, at the expense of the weaker privacy guarantees due to DP as we detailed in Section 5.3. Ultimately the biggest drawback in these schemes remains the synchronicity assumption. Schemes like Karaoke can serve close to 17 million clients in tens of seconds, but the expectation that all these clients will remain online without any disruption in perpetuity is extremely unrealistic, and disconnections of a user during a conversation harms the privacy of both participants of the conversation.

Asynchronous Constructions. PIR can be leveraged to design asynchronous MPCS, as seen in Express [30], Talek [17], and Sabre [70]. The only other asynchronous design is Loopix (mixnet), but it requires trusting service provider nodes to support asynchrony.

Sabre is the latest iteration of private-write based metadata-hiding communication networks. In their work, their experiments demonstrate that in a two-server instance of Sabre with 2^{18} registered mailboxes a client can deliver a message to any of these mailboxes in 0.05 s, which is the smallest latency (ignoring throughput) by almost two orders of magnitude in comparison with the other works we have detailed. However as we mention in Section 5.5, as more users enter such systems the cost to deliver a message increases linearly. Furthermore, these systems incur quadratic overheads for scaling horizontally.

Loopix on the other hand attains asynchronous communication capabilities by leveraging a semi-trusted service provider node in their construction. Their experimental evaluation is limited to a 500-client instantiation with a network comprising of 4 service providers and 6 mixnet nodes. Their end-to-end latency in this setup is 1.9 s with other experiments that justify that with increase in clients and servers this end-to-end cost would remain unaffected, and is mostly dependent on the system parameter that tunes the delay for a message at each of these mixes. However, as we detail in Section 5.2, a malicious service provider node undermines recipients' metadata privacy.