tigro: Trust Infrastructure for Grassroots Organizing via Grounded Digital Annotations

Leah Namisa Rosenbloom
Northeastern University
Boston, MA, USA
l.rosenbloom@northeastern.edu

Seny Kamara
Brown University and
MongoDB Research
New York, NY, USA
seny.kamara@mongodb.com

Zachary Espiritu
MongoDB Research
New York, NY, USA
zachary.espiritu@mongodb.com

Tarik Moataz
MongoDB Research
New York, NY, USA
tarik.moataz@mongodb.com

Amine Bahi
École Normale Supérieure
Paris, France
amine.bahi@ens.psl.eu

John Wilkinson
Brown University
Providence, RI, USA
john_wilkinson@alumni.brown.edu

Abstract

Grassroots organizing requires establishing trust in digital artifacts (like event announcements or calls to action) while navigating significant security threats including surveillance, infiltration, and state violence. Traditional trust infrastructures like PKI and Web of Trust fail to address these specific needs, as they create public records of trust relationships that can expose activist networks and require institutional involvement that may be inaccessible or dangerous for marginalized communities.

To address this, we introduce tigro, a novel trust infrastructure and system designed specifically for grassroots organizing contexts. Unlike conventional trust infrastructures, tigro implements a two-tier trust model: *ground trust*, which cryptographically binds digital annotations to physically vetted individuals, and *artifact trust*, which enables private, need-to-know sharing of assessments about digital content via annotations. Our protocol begins with an in-person key exchange that establishes a shared cryptographic key, creating a secure bridge between activists' existing physical vetting practices and their digital trust needs.

To realize this approach, we define a new cryptographic primitive called an encrypted annotation system (EAS) and construct tigro using structured encryption and anonymous channels. We present two implementations with different security-performance tradeoffs: an efficient version for practical deployment that handles annotations in under a second, and a subliminal version that reveals virtually no metadata. Through this design, tigro enables activists to securely verify digital content without compromising relationship privacy or creating surveillance vulnerabilities, addressing a critical gap in existing trust infrastructure.

1 Introduction

The establishment of trust in digital objects is a foundational challenge in computer science and broadly underpins modern technological infrastructure. Since the advent of networked computing and the rise of the Internet, digital trust has been essential for ensuring the security, reliability, and authenticity required by electronic commerce, communication systems, and critical infrastructure systems. Recognizing this centrality, governments, standardization bodies and tech companies have collaborated to create and maintain global-scale digital trust infrastructures that are critical to the functioning of the entire digital ecosystem. Prominent examples include the Public-Key Infrastructure (PKI), which provides mechanisms for authenticating public keys and forms the backbone of the Internet. Another crucial digital trust mechanism is Certificate Transparency, which enhances PKI by enabling public auditing of certificate issuance. The Domain Name System Security Extensions (DNSSEC) is a mechanism to authenticate domain names and the web of trust is a decentralized way to authenticate public keys.

Beyond conventional trust. Our current trust infrastructures are essential, but they were primarily designed by industry and governments to support electronic commerce and governmental needs. Consequently, they mainly address the specific problems and requirements of these entities. Electronic commerce and governmental applications, however, are not the only contexts that require digital trust solutions. Numerous other groups, such as activist communities, non-governmental organizations, journalists and marginalized populations with limited institutional access, also need to establish trust in digital objects. These groups frequently possess distinct requirements, constraints, and threat models, which conventional trust infrastructures do not accommodate. For example, grassroots organizers often face threats such as infiltration by state actors posing as legitimate activists, fake events designed to identify and surveil movement participants, misinformation campaigns designed to disrupt organizing efforts, and the risk of social network mapping through public verification.

A trust infrastructure for grassroots organizing. Given these unique challenges, we propose a new trust infrastructure called tigro specifically designed for grassroots organizing contexts. Unlike conventional trust infrastructures, ours is built upon activists' existing security practices, particularly their physical vetting protocols and need-to-know information sharing approaches. It implements a two-tiered trust model: *ground trust*, which cryptographically binds digital annotations to physically vetted individuals; and *artifact trust*, which enables the secure sharing of evaluations about digital

Proceedings on Privacy Enhancing Technologies 2026(2), 354–383

https://doi.org/10.56553/popets-2026-0052

artifacts through annotations. This design addresses the specific threat models faced by activists by: (1) keeping trust relationships private rather than public; (2) requiring physical co-presence for initial trust establishment, making infiltration more difficult and detectable; (3) compartmentalizing information on a need-to-know basis; and (4) depending on its instantiation, enabling community-based assessment of digital content without revealing social ties. By rooting digital trust in existing physical vetting practices rather than institutional verification, we create a trust infrastructure that aligns with grassroots organizing practices while providing cryptographic security guarantees appropriate for high-risk contexts.

Community-driven cryptography. We note that the rethinking of technology and computer science research away from the sole needs of companies and governments and towards the needs of communities and marginalized groups is a recent but important endeavor. Recent examples of such work in cryptography include [62, 101, 116, 125, 150, 156]. This shift represents a broader recognition that technology should serve diverse social contexts and needs, not just commercial and governmental interests.

This paper contributes to this emerging body of work by addressing specific trust challenges faced by grassroots organizers: the need to verify digital artifacts within trusted networks while maintaining privacy and security in high-risk environments. Rather than imposing external technical solutions, we build upon ethnographic studies of activist security practices to develop a trust infrastructure that extends, rather than replaces, existing community verification processes. By centering the security and privacy requirements of grassroots communities, we can develop trust infrastructures that are more inclusive, contextually appropriate, and aligned with the actual practices of these communities.

1.1 Trust in Grassroots Organizing

Grassroots organizing is a process by which people work from within marginalized communities to effect change. Because participants in grassroots movements (activists) are working to challenge dominant power structures, they are disproportionately subject to police brutality, incarceration, surveillance, censorship, and assassination [32, 34, 46, 100, 137, 143, 161].

Safety vs. visibility. The complex and dynamic relationship between grassroots organizing efforts and digital technologies is very well-documented [6, 8, 17, 23, 31, 33, 49, 53, 60, 74–76, 84, 85, 87, 88, 96, 117–119, 124, 133, 138, 154, 159, 165, 167, 168, 170–172], and emphasizes that digital technologies are both a facilitator of organizing work and an added risk for movement participants. Specifically, digital technologies can increase the speed, scope, and scale of "frontstage" movement work like protests [53, 133, 153, 168], while also fortifying "backstage" communication infrastructure for the everyday, movement-sustaining work of building community and shared identity, planning, and reflection [53, 118, 133, 139, 140, 168, 169, 172]. They also greatly widen the surface of contact between activists and surveillance structures, both mass and targeted, of state and local adversaries [32, 87, 129, 133, 143, 173]. This contact puts them at greater risk of technology-facilitated harm. Lokot poses the problem of technology for grassroots organizing as a nuanced trade-off between safety and visibility [124].

Digital visibility. While the safety-visibility paradigm for movement building existed long before digital technologies, there are aspects of the current trade-off that are unique to the digital era. The risks associated with digital visibility are exacerbated by technology design and deployment that, despite all of the evidence of activists' relationships given above, ignore activists' needs and experiences. For example, cryptographers and technology designers have historically defined security and trust on an individualized basis, enforcing single-party authentication and access protocols [18, 174]. Revoking visibility via digital content deletion, which is immensely important to activists [6, 32, 53], is understudied in the cryptography and privacy technology literature. Few digital tools and platforms give users the option of accessible risk-based privacy differentiation [119], and most fail to communicate specific guarantees, rationales, and procedures, leading to rampant misconceptions about digital privacy practices [2, 6, 56, 61]. There are few if any information and communication technologies (ICTs) designed in collaboration with activists and with activists' specific needs in mind. In fact, it is a stated goal of many platforms to be universally accessible, while in reality they are largely modeled after the needs of a minority [51], and can lead to feelings of content inundation [138] and context collapse [128], which occurs when people try to use the same interface for different functions and among disparate groups of people.

Safety in digital spaces. Scholars and activists have critiqued cryptographers for proposing technological solutions for marginalized people such as activists that ignore the important context of their experiences [18, 75, 83, 101, 151, 155, 174]. In this work, we apply insights from transdisciplinary studies of grassroots organizing technology praxis toward developing cryptographic tools that facilitate two specific and important tasks of grassroots organizers: *vetting* and *sharing information on a need-to-know basis*. More broadly, we ask:

> (**Q**) *How might we adapt the existing trust and communication protocols of grassroots organizers from physical to digital spaces, without increasing the risk of surveillance, disinformation, and infiltration of grassroots movements?*

Vetting. Vetting is the act of verifying the identity, intentions, and trustworthiness of people, places, or things for the purposes of including them in spaces and actions associated with collective risk, and an essential part of the "onboarding" process in organizing work [6, 32]. For people, vetting is a time-consuming process involving physical meetings and existing point-to-point communication networks, which organizers use to talk privately with trusted contacts in order to verify details of the person's background and previous engagement in shared communities, values, and related action. Once people are vetted and have developed trusted connections with other members of the organization, they can wield their collective knowledge and connections to vet other people, places, or things such as externally-organized events, social media profiles, or news articles [53]. While digital tools that provide end-to-end encryption can facilitate the process of private point-to-point communication among organizers, they are not designed specifically for vetting: each conversation must happen separately

(or divided amongst various group chats) and in a non-organized fashion (in-line with potentially unrelated personal conversations, leading to confusion and context collapse [128, 138]). We consider the in-person vetting of people to be a foundational component of grassroots organizing, and aim to build technology that leverages in-person connection, or what we call *ground trust*, to facilitate the vetting of digital artifacts.

Vetting digital artifacts. Vetting non-human entities such as digital identities, public group pages, event invitations, news bulletins, social media posts, and especially direct action proposals is also crucial, as actions organized by untrustworthy, inexperienced, or misinformed parties can put activists in danger [53, 154]. For example, during the rapid expansion of organizing in June 2020 following the police killing of George Floyd, Black Lives Matter (BLM) activists in Philadelphia encountered events on social media that appeared to be BLM events, but were actually organized by members of the neo-fascist White nationalist group Proud Boys posing as BLM organizers [154]. In order to prevent the spread of untrustworthy digital content from destabilizing the movement, Philadelphia BLM activists turned to existing social networks and trusted long-time organizers for digital artifact vetting and fact-checking. Similarly, participants in the Sudanese revolution used a chain of trusted one-to-one contacts and "manual heuristics" to unearth primary sources for information, as well as a form of "crowdsourced content moderation" which leveraged existing trust networks to establish collective digital content-vetting procedures [53]. Daffalla et al. also note that while "some app features (such as livestream and reporting mechanisms) supported activists in building trust and disseminating verifiable information," many of social media platforms' anti-misinformation practices, for instance requiring singular individualized identities and context collapse, subverted activists' needs and interests.

1.2 Encrypted Annotation Systems

To address (**Q**), our goal is to design a system that allows activists to vet digital artifacts by leveraging the judgment of their community, whom they know and trust [6, 53, 133, 154]. Trusted group members, in this context, refers to individuals that they have personally met and vetted through their own *real-world protocols*, e.g., an organizer has met these individuals in person and confirmed their credibility using established organization-specific vetting protocols.

Digital annotation systems. The vetting of digital artifacts by trusted community members would be relatively straightforward if it could be conducted in person. However, in practice, being physically present with community members every time an artifact needs to be vetted is neither realistic nor scalable. Therefore, to ensure scalability and usability, we offer a vetting process that can be conducted digitally, although an initial one-time vetting of group members must occur in person.

To address this, we propose that a group use a digital annotation system which allows users to attach messages, referred to as annotations, to digital artifacts and share them with other members of their community. These annotations serve as an indication of an artifact's credibility based on the judgment of trusted individuals, enabling the community to vet and evaluate digital artifacts. Digital annotation systems like Hypothesis [90] and NB [179] are becoming increasingly common.

Threats. Such a system would allow organizers to address misinformation, but it also introduces several non-standard threats that must be carefully considered:

- *data confidentiality:* digital annotations could expose very sensitive and private information if the system is not properly secured. Ensuring that only authorized individuals can view or modify annotations is crucial to protect member privacy and safety.
- *re-identification*: even if the annotations themselves are secured, the identities of the community members and the structure of the community's social graph can still be highly sensitive. Information about who is connected to whom, as well as the patterns of interactions between members may inadvertently reveal personal details about individuals or their affiliations which can pose a significant privacy risk.
- *infiltration:* if malicious actors infiltrate the community, they could deliberately mislead others by attaching false annotations to digital artifacts. To mitigate this, the system must include safeguards to detect and prevent unauthorized access or manipulation of the annotation process.

Encrypted annotation systems. To support activists in the vetting process, we propose a new kind of digital communication system we refer to as an *encrypted annotation system* (EAS) that addresses the above threats as follows:

- *end-to-end encryption*: an annotation that is sent from a vetter to a vetee is end-to-end encrypted using a key that is known only to this pair of participants. As such, the server on which the EAS is run does not learn any information about the annotations it manages.
- *anonymity*: the real-world identities of the group members are hidden from the server and group infiltrators.
- *ground trust*: members can only receive annotations from parties they have vetted in person.

Overview. We envision that participants in an encrypted annotation system will form a pairwise shared identity, authenticated by a shared secret key, that is rooted in a physical meeting rather than a public-key infrastructure. All of the participants' communications with the EAS server are performed and authenticated based on these shared keys. Thus, participants have a *collective* rather than individual identity with respect to the server. They use the shared identity to create a shared and encrypted *box* through which they can communicate asynchronously.

An EAS facilitates vetting by allowing participants to write a private annotation associated with a digital artifact (e.g., group page, social media post, or digital event flyer) they wish to vet. The "vetter" labels the annotation with the globally unique identifier of the artifact (e.g., computed using a cryptographic hash function) and drops it in the private boxes it shares with its trusted contacts. The annotation will stay in the box unless it is deleted. If any of the group members wish to ascertain whether any of their trusted contacts has evaluated the trustworthiness of an artifact, they can compute the artifact's unique identifier and search their shared private boxes for annotations related to the identifier. Note that

this approach also allows members to vet public digital identities, since the artifact could be a social media profile. In the case of Philadelphia BLM protests, activists would have been able to use such a system to vet protests by quickly finding out (1) whether trustworthy people they had met "on-the-ground" at previous BLM actions would be attending, and (2) whether any of their trusted contacts could vouch for the organizers.

This asynchronous, label-restricted communication style is motivated by the collective security culture principle of sharing information on a need-to-know basis [6, 32, 161]. This principle protects movement participants in multiple respects: people who need to know sensitive information in order to safely engage in specific activities are always informed, while people who do not need access cannot be coerced into harming themselves or others based on retaining the information in their memory (physical or digital). Our second goal is to implement the EAS in such a way that is within the rhythm of activists' existing security praxis—to facilitate annotation sharing on a need-to-know basis.

1.3 Our Contributions

In this work, we demonstrate how trust infrastructure design can be reconceptualized to center the practices of grassroots organizers rather than commercial and governmental requirements. Specifically, we make the following contributions.

Synthesizing design goals. One of the core contributions of this work is the synthesis of relevant qualitative scholarship and sociotechnical context from historical grassroots movements into design considerations for encrypted annotation systems. We follow the paradigm shift in cryptographic protocol design from *Cryptography and Collective Power* [155], which centers the design justice principle of "one size fits one" [51], models digital trust as an extension of human trust, aims toward a more comprehensive notion of "big-picture" security, and optimizes on a grassroots scale. Due to space limitations, the tigro protocol's three core design goals are described in detail in Appendix B. At a high level, they include:

(1) *Digital trust = collective physical trust*: our first design goal is to establish digital trust that reflects the collective physical trust built as part of grassroots organizing. Movement participants often navigate between physical and digital spaces, and their trust in digital objects is grounded in objects' perceived trustworthiness within a shared community.
(2) *Information on a need-to-know basis*: the second design goal focuses on information sharing based on a need-to-know basis, ensuring that participants only access content relevant to their current context. This is achieved through a unique "drop and retrieve" mechanism within a digital private box, where annotations are associated with specific labels and can only be retrieved by participants who are familiar with the associated content. This system prevents the overload of information and protects against metadata collection often seen in traditional messaging platforms.
(3) *Think toward deployment*: the third design goal centers on the practical deployment of the system, with a focus on implementation details that are mostly left for future work. While we implement the cryptographic and functional principles of the system in a preliminary Android application, certain aspects such as device security and user interface design are best addressed in later iterations of the project.

Modeling identity and location. As discussed in the previous section, encrypted annotation systems must provide a set of security guarantees that span both digital and physical spaces. Specifically, these systems need to ensure confidentiality and anonymity against a digital adversary who may corrupt the server, observe the network, and attempt to infiltrate the group both digitally and physically. As far as we know, this adversarial setting has not been formally considered or defined in prior work, making one of our contributions the development of a definitional framework that that captures such adversaries.

To model location, we assume the existence of a metric space equipped with a notion of distance, and for every operation, each party is assigned a location within this space. The notion of identity is also central to our work because we need to both protect it and leverage it. As discussed in Section 1.2, the real-world identities of group members must be safeguarded (anonymity), while only annotations from parties who have been vetted in the real world can be received (groundedness). To capture this dual role, we define the identity of a party as the combination of a *physical identifier*, such as their appearance, and an *institutional identifier*, such as their name or address. In our setting, achieving anonymity requires hiding a party's institutional identifier, whereas enforcing groundedness relies on the party's physical identifier and, specifically, on a real-world process that can attest to that physical identity.

Grounded cryptography. In our setting, it is essential that all cryptographic protocols be *grounded*, meaning that if party A successfully executes a protocol with party B, then both parties must have vetted each other through an out-of-band process while in *physical proximity*. The requirement for physical proximity is crucial because it ensures that any potential adversary attempting to infiltrate the community must also be physically present, in which case, the standard vetting procedures of the activists can be employed to detect infiltration. To formally analyze the security guarantees of our protocols, we need to formalize the notion of groundedness. Our security definitions are expressed within the real/ideal-world paradigm, and we capture it by considering a (standalone) environment that generates the operations, along with a location function λ that maps each party to a location in the metric space. This location function λ is made available to all ideal functionalities, and groundedness is enforced by having the functionalities abort whenever either the physical distance between the parties exceeds a predetermined threshold or the receiver's physical identity does not match the one expected by the sender.

Technical overview. Our EAS, tigro, is a grounded encrypted system that enables parties to anonymously exchange and retrieve encrypted annotations with previously vetted parties. With the notion of groundedness now formulated, we extend the formal definitions of structured encryption to grounded structured encryption. More precisely, we build upon the simulation-based definitions from [63], which model multi-client structured encryption schemes as reactive secure computation protocols between a set of clients and a server. By adapting this framework, we show how to define and design the various components of our system:

- *grounded key exchange*: a core component of our protocols is a grounded key exchange, which enables two parties to generate a secret key over a public channel if and only if they are in close physical proximity. As we will see, using such a protocol to generate the secret key of an STE scheme results in a grounded STE scheme. To construct this protocol, we show that it is sufficient to execute a standard key exchange over what we call a *physically-attested proximity channel*. Roughly speaking, this is a channel that ensures a message can only be delivered if the sender and receiver are in close proximity and if the receiver's *physical* identity matches the one designated by the sender. We further demonstrate that physically-attested proximity channels can be instantiated using the exchange of QR codes via mobile devices.
- *grounded encrypted dictionaries*: a grounded encrypted dictionary is a multi-client encrypted data structure that stores label/value pairs and that supports Get, Put and Delete operations. The Get operation retrieves the label/value pair associated with a specific label, while the Put operation inserts a new label/value pair into the structure and the Delete operation removes the pair associated with a specific label. As mentioned above, we construct a grounded encrypted dictionary by using a grounded key exchange protocol to generate the secret key of any encrypted dictionary scheme.
- *grounded encrypted box-bank*: we extend the notion of a grounded encrypted dictionary to build a grounded encrypted box-bank, which consists of multiple mailboxes each holding uniquely identified and encrypted messages exchanged between two parties. Each box supports GetMail, DropMail and DeleteMail operations. The GetMail operation retrieves the message associated with a given label, the DropMail operation inserts a label/message pair into the box, and the DeleteMail operation removes the pair associated with a given label. We build a grounded encrypted box-bank by executing several grounded encrypted dictionaries in parallel.
- *grounded encrypted annotation system*: finally, we describe how to integrate these grounded encrypted structures into a complete grounded encrypted annotation system, which allows vetted parties to anonymously exchange and retrieve encrypted annotations. Participants securely store and access annotations using grounded encrypted box-banks.

By extending structured encryption to incorporate groundedness, our system ensures that all interactions between parties are not only encrypted but also bound to trusted group members that have been vetted in a physical context.

Leakage. Like any sub-linear encrypted search solution, structured encryption schemes can reveal information about the data and/or queries. The specific leakage a scheme reveals is formalized as a leakage profile.

We analyze the security of tigro and its building blocks by proving that they achieve a well-defined leakage profile. Our analysis, however, is black-box, meaning that the leakage profiles we obtain are a function of how tigro's underlying building blocks are instantiated. Black-box STE constructions and black-box leakage analysis are essential because they enable modular design, allowing us to achieve different efficiency vs. leakage tradeoffs depending on how the building blocks are instantiated.

An efficient instantiation. Based on our black-box design and leakage analysis, we show how to instantiate tigro using a variant of an efficient encrypted dictionary scheme from Cash, Jaeger, Jarecki, Jutla, Krawczyk, Rosu, and Steiner [39]. The resulting construction relies only on symmetric-key encryption $\mathsf{SKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, a pseudo-random function F, a hash function H and a key agreement protocol which we instantiate with AES-256, SHA-256, and ephemeral Diffie-Hellman, respectively. At a very high level, two parties $\mathbf{C}_a$ and $\mathbf{C}_b$ and the server start by generating a box identifier bxid by executing a key agreement protocol. The server then instantiates an empty dictionary structure for bxid. $\mathbf{C}_a$ and $\mathbf{C}_b$ then generate two shared keys K_F and K_E by executing a key exchange via QR codes. When one of the parties, say $\mathbf{C}_a$, wishes to send an annotation anno for a digital object obj to $\mathbf{C}_b$, it creates an object identifier by computing $\mathsf{oid} := H(\mathsf{obj})$ and sends the bxid and the pair $\big(F\big(F(K_F, \mathsf{oid}), 1\big), \mathsf{Enc}_{K_E}(\mathsf{anno})\big)$ to the server via Tor, which stores it in the dictionary. To retrieve an annotation for object obj, $\mathbf{C}_b$ computes $\mathsf{oid} := H(\mathsf{obj})$ and sends the bxid and $\mathsf{tag} := F\big(F(K_F, \mathsf{oid}), 1\big)$ to the server via Tor who queries the appropriate encrypted dictionary on tag and returns the associated ciphertext via Tor. Finally, $\mathbf{C}_b$ decrypts the ciphertext using K_E.

This instantiation, which we refer to as tigro$[\pi^*_{\mathsf{bas}}]$, is efficient and reveals the following: box sizes, correlations between add, get, and delete annotation operations for the same artifact, and the social network topology of the community. It also discloses metadata such as the times at which operations occur and the number of operations performed on each box. Perhaps surprisingly, we will demonstrate that, in our setting, the correlations revealed by this instantiation are not as problematic from a security perspective as they might initially appear. There are two key reasons for this. First, while it is theoretically possible for the scheme to reveal information about the artifacts, it is important to note that these artifacts are assumed to be public in the first place. More importantly, we will show that *regardless of how* tigro *is instantiated, it never leaks any information about the annotations themselves.*

We implemented tigro$[\pi^*_{\mathsf{bas}}]$ as a client-side library in Python and a server-side library in C++ and Protobuf that communicate using three different configurations: a *stub configuration*, which does not use Tor at all; a *single-circuit configuration*, which uses a single Tor circuit for all of a user's accesses (affording network-layer anonymity); and a *circuit-per-box configuration*, where a unique Tor circuit is used for every box (affording both network-layer anonymity and unlinkability). Our experiments show that even our unoptimized implementation of tigro$[\pi^*_{\mathsf{bas}}]$ is relatively practical. Specifically, our implementation took:

- *single-circuit*: with this configuration, with previously accessed boxes, uploading a 1KB annotation to 24 trusted contacts or searching for an annotation related to a given artifact takes about 0.9 seconds.
- *circuit-per-box*: with this configuration with previously accessed boxes, uploading a 1KB annotation to 24 trusted contacts takes about 24 seconds, and searching for an annotation takes about 26 seconds.

We note that because tigro$[\pi^*_{\text{bas}}]$ already reveals the social network topology, there is little additional security benefit in using the circuit-per-box configuration. Finally, we build an Android OS prototype of the client in Java using the tigro$[\pi^*_{\text{bas}}]$ client-side library.

A subliminal instantiation. As an alternative to tigro$[\pi^*_{\text{bas}}]$, we also demonstrate how to instantiate tigro with the EXH scheme from [24] in subliminal mode. This instantiation, which we refer to as tigro[EXH], effectively eliminates leakage, revealing no information beyond the existence of the structure itself and the number of boxes. Additionally, it hides the social network topology and all metadata, including the time and number of operations. By offering a much higher level of privacy, this instantiation prioritizes confidentiality, but it comes at the cost of reduced efficiency and correctness. Specifically, there could be queries for which no response will be received.

To achieve the tigro[EXH] construction, we had to extend the analysis of EXH in non-trivial ways. The key challenge is that EXH was originally analyzed as a client/server scheme, and does not naturally extend to a multi-client setting, which is critical for tigro. To address this limitation, we provide a new analysis of EXH in the context of a two-client setting, which is sufficient for our needs. This extended analysis requires non-trivial techniques from queuing theory and could be of independent interest. Due to space restrictions, we describe the analysis and empirical evaluation of the EXH-based tigro construction in Appendix K.

2 Related Work

Digital organizing technologies. There are many digital technologies with applications to grassroots organizing. We discuss additional details of these technologies in Appendix A and use this section to highlight the context most relevant to our setting. Among the most widely used and studied technologies for organizing are social media platforms [6, 21, 54, 85, 86, 96, 122, 124, 128, 133, 165–167, 176, 177], which afford activists networking and trust-building capabilities at the expense of unwanted attention from local adversaries and law enforcement [32, 133, 143, 161] and frustration surrounding algorithmic suppression [47, 142, 152], paternalism [57], and context collapse [128]. There is no privacy-preserving equivalent of social media in terms of efficiency, accessibility, and number of users, leading activists to engage with social media despite risks. And while they afford strong security guarantees, end-to-end encrypted communication tools [1, 9, 19, 28, 48, 76, 94, 126, 163, 169] are neither organized nor expressive enough to prevent communication overload [138] and context collapse [128].

Other cryptographic applications, such as the recently proposed *Anix* mesh messaging system [99], provide a mechanism for trusted contacts to anonymously send and vote on messages. While Anix could, in principle, be used to privately annotate digital content, it is designed to operate over mesh networks. We consider the lived settings of both movement *activity* (such as protest) and *abeyance* (suspension), where participants alternate between physical proximity and remote messaging. Moreover, the Anix cryptographic protocols are not proven secure with respect to any formal definition, and prior usable mesh messengers have been proven insecure in practice [7, 54].

Encrypted search. We make use of structured encryption (STE) schemes, which are a broad class of *encrypted search algorithms* (ESAs) capable of achieving a variety of tradeoffs between efficiency, expressiveness, and leakage. Structured encryption was introduced by Chase and Kamara [42] as a generalization of index-based searchable symmetric encryption (SSE) schemes [52, 160]. Our formalization of STE follows that of [105], and we recall its security definitions in Appendix F. The most important kinds of STE schemes are *encrypted dictionaries* and *encrypted multi-maps*. A large number of constructions have been proposed, ranging from schemes that leak nothing at all [24]—not even metadata such as the timing or occurrence of operations—at some cost in performance and correctness, to highly efficient constructions with little to no leakage in weaker adversarial models [10, 105], and to relatively practical constructions that leak very little in stronger adversarial models [13, 72, 73, 104, 106, 147].

In addition to this large body of constructions, a rich line of work has analyzed the leakage of STE schemes both *cryptanalytically* [30, 38, 65, 79, 80, 82, 91, 102, 108, 112–114, 127, 145, 178] and *theoretically* [63, 65, 111]. In contrast, other approaches to designing ESAs—such as property-preserving encryption [5, 26] or trusted execution environments [11]—have received less attention from both cryptanalytic [29, 59, 64, 81, 136] and theoretical [97, 98] perspectives. While oblivious RAM (ORAM) is a possible solution for encrypted search, it provides only a low-level functionality (retrieving a block by address) and, when used as a building block for more complex search primitives, can also incur non-trivial leakage which can be exploited in certain settings [25, 108].[1]

Because of this flexibility, STE has become one of the most widely used primitives for designing practical (and deployable) encrypted search systems, including, for example, encrypted databases [39, 40, 131, 146], decentralized database registries [107] and encrypted analytics systems [62].

Anonymous channels. While there exist provably secure anonymous channels that leverage mix-nets [44, 115], DC-nets [43, 50], and onion routers [14–16], such schemes have not been integrated into real-world systems and therefore do not have the critical mass of users necessary to afford strong anonymity guarantees in practice [58]. Our protocols make black-box use of an ideal anonymous channel functionality, $\mathcal{F}_{\text{Anon}}$. In the current iteration of tigro we implement $\mathcal{F}_{\text{Anon}}$ using the widely-deployed Tor network [58, 148].

3 Preliminaries

We use standard notation, available in Appendix D.

Dictionaries and multi-maps. A dictionary $\mathsf{DX} : \mathbb{L} \to \mathbb{V}$ over a label space $\mathbb{L}$ and value space $\mathbb{V}$ is a collection of label/value pairs $(\ell_i, v_i)_{i \le n}$, where $\ell_i \in \mathbb{L}$ and $v_i \in \mathbb{V}$. Dictionaries typically support Get and Put operations. We write $v_i := \mathsf{DX}[\ell_i]$ to denote getting the value associated with label ℓ_i and $\mathsf{DX}[\ell_i] := v_i$ to denote the operation of putting the value v_i in DX with label ℓ_i. The removal of a pair (ℓ, v) from DX is written as $\mathsf{DX} - (\ell, v)$ or sometimes $\mathsf{DX} - \ell$. A two-dimensional dictionary $\mathsf{DX} : \mathbb{L}_1 \times \mathbb{L}_2 \to \mathbb{V}$ over a label spaces $\mathbb{L}_1$ and $\mathbb{L}_2$ and values space $\mathbb{V}$ is a collection of label/label/value

[1] The line between STE and ORAM is blurry because ORAM can also be viewed as an instance of STE and some STE schemes make use of ORAM as a building block.

Functionality $\mathcal{F}_{\text{prox}}$

$\mathcal{F}_{\text{prox}}$ is parameterized with a distance $\delta \in \mathbb{R}_{\geq 0}$ and identity functions (φ, γ). It interacts with a trusted party $\mathbf{T}$, two clients $\mathbf{C}_a$ and $\mathbf{C}_b$, an adversary $\mathcal{A}$ and manages a location function λ:

- $\text{SetLoc}_T(\lambda)$:
  (1) set $\lambda := \lambda$;
  (2) set $\text{out}_T := \perp$;
  (3) output (out_T);
- $\text{Send}_{\mathbf{C}_a,\mathbf{C}_b}(m; \varphi_\alpha)$:
  (1) if $\mathcal{D}(\lambda_a, \lambda_b) \leq \delta$ and $\varphi_a = \varphi_\alpha$,
    (a) set $\text{leak}_{\mathcal{A}} := (\varphi_a, \varphi_b, m)$;
    (b) output $(\perp; m; \text{leak}_{\mathcal{A}})$;
  (2) else,
    (a) set $\text{leak}_{\mathcal{A}} := (\varphi_a, \varphi_b)$;
    (b) output $(\perp_A; \perp_A; \text{leak}_{\mathcal{A}})$;

Figure 1: Physically-attested proximity channel

triplets $(\ell_{i1}, \ell_{i2}, v_i)_{i \leq n}$, where $\ell_{i_1} \in \mathbb{L}_1$, $\ell_{i_2} \in \mathbb{L}_2$ and $v_i \in \mathbb{V}$. Two-dimensional dictionaries support standard Get and Put operations. We write $v_i := \text{DX}[\ell_{i_1}, \ell_{i_2}]$ to denote getting the value associated with labels ℓ_{i_1} and ℓ_{i_2} and $\text{DX}[\ell_{i_1}, \ell_{i_2}] := v_i$ to denote the operation of putting the value v_i in DX with labels ℓ_{i_1} and ℓ_{i_2}.

A multi-map $\text{MM} : \mathbb{L} \to \mathbb{V}^*$ over a label space $\mathbb{L}$ and value space $\mathbb{V}$ is a collection of label/tuple pairs $(\ell_i, \mathbf{v}_i)_{i \leq n}$ where $\ell_i \in \mathbb{L}$ and each $\mathbf{v}_i \in \mathbb{V}^*$. Multi-maps also support Get and Put operations. We write $\mathbf{v}_i := \text{MM}[\ell_i]$ to denote getting the tuple associated with label ℓ_i and $\text{MM}[\ell_i] := \mathbf{v}_i$ to denote operation of putting the tuple $\mathbf{v}_i$ in MM with label ℓ_i. If a multi-map supports appends we denote by $\text{MM}[\ell] \stackrel{+}{=} v$ the operation that consists of appending the value v to ℓ's tuple. The removal of a pair (ℓ, v) from MM is written as $\text{MM} - (\ell, v)$ or sometimes $\text{MM} - \ell$. A two-dimensional multi-map $\text{MM} : \mathbb{L}_1 \times \mathbb{L}_2 \to \mathbb{V}^*$ over a label spaces $\mathbb{L}_1$ and $\mathbb{L}_2$ and values space $\mathbb{V}$ is a collection of label/label/tuple triplets $(\ell_{i1}, \ell_{i2}, \mathbf{v}_i)_{i \leq n}$, where $\ell_{i_1} \in \mathbb{L}_1$, $\ell_{i_2} \in \mathbb{L}_2$ and $\mathbf{v}_i \in \mathbb{V}$. Two-dimensional multi-maps support standard Get and Put operations. We write $\mathbf{v}_i := \text{MM}[\ell_{i_1}, \ell_{i_2}]$ to denote getting the tuple associated with labels ℓ_{i_1} and ℓ_{i_2} and $\text{MM}[\ell_{i_1}, \ell_{i_2}] := \mathbf{v}_i$ to denote the operation of putting the tuple $\mathbf{v}_i$ in MM with labels ℓ_{i_1} and ℓ_{i_2}.

Identities. In this work we consider a set of n clients $\mathbb{C} = (\mathbf{C}_1, \ldots, \mathbf{C}_n)$, where n is assumed to be independent of the security parameter, and a server $\mathbf{S}$. The notion of identity is central to our framework, as several of the security guarantees we aim for, such as anonymity, seek to hide it, while others, such as grounded trust, depend on it. To capture this, we define two identification functions $\varphi : \mathbb{C} \to \mathcal{P}$ and $\gamma : \mathbb{C} \to \mathcal{I}$ that assign to each client $\mathbf{C}_a$ a physical identifier $\varphi_a \stackrel{\circ}{=} \varphi(\mathbf{C}_a)$ and an institutional identifier $\gamma_a \stackrel{\circ}{=} \gamma(\mathbf{C}_a)$. A client's physical identity can be, for example, its visual identity and its institutional identity can be, for example, its name and address. In each case, we assume the physical and institutional identities are unique and unforgeable. For any client $\mathbf{C}_a$, we further define its identity $\mathbf{i}_a$ as the pair (φ_a, γ_a).

Locations and distances. The concept of proximity is crucial to our notion of groundedness, which requires that participants in a cryptographic protocol be authenticated through a process that includes physical proximity. We capture this by a location function $\lambda : \mathbb{C} \to \mathcal{M}$ that maps each client to a point in a metric space $\mathcal{M}$ with distance function $\mathcal{D}$. We assume locations and distances can be represented in a number of bits that is polynomial in the security parameter. Locations are integrated into our definitions, functionalities, and protocols in various ways, which we describe in Section 4.

Anonymous channels. Our protocols make use of several kinds of channels. The first are private *anonymous channels*, defined in Appendix E, which transmit a message from a sender to a receiver without disclosing the sender's identity $\mathbf{i}_a = (\varphi_a, \gamma_a)$ to the receiver. Given a protocol Π_{P_1,P_2}, we denote by $\mathcal{F}_{\text{anon}} \circ \Pi_{P_1,P_2}$ the execution of Π where P_1 and P_2 send all their messages via $\mathcal{F}_{\text{anon}}$.Send. Note that, throughout, we assume that unless a message is sent through $\mathcal{F}_{\text{anon}}$, the server learns the identity of parties it interacts with.

Proximity channels. At a high level, we will achieve groundedness by requiring the parties to run a key exchange protocol over what we call a *physically-attested proximity channel*. If the channel satisfies the appropriate security properties, the resulting key material can be used as a secure key for a higher-level protocol, ensuring that the protocol is grounded. A physically-attested proximity channel is a special communication channel that meets two critical conditions: (1) a message is only delivered to the receiver if the sender and receiver are in proximity to each other; and (2) a message is only delivered to the receiver if the physical identity of the sender is that of a vetted individual. This mechanism helps ensure that the key exchange occurs in a secure context where both participants are physically present and have physically attested each other. Importantly, we do not consider location forgery: proximity channels are expected to maintain security as long as at least one participant in the exchange is honest. Location-forged exchanges between two malicious participants have no impact on honest participants.

We formally define physical proximity channels in Figure 1 as an ideal functionality $\mathcal{F}_{\text{prox}}$ that is parameterized by a distance $\delta \in \mathbb{R}_{\geq 0}$ and identity functions (φ, γ). Given a message from the sender and an expected physical identifier from the receiver, it checks if the sender and receiver's locations are within distance δ and if the sender's physical identifier matches the expected physical identifier. If this is the case, it delivers the message to the receiver; otherwise, it aborts. Notice that the functionality leaks the physical identifiers of the clients involved and the message but not their institutional identities. This captures settings where a physical adversary in close proximity to the clients can observe the exchange.

Concretely, a proximity channel could be instantiated using any short-range communication protocol like Bluetooth. The challenge with Bluetooth, however, is that it does not provide a way to verify the identity of the sender of a message. In other words, if three parties, A, B, C, are in close proximity and party A receives a message, it has no way to determine whether the message was sent by B or C. Clearly, such a channel would not be strong enough for our purposes since an infiltration adversary in close proximity could subvert the protocol. To address this issue, we propose an alternative approach that leverages the exchange of QR codes using the cameras of mobile devices. This method ensures that the message can only be exchanged when parties are physically close and it incorporates physical attestation since a receiver will only

scan a QR code from a physically vetted sender. QR codes are also accessible, as activists already use them for resource sharing or device linking and safety number verification on Signal. Generating them locally in the context of a single ephemeral key exchange would also prevent known security vulnerabilities surrounding QR code exploitation and surveillance [135, 149].

Given a protocol $\Pi_{\mathbf{C}_a,\mathbf{C}_b}$, we denote by $\mathcal{F}_{\text{prox}}[\varphi_\beta, \varphi_\alpha] \circ \Pi_{\mathbf{C}_a,\mathbf{C}_b}$ the execution of Π such that every message m from $\mathbf{C}_a$ to $\mathbf{C}_b$ is sent via $\mathcal{F}_{\text{prox}}.\text{Send}_{\mathbf{C}_a,\mathbf{C}_b}(\varphi_\beta, m; \varphi_\alpha)$ and every message from $\mathbf{C}_b$ to $\mathbf{C}_a$ is sent via $\mathcal{F}_{\text{prox}}.\text{Send}_{\mathbf{C}_b,\mathbf{C}_a}(\varphi_\alpha, m; \varphi_\beta)$.

Key agreement. Our construction makes use of a key agreement protocol which is a two-party protocol $(K; K) \leftarrow \text{KeyAgree}_{\mathbf{C}_a,\mathbf{C}_b}(1^k; 1^k)$, where K is a k-bit string. A key agreement protocol is secure if it emulates an ideal key agreement functionality that samples a k-bit key uniformly at random and returns it to the parties.

Cryptographic identifiers. We make use of an identifier functionality $\mathcal{F}_{\text{id}}$ to generate globally unique and mutually known and agreed-upon identifiers. The functionality has three operations defined as follows:

- $(\text{id}; \text{id}) \leftarrow \mathcal{F}_{\text{id}}.\text{GetId}_{\mathbf{C}_a,\mathbf{C}_b}(1^k, \Sigma; 1^k, \Sigma)$: the operation takes as input a security parameter 1^k from clients $\mathbf{C}_a$ and $\mathbf{C}_b$ with physical identifiers φ_a and φ_b, respectively, and the name of the scheme for which the identifier is needed. It returns to both clients an identifier id chosen uniformly at random.
- $\text{id} \leftarrow \mathcal{F}_{\text{id}}.\text{GetId}_{\mathbf{C}_a}(1^k, \Sigma)$: the operation takes as input a security parameter 1^k from client $\mathbf{C}_a$, physical identifier φ_a, and the name of the scheme for which the identifier is needed. It returns an identifier id chosen uniformly at random;
- $(\text{id}; \text{id}; \text{id}) \leftarrow \mathcal{F}_{\text{id}}.\text{GetId}_{\mathbf{C}_a,\mathbf{C}_b,\mathbf{S}}(1^k, \Sigma; 1^k, \Sigma; \perp)$: the operation takes as input a security parameter 1^k, the name of the scheme for which the the identifier is needed from $\mathbf{C}_a$ and $\mathbf{C}_b$, and $\perp$ from the server $\mathbf{S}$. It returns to the clients and server an identifier id chosen uniformly at random.

4 Grounded Structured Encryption

We make use of various structured encryption (STE) schemes which can be viewed as reactive multi-party protocols. In this formulation, an STE scheme for an abstract data type $\mathcal{T}$ is a multi-round stateful protocol whose first round executes an initialization protocol and every round thereafter executes an operation protocol on adaptively-generated inputs (i.e., inputs chosen as a function of the state and previous outputs) from the parties and returns new states and outputs. Our constructions are grounded variants of STE schemes which requires us to generalize the security definitions of STE. To describe leakage, we make use of leakage graphs as introduced in [105]. We recall the syntax of STE and the description of leakage graphs in Appendix F.

Adversarial model. We consider semi-honest adversaries that can corrupt the server and observe physical interactions and exchanges between clients. In particular, the information revealed by observing physical interactions is captured and revealed to the adversary through the leakage of the proximity channel which includes the clients' physical identifiers and the messages exchanged.

Formalizing physical locations. As mentioned in Section 3, the notion of groundedness is rooted in physical proximity and is crucial to the security guarantees that tigro is designed to achieve. Therefore, to properly analyze the security of our protocols we need to formally capture physical proximity as a security guarantee. An important assumption we make about physical locations is that they are unforgeable. While this is an obvious property of locations in the real-world, capturing this formally introduces some challenges. One approach to formalizing unforgeable locations is to have the environment associate a location λ_a to every client $\mathbf{C}_a$ and augment the protocols and ideal functionalities to take locations as input. In the setting of semi-honest clients, this might be reasonable because clients will not change their locations. The problem with this approach, however, is that it cannot be extended to the malicious setting since clients could input any location they want. A better approach is the following. First, we augment the proximity channel in the real world and the ideal functionality in the ideal world to hold the locations of all clients. Then we let the environment choose client locations for every operation but have a trusted party $\mathbf{T}$ update the proximity channel in the real world and the ideal functionality in the ideal world with the new locations. This approach allows us to consider environmentally-chosen locations that can change after every operation but that cannot be forged by clients even in the malicious setting.

Grounded real-world execution. The environment $\mathcal{Z}$ and adversary $\mathcal{A}$ take as input a string $z \in \{0,1\}^*$ and a distance $\delta \in \mathbb{R}_{\geq 0}$ and $\mathcal{A}$ corrupts the server $\mathbf{S}$. The environment $\mathcal{Z}$ adaptively chooses a polynomial number of operations of the following form:

- (grounded) a *grounded operation* is an operation of the form

  $$(\varphi, \gamma, \lambda, \mathbf{C}_a, \varphi_\beta, \mathbf{C}_b, \varphi_\alpha, \text{op}\langle\omega_i\rangle, \text{in}_a, \text{in}_b, \text{in}_\mathbf{S}),$$

  where φ and γ determine the clients' physical and institutional identifiers, λ determines the clients' locations, $\mathbf{C}_a$ and $\mathbf{C}_b$ are the clients that execute the grounded operation, φ_β is the physical attribute that $\mathbf{C}_a$ expects (i.e., φ_β is vetted) and φ_α is the physical attribute that $\mathbf{C}_b$ expects (i.e., φ_α is vetted), and in_a, in_b and $\text{in}_\mathbf{S}$ are $\mathbf{C}_a$, $\mathbf{C}_b$ and $\mathbf{S}$'s inputs, respectively. To execute this operation, the trusted party $\mathbf{T}$ first updates the proximity channel $\mathcal{F}_{\text{prox}}$ by executing $\mathcal{F}_{\text{prox}}.\text{SetLoc}(\lambda_i)$ and $\mathbf{C}_a$, $\mathbf{C}_b$, $\mathbf{S}$ and $\mathcal{A}$ execute

  $$(\text{st}_a, \text{out}_a; \text{st}_b, \text{out}_b; \text{st}_\mathbf{S}, \text{out}_\mathbf{S}) \leftarrow \Sigma_\mathcal{T}.\text{Op}\langle\omega_i\rangle(\varphi_\beta, \text{in}_a; \varphi_\alpha, \text{in}_b; \text{in}_\mathbf{S}).$$

  Finally, $\mathbf{C}_a$, $\mathbf{C}_b$, $\mathbf{S}$ and $\mathcal{A}$ send out_a, out_b, $\text{out}_\mathbf{S}$ and a message w to $\mathcal{Z}$, respectively.
- (standard) a *standard operation* is an operation of the form

  $$(\mathbf{C}_a, \text{op}\langle\omega_i\rangle, \text{in}_a, \text{in}_\mathbf{S}),$$

  where $\mathbf{C}_a$ is the client that executes the operation and in_a and $\text{in}_\mathbf{S}$ are $\mathbf{C}_a$ and $\mathbf{S}$'s inputs, respectively. To execute this operation $\mathbf{C}_a$, $\mathbf{S}$ and $\mathcal{A}$ execute

  $$(\text{st}_a, \text{out}_a; \text{st}_\mathbf{S}, \text{out}_\mathbf{S}) \leftarrow \Sigma_\mathcal{T}.\text{Op}\langle\omega_i\rangle(\text{st}_a, \text{in}_a; \text{st}_\mathbf{S}, \perp).$$

  Finally, $\mathbf{C}_a$, $\mathbf{S}$ and $\mathcal{A}$ send out_a, $\text{out}_\mathbf{S}$ and a message w to $\mathcal{Z}$, respectively.

After all the operations are executed, $\mathcal{Z}$ outputs a bit b. We denote the random variable corresponding to b $\mathcal{Z}(\mathcal{A}, \mathbf{T}, \mathbf{C}_1, \ldots, \mathbf{C}_n :: \Sigma_\mathcal{T})$.

Grounded ideal-world execution. In the ideal-world execution, every party has access to an ideal functionality $\mathcal{F}_{\mathscr{J}}^{\mathcal{L}}$ that is parameterized with a stateful leakage profile $\mathcal{L}$ and a distance $\delta \in \mathbb{R}_{\geq 0}$. The environment $\mathcal{Z}$ and simulator $\mathcal{S}$ take as input a string $z \in \{0,1\}^*$ and distance $\delta \in \mathbb{R}_{\geq 0}$. The environment $\mathcal{Z}$ adaptively chooses a polynomial number of operations of the following form:

- (grounded) a grounded operation

$$(\varphi, \gamma, \lambda, \mathbf{C}_a, \varphi_\beta, \mathbf{C}_b, \varphi_\alpha, \mathsf{op}\langle\omega_i\rangle, \mathsf{in}_a, \mathsf{in}_b, \mathsf{in}_{\mathbf{S}}),$$

  defined as above. To execute this operation, the trusted party $\mathbf{T}$ first updates the ideal functionality by executing $\mathcal{F}_{\mathscr{J}}^{\mathcal{L}}.\mathsf{SetLoc}(\lambda_i)$ and $\mathbf{C}_a$, $\mathbf{C}_b$, $\mathbf{S}$ and $\mathcal{S}$ execute $\mathcal{F}_{\mathscr{J}}^{\mathcal{L}}.\mathsf{Op}\langle\omega_i\rangle$ and send out_a, out_b, $\mathsf{out}_{\mathbf{S}}$ and an arbitrary message w_i, respectively, to $\mathcal{Z}$.
- (standard) a standard operation

$$(\mathbf{C}_a, \mathsf{op}\langle\omega_i\rangle, \mathsf{in}_a, \mathsf{in}_b, \mathsf{in}_{\mathbf{S}}),$$

  defined as above. To execute this operation $\mathbf{C}_a$, $\mathbf{S}$ and $\mathcal{S}$ execute $\mathcal{F}_{\mathscr{J}}^{\mathcal{L}}.\mathsf{Op}\langle\omega_i\rangle$ and send out_a, $\mathsf{out}_{\mathbf{S}}$ and an arbitrary message w_i, respectively, to $\mathcal{Z}$.

After all the operations are executed, $\mathcal{Z}$ outputs a bit b. We denote the random variable corresponding to this bit $\mathcal{Z}\left(\mathcal{S}, \mathbf{T}, \mathbf{C}_1, \ldots, \mathbf{C}_n :: \mathcal{F}_{\mathscr{J}}^{\mathcal{L}}\right)$.

Definition 4.1 (security). *We say that* $\Sigma_{\mathscr{J}} = (\mathsf{Op}\langle 0\rangle, \ldots, \mathsf{Op}\langle m\rangle)$ *is* $\mathcal{L}$*-secure if for all* p.p.t. *adversaries* $\mathcal{A}$*, there exists a* p.p.t. *simulator* $\mathcal{S}$ *such that for all* p.p.t. *standalone environments* $\mathcal{Z}$*, for all* $z \in \{0,1\}^*$*, for all* $\delta \in \mathbb{R}_{\geq 0}$*,*

$$\Big|\Pr\left[\,\mathcal{Z}(\mathcal{A}, \mathbf{T}, \mathbf{C}_1, \ldots, \mathbf{C}_n :: \Sigma_{\mathscr{J}}) = 1\,\right] - \Pr\left[\,\mathcal{Z}(\mathcal{S}, \mathbf{C}_1, \ldots, \mathbf{C}_n :: \mathcal{F}_{\mathscr{J}}^{\mathcal{L}}) = 1\,\right]\Big| \leq \mathsf{negl}(k).$$

Note that standard STE security can be recovered by considering only standard operations.

5 Grounded Encrypted Dictionaries

In this section, we show how to construct a grounded encrypted dictionary. Our scheme makes use of a proximity channel $\mathcal{F}_{\mathsf{prox}}$, a key agreement protocol KeyAgree and a standard encrypted dictionary Σ_{DX}. We make two mild assumptions about the encrypted dictionary Σ_{DX} which we now discuss.

Key generation. The first is that its initialization protocol can be decomposed into: (1) a key generation algorithm KeyGen; and (2) an initialization protocol. More precisely, we assume it has syntax $\Sigma_{\mathsf{DX}} = (\mathsf{KeyGen}, \mathsf{Init}, \mathsf{Put}, \mathsf{Get}, \mathsf{Delete})$, where KeyGen and Init work as follows:

- $K \leftarrow \mathsf{KeyGen}_{\mathbf{C}_a}(1^k)$: is an algorithm run by $\mathbf{C}_a$ that samples a k-bit key K uniformly at random;
- $(\mathsf{st}_a; \mathsf{EDX}) \leftarrow \mathsf{Init}_{\mathbf{C},\mathbf{S}}(K, \mathsf{dxid}, \mathbb{L}, \mathbb{V}; \perp)$: is a two-party protocol executed between a client $\mathbf{C}$ and the server $\mathbf{S}$. It takes as input from the client a key K, a dictionary identifier dxid, a label space $\mathbb{L}$, a value space $\mathbb{V}$ and returns client state st_a. It takes as input from the server $\perp$ and returns an encrypted dictionary EDX.

Here, we assume that every operation includes the dictionary identifier dxid in the input. Every encrypted dictionary and multi-map construction we are aware of can be decomposed in this manner.

Concurrency. The second assumption is that Σ_{DX} is concurrent by which we mean that it can handle concurrent Get, Put and Delete operations without affecting its correctness.[2] This can be trivially achieved for any construction as follows. The clients encrypt the scheme's local state under a key derived from the scheme's main key and store it on the server. Before any Get, Put or Delete operation, they first acquire a lock that covers both the encrypted state and the encrypted dictionary, download the encrypted state, decrypt it, execute the operation protocol, encrypt the new updated state, send it to the server and release the lock. A construction with higher throughput could potentially be achieved by using techniques from recent work on concurrent STE [3, 4, 36].

Overview. Our construction, $\Sigma_{\mathsf{GDX}} = (\mathsf{Init}, \mathsf{Put}, \mathsf{Get}, \mathsf{Delete})$ is described in detail in Figure 7 in Appendix G. During the Init protocol, $\mathbf{C}_a$ and $\mathbf{C}_b$ execute KeyAgree over a physically-attested proximity channel in order to generate two k-bit keys. If the physically-vetted key agreement fails, all parties abort. If it succeeds, $\mathbf{C}_a$ uses one of the keys to execute $\Sigma_{\mathsf{DX}}.\mathsf{Init}$ with the server over $\mathcal{F}_{\mathsf{anon}}$ and uses the second key to generate and send an encryption of the dictionary's state to $\mathbf{C}_b$ using $\mathcal{F}_{\mathsf{prox}}$. Put, Get and Delete are the same as in Σ_{DX}, with the exception that all communication with the server $\mathbf{S}$ goes through $\mathcal{F}_{\mathsf{anon}}$.

5.1 Security

We now turn to the security of Σ_{GDX}. Its leakage profile is described in Figure 8 in Appendix G and its security is analyzed in the following Theorem, the proof of which is also in Appendix G.

Theorem 1. *If* Σ_{DX} *is* $\mathcal{L}_{\mathsf{DX}}^{\delta}$*-secure and* KeyAgree *is a secure key agreement protocol,* Σ_{GDX} *as described in Figure 7 is* $\mathcal{L}_{\mathsf{GDX}}^{\delta}$*-secure in the* $(\mathcal{F}_{\mathsf{prox}}, \mathcal{F}_{\mathsf{anon}})$*-hybrid model.*

Discussion. The leakage profile of Σ_{GDX} (Figure 8) is essentially the same as the leakage profile of Σ_{DX}; the difference being that the former's Init protocol reveals the locations, physical identifiers and the expected physical identifiers of $\mathbf{C}_a$ and $\mathbf{C}_b$. The locations and physical identifiers are revealed because the protocol relies on what is effectively a physical key exchange that is observable by corrupted parties in proximity of $\mathbf{C}_a$ and $\mathbf{C}_b$. Finally, we also observe that because Σ_{GDX} makes black-box use of Σ_{DX}, the latter can be instantiated with any encrypted dictionary, including low-leakage or zero-leakage constructions [12, 13, 22, 24, 73, 104, 106, 147].

6 Encrypted Box Banks

An *encrypted box bank* is an encrypted structure that allows multiple pairs of clients to create a shared mailbox in which they can drop labeled messages for each other, retrieve messages by label and delete messages by label. In this section, we describe our encrypted box bank construction. It makes use of a grounded encrypted dictionary and an identifier functionality $\mathcal{F}_{\mathsf{id}}$. It is described in detail in Figures

[2] More precisely, it should achieve a strong notion of consistency under concurrent operations.

10 and 11 in Appendix H and the ideal box bank functionality is described in Figure 9 in Appendix H.

Init. Initialization is an algorithm executed by the server S that initializes a dictionary EBB, and outputs EBB as its state.

CreateBox. The box creation protocol is executed between two clients, $\mathbf{C}_a$ and $\mathbf{C}_b$, and the server S. $\mathbf{C}_a$ and $\mathbf{C}_b$ first use $\mathcal{F}_{\mathsf{id}}$ to generate a mutually-known and agreed upon box identifier bxid. $\mathbf{C}_a$, $\mathbf{C}_b$, and S then run the initialization operation of the underlying grounded encrypted dictionary Σ_{GDX} using bxid as the dictionary identifier dxid. If the operation succeeds, $\mathbf{C}_a$ stores the dictionary state dxst and the box identifier bxid together as the box state bxst. $\mathbf{C}_b$ does the same, setting bxst := (dxst, bxid). Finally, S sets EBB[bxid] := EDX.

DropMail. A client $\mathbf{C}_a$ "drops" mail m with label ℓ in the box with identifier bxid on the server S as follows. First, it sends bxid to S via $\mathcal{F}_{\mathsf{anon}}$. S retrieves the relevant encrypted dictionary EDX := EBB[bxid]. $\mathbf{C}_a$ and S then run the Put operation of the underlying grounded encrypted dictionary Σ_{GDX} on client input (bxid, ℓ, m).

GetMail. To retrieve mail with label ℓ from the box with identifier bxid, a client $\mathbf{C}_a$ sends bxid to S via $\mathcal{F}_{\mathsf{anon}}$. S retrieves the relevant encrypted dictionary EDX := EBB[bxid] and $\mathbf{C}_a$ and S run the Get protocol of Σ_{GDX} on client input (bxid, ℓ). $\mathbf{C}_a$ outputs m.

DeleteMail. Finally to delete mail m with label ℓ from the box with identifier bxid, again a client $\mathbf{C}_a$ sends bxid to S via $\mathcal{F}_{\mathsf{anon}}$. The server S retrieves the encrypted dictionary EDX := EBB[bxid] and $\mathbf{C}_a$ and S run the Delete operation of Σ_{GDX} on client input (bxid, ℓ).

6.1 Security

We now analyze the security of Σ_{BXB}. Its leakage is described in Figure 12 in Appendix H and its security is shown in Theorem 2, the proof of which is also in Appendix H.

Theorem 2. *If Σ_{GDX} is $\mathcal{L}^{\delta}_{\mathsf{GDX}}$-secure, then Σ_{BXB} as described in Figures 10 and 11 is $\mathcal{L}^{\delta}_{\mathsf{BXB}}$-secure in the $(\mathcal{F}_{\mathsf{anon}}, \mathcal{F}_{\mathsf{id}})$-hybrid model.*

Discussion. The leakage profile of Σ_{BXB} (Figure 12) is the same as the leakage profile of Σ_{GDX} with the exception that it also reveals the total number of boxes, which box is accessed during an operation and the creation time and rank of that box (which we denote by bxrank). With this information, the server can learn which boxes are accessed more than others but it cannot learn anything about the contents of the box or the identity of the clients using it. Σ_{BXB} makes black-box use of Σ_{GDX} so the latter can be instantiated with any construction including one that results from instantiating Σ_{DX} with a low-leakage or even zero-leakage encrypted dictionary [12, 13, 22, 24, 73, 104, 106, 147].

7 The tigro Protocol

We now describe our encrypted annotation scheme, tigro. It makes use of a grounded encrypted box bank Σ_{BXB} which can be instantiated with the construction given in Section 6. The scheme is detailed in Figure 2 and Appendix J and works as follows.

Init. The initialization procedure of the annotation scheme is identical to that of the box bank scheme with the exception that each client $\mathbf{C}_i$ also initializes an empty list $\mathsf{Contacts}_i$.

Handshake. The tigro handshake between two clients $\mathbf{C}_a$ and $\mathbf{C}_b$ and the server S consists of the box bank scheme's CreateBox operation. $\mathbf{C}_a$ (resp. $\mathbf{C}_b$) also sets $\mathsf{Contacts}_a[\tau_b]$ (resp. $\mathsf{Contacts}_b[\tau_a]$) to be the box state bxst (resp. bxst_{ba}). The server's process is the same as in the box bank scheme. The groundedness of the handshake and subsequent shared box *roots digital trust in collective physical trust*, adapting activists' existing physical practices into digital space as discussed in Sections 1.1 and 1.3.

Annotate. A client $\mathbf{C}_a$ can send an annotation anno for an object with identifier oid to a list authlist $\subseteq \mathsf{Contacts}_a$ of trusted contacts as follows. First, $\mathbf{C}_a$ retrieves $\mathsf{Contacts}_a$ and, for all $\tau_b \in$ authlist, retrieves the box state bxst := $\mathsf{Contacts}_a[\tau_b]$. $\mathbf{C}_a$ and S then execute the DropMail operation of the underlying box bank scheme Σ_{BXB} on input (bxst, oid, anno). The anno operation satisfies the desired functionality of *vetting digital artifacts* in a way that is directly and efficiently attached to the object being vetted (Section 1.1).

GetAnnotation. To retrieve the annotations associated with an object identifier oid, a client $\mathbf{C}_a$ retrieves $\mathsf{Contacts}_a$ and initializes an empty list Annotations. For all $\tau_b \in \mathsf{Contacts}_a$, $\mathbf{C}_a$ and S execute Σ_{BXB}'s GetMail protocol on client input (bxst, oid). $\mathbf{C}_a$ adds any returned annotations to the list Annotations. The GetAnnotation operation based on dropping rather than pushing mail satisfies the design principle of *information on a need-to-know basis* (Section 1.3).

DeleteAnnotation. To delete an annotation for object identifier oid from the boxes shared with trusted contacts authlist $\subseteq \mathsf{Contacts}_a$, $\mathbf{C}_a$ proceeds as follows. For all $\tau_b \in$ authlist, $\mathbf{C}_a$ retrieves bxst as above, executes Σ_{BXB}'s DeleteMail operation with S on input (bxst, oid), and updates $\mathsf{Contacts}_a[\tau_b]$. Finally, DeleteAnnotation works toward deployment (Section 1.3), though more work is needed to guarantee deletion, which we discuss further in Appendix C.

Efficiency. The communication and server-side computational complexity of tigro are as follows, assuming the dictionary EBB supports $O(1)$ time get operations: Annotate is $O(\#\text{authlist} \cdot \mathsf{cost}(\Sigma_{\mathsf{DX}}.\mathsf{Put}(\mathsf{oid}, m)))$, GetAnnotation is $O(\#\mathsf{Contacts} \cdot \mathsf{cost}(\Sigma_{\mathsf{DX}}.\mathsf{Get}(\mathsf{oid})))$, and DeleteAnnotation is $O(\#\text{authlist} \cdot \mathsf{cost}(\Sigma_{\mathsf{DX}}.\mathsf{Delete}(\mathsf{oid})))$, where $\mathsf{cost}(\pi)$ is the asymptotic cost in communication or server-side computation of protocol π.

7.1 Security

We now analyze the security of our construction. Its leakage is described in Figure 14 in Appendix J and its security is shown in the following Theorem, the proof of which is in Appendix I.

Theorem 3. *If Σ_{BXB} is $\mathcal{L}^{\delta}_{\mathsf{BXB}}$-secure, then* tigro *as described in Figure 2 is $\mathcal{L}^{\delta}_{\mathsf{tigro}}$-secure in the $\mathcal{F}_{\mathsf{anon}}$-hybrid model.*

Discussion. The leakage profile of Σ_{tigro} (Figure 14) reveals, in addition to the leakage of Σ_{BXB}, the sizes of the authlist when storing and deleting annotations and the size of the client's contact list when retrieving an annotation. As was the case for Σ_{GDX} and Σ_{BXB}, tigro makes black-box use of Σ_{BXB}.

Instantiations. Because our tigro construction and analysis is black-box, it can be instantiated in a variety of ways depending on which encrypted dictionary is used. As described in [24], encrypted

The tigro Protocol

Let Σ_{BXB} = (Init, CreateBox, DropMail, GetMail, DeleteMail) be a grounded box bank scheme and consider the annotation protocol tigro = (Init, Handshake, Annotate, GetAnnotation, DeleteAnnotation) defined as follows:

$\mathsf{Init}_{\mathsf{S}}(\perp)$:

(1) S execute EBB $\leftarrow \Sigma_{\mathsf{BXB}}.\mathsf{Init}_{\mathsf{S}}(\perp)$;
(2) output EBB;

$\mathsf{Handshake}_{\mathsf{C}_a,\mathsf{C}_b,\mathsf{S}}(\tau_a, \varphi_\beta; \tau_b, \varphi_\alpha; \mathsf{EBB})$:

(1) $\mathbf{C}_a$, $\mathbf{C}_b$ and S execute $(\mathsf{out}_a; \mathsf{out}_b; \mathsf{out}_{\mathsf{S}}) \leftarrow \Sigma_{\mathsf{BXB}}.\mathsf{CreateBox}_{\mathsf{C}_a,\mathsf{C}_b,\mathsf{S}}(\tau_a, \varphi_\beta, \mathbb{L}, \mathbb{V}; \tau_b, \varphi_\alpha; \mathsf{EBB})$;
(2) if $\mathsf{out}_a = \mathsf{out}_b = \mathsf{out}_{\mathsf{S}} = \perp_A$,
  (a) output $(\perp_A; \perp_A; \perp_A)$;
(3) else,
  (a) $\mathbf{C}_a$ parses out_a as $(\mathsf{bxst}_a, K_{\mathsf{BXB}}, \mathsf{bxid})$;
  (b) if $\mathsf{Contacts}_a = \perp$,
    (i) $\mathbf{C}_a$ initializes a dictionary $\mathsf{Contacts}_a$;
  (c) $\mathbf{C}_a$ sets $\mathsf{Contacts}_a[\tau_b] := (\mathsf{bxst}_a, K_{\mathsf{BXB}}, \mathsf{bxid})$;
  (d) $\mathbf{C}_a$ sets $\mathsf{tgst}_a := \mathsf{Contacts}_a$;
  (e) $\mathbf{C}_b$ parse out_b as $(\mathsf{bxst}_b, K_{\mathsf{BXB}}, \mathsf{bxid})$;
  (f) if $\mathsf{Contacts}_b = \perp$,
    (i) $\mathbf{C}_b$ initializes a dictionary $\mathsf{Contacts}_b$;
  (g) $\mathbf{C}_b$ sets $\mathsf{Contacts}_b[\tau_a] := (\mathsf{bxst}_b, K_{\mathsf{BXB}}, \mathsf{bxid})$;
  (h) $\mathbf{C}_b$ sets $\mathsf{tgst}_b := \mathsf{Contacts}_b$;
  (i) output $(\mathsf{tgst}_a; \mathsf{tgst}_b; \mathsf{EBB})$

$\mathsf{Annotate}_{\mathsf{C}_a,\mathsf{S}}(\mathsf{tgst}_a, \mathsf{oid}, \mathsf{anno}, \mathsf{authlist}; \mathsf{EBB})$:

(1) $\mathbf{C}_a$ parses tgst_a as $\mathsf{Contacts}_a$;
(2) for all $\tau_b \in \mathsf{authlist}$,
  (a) $\mathbf{C}_a$ computes $(\mathsf{bxst}, K_{\mathsf{BXB}}, \mathsf{bxid}) := \mathsf{Contacts}_a[\tau_b]$;
  (b) $\mathbf{C}_a$ and S execute $(\mathsf{bxst}; \mathsf{EBB}) \leftarrow \Sigma_{\mathsf{BXB}}.\mathsf{DropMail}_{\mathsf{C}_a,\mathsf{S}}(\mathsf{bxst}, K_{\mathsf{BXB}}, \mathsf{oid}, \mathsf{anno}; \mathsf{EBB})$;
  (c) $\mathbf{C}_a$ sets $\mathsf{Contacts}_a[\tau_b] := (\mathsf{bxst}, K_{\mathsf{BXB}}, \mathsf{bxid})$;
(3) $\mathbf{C}_a$ sets $\mathsf{tgst} := \mathsf{Contacts}_a$;
(4) output $(\mathsf{tgst}; \mathsf{EBB})$;

$\mathsf{GetAnnotation}_{\mathsf{C}_a,\mathsf{S}}(\mathsf{tgst}, \mathsf{oid}; \mathsf{EBB})$:

(1) $\mathbf{C}_a$ parses tgst as $\mathsf{Contacts}_a$;
(2) $\mathbf{C}_a$ sets $\mathbf{anno} := \emptyset$;
(3) for all $\tau_b \in \mathsf{Contacts}_a$,
  (a) $\mathbf{C}_a$ computes $(\mathsf{bxst}, K_{\mathsf{BXB}}, \mathsf{bxid}) := \mathsf{Contacts}_a[\tau_b]$;
  (b) $\mathbf{C}_a$ and S execute $(\mathsf{bxst}, \mathsf{anno}; \mathsf{EBB}) \leftarrow \Sigma_{\mathsf{BXB}}.\mathsf{GetMail}_{\mathsf{C}_a,\mathsf{S}}(\mathsf{bxst}, K_{\mathsf{BXB}}, \mathsf{oid}; \mathsf{EBB})$;
  (c) $\mathbf{C}_a$ sets $\mathbf{anno} := \mathbf{anno} \cup \mathsf{anno}$;
  (d) $\mathbf{C}_a$ sets $\mathsf{Contacts}_a[\tau_b] := (\mathsf{bxst}, K_{\mathsf{BXB}}, \mathsf{bxid})$;
(4) $\mathbf{C}_a$ sets $\mathsf{tgst} := \mathsf{Contacts}_a$;
(5) output $(\mathsf{tgst}, \mathbf{anno}; \mathsf{EBB})$;

$\mathsf{DeleteAnnotation}_{\mathsf{C}_a,\mathsf{S}}(\mathsf{tgst}, \mathsf{oid}, \mathsf{authlist}; \mathsf{EBB})$:

(1) $\mathbf{C}_a$ parses tgst as $\mathsf{Contacts}_a$;
(2) for all $\tau_b \in \mathsf{authlist}$,
  (a) $\mathbf{C}_a$ computes $(\mathsf{bxst}, K_{\mathsf{BXB}}, \mathsf{bxid}) := \mathsf{Contacts}_a[\tau_b]$;
  (b) $\mathbf{C}_a$ and S execute $(\mathsf{bxst}; \mathsf{EBB}) \leftarrow \Sigma_{\mathsf{BXB}}.\mathsf{DeleteMail}_{\mathsf{C}_a,\mathsf{S}}(\mathsf{bxst}, K_{\mathsf{BXB}}, \mathsf{oid}; \mathsf{EBB})$;
  (c) $\mathbf{C}_a$ sets $\mathsf{Contacts}_a[\tau_b] := (\mathsf{bxst}, K_{\mathsf{BXB}}, \mathsf{bxid})$;
(3) $\mathbf{C}_a$ sets $\mathsf{tgst} := \mathsf{Contacts}_a$;
(4) output $(\mathsf{tgst}; \mathsf{EBB})$;

Figure 2: The tigro protocol.

search algorithms can be categorized into several types based on the amount and type of information they leak:

- *high-leakage*: these are constructions that reveal non-trivial information about the data and/or queries from just the encrypted data itself. Such a scheme would not mitigate the threats of *data confidentiality* discussed in Section 1.2.
- *mid-leakage*: these schemes offer the best efficiency but can reveal non-trivial information about the queries, such as the *query equality pattern* (i.e., whether two queries are the same) and the *volume pattern* (i.e., the number of items returned by a query). Some examples of mid-leakage schemes include [39, 41, 52, 68, 69, 77]. There are known leakage attacks against the query equality [102, 144] and volume patterns [30, 79, 80, 108] but whether they are applicable in practice is nuanced and depends on the application scenario, data type, and a variety of assumptions about the data and queries; we refer the reader to papers themselves for further discussions. We note that since queries in our system are derived from public URLs and no data is revealed about the annotations themselves, mid-leakage schemes are likely sufficient to satisfy data confidentiality for typical use. We additionally discuss (and later realize) lower-leakage schemes.
- *low-leakage*: these schemes generally hide both the query equality and volume patterns but may still reveal information like the total number of items returned by a sequence of queries. There are no known leakage attacks against these leakage profiles and, intuitively speaking, they seem very difficult to exploit. Some examples of low-leakage constructions include [22, 71, 106, 106].
- *zero-leakage*: these schemes typically only reveal the size of the data structure and the maximum volume over all possible queries. As in the case of low-leakage constructions, there are no known attacks against these leakage profiles and they seem even more difficult to exploit. An example of such a construction is the FZL scheme of [106].
- *subliminal leakage*: these constructions reveal no information beyond the existence of the structure itself. They even hide metadata such as the number of operations, the timing of operations, and whether operations occurred at all. An example of a subliminal scheme is the EXH construction from [24], which can be instantiated in one of two modes: a low-leakage mode, which guarantees full correctness by ensuring that the entire response associated with a query will be returned, and a subliminal mode, which cannot guarantee that a complete response will always be provided. A subliminal scheme might be preferable for activists in extremely repressive environments, where any information about system usage could be leveraged against them.

We construct two tigro instantiations: one based on a mid-leakage encrypted dictionary construction (Section 8) and one based on a subliminal construction (Appendix K).

8 An Efficient Instantiation

We describe and analyze a concrete instantiation of the tigro protocol based on a variant of highly efficient dynamic encrypted dictionary construction called π_{dyn} [39]. We denote the variant we

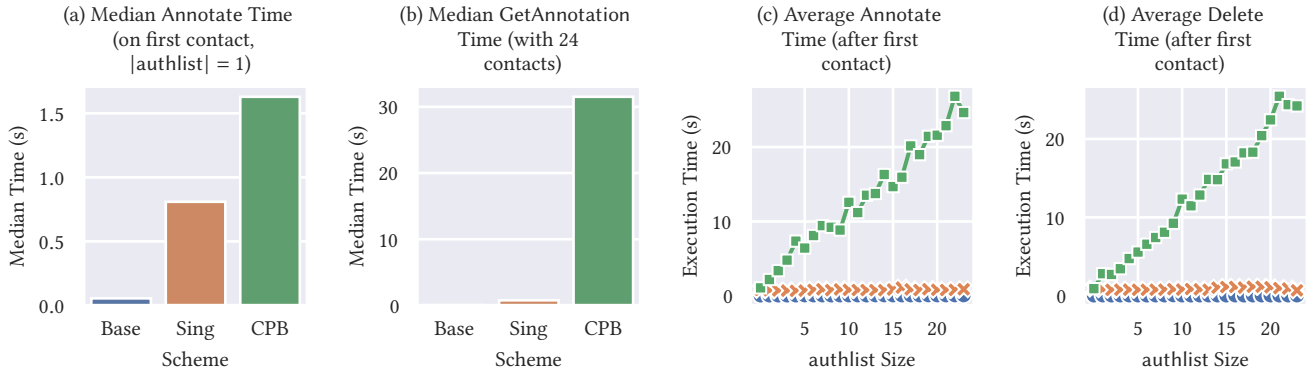


Figure 3: Evaluation results of Tigro as described in Section 8. *Base* (-●-) denotes the baseline client with no Tor, *Sing* (-✕-) denotes the single-Tor-circuit client, and *CPB* (-■-) denotes the client that uses a unique circuit per box.

use by π^*_{bas} and note that it is simpler than the original construction and achieves a similar leakage profile. Technically, π^*_{bas} is a multi-map encryption scheme, which generalizes encrypted dictionaries; specifically, an encrypted dictionary is an encrypted multi-map with single-element tuples.

π^*_{bas} details. The scheme $\pi^*_{\text{bas}} = (\text{Init}, \text{Put}, \text{Get}, \text{Delete})$ is a semi-dynamic multi-map encryption scheme that utilizes a pseudo-random function F and a symmetric encryption scheme $\text{SKE} = (\text{Gen}, \text{Enc}, \text{Dec})$. To initialize an encrypted multi-map, the client samples a key $K_F \leftarrow\$ \{0,1\}^k$ for F and generates a symmetric encryption key $K_E \leftarrow \text{Gen}(1^k)$. It then initializes a state dictionary sDX and an empty dictionary DX, which it sends to the server. To put a label/tuple pair $(\ell, \mathbf{v})$, where $\mathbf{v} = (v_1, \ldots, v_m)$, the client computes $\text{ctr}_\ell \leftarrow \text{sDX}[\ell]$, sets $\text{ctr}_\ell = 0$ if $\text{ctr}_\ell = \perp$, and sends the m pairs

$$\left(\text{tag}_i, \text{ct}_i\right)_{i\in[m]} = \left(F\left(F\left(K_F, \ell\right), \text{ctr} + i\right), \text{Enc}_{K_E}(v_i)\right)_{i\in[m]}$$

to the server, which the server inserts into DX. The client then sets $\text{sDX}[\ell] := \text{ctr} + 1$. To Get the tuple associated with a label ℓ, the client sends $K_\ell := F(K_F, \ell)$ to the server, which retrieves and returns $\text{ct}_1 := \text{DX}[F(K_\ell, 1)]$, $\text{ct}_2 := \text{DX}[F(K_\ell, 2)]$, and so on until $\text{DX}[F(K_\ell, n)] = \perp$. To Delete the tuple associated with a label ℓ, the client sends $K_\ell := F(K_F, \ell)$ to the server, which deletes all the pairs with labels $F(K_\ell, 1)$, $F(K_\ell, 2)$ and so on until $\text{DX}[F(K_\ell, n)] = \perp$.

Efficiency analysis. The tigro$[\pi^*_{\text{bas}}]$ instantiation requires communication and server-side computational complexity $O(\#\text{authlist})$, $O(\#\text{Contacts})$ and $O(\#\text{authlist})$ for Annotate, GetAnnotation and DeleteAnnotation operations, respectively. This follows from the fact that π^*_{bas} has $O(1)$ communication and server-side computation for gets, puts and deletes when used as a dictionary.

Concrete leakage. The leakage profile of tigro$[\pi^*_{\text{bas}}]$ is described in detail in Figures 15 and 16 in Appendix J. At a high level, there are three types of information revealed:

- *sizes*: the server learns the number and size of the boxes.
- *correlations*: the server learns if and when a user searches for, adds or deletes the same oid on a per-box basis (i.e., not across boxes). If a user searches for an oid but only *later* annotates that oid, the server will be able to correlate the new annotation with the previous search.
- *social network topology*: the server learns which boxes are co-owned by the same client and since a box is owned by two clients, this also reveals the topology of the community's social network.
- *metadata*: the server learns how many times a box has been accessed (i.e., searched, written to or delete from) and the times at which it was accessed.

The correlations that are revealed are known as the *operation equality* pattern which includes the *query equality pattern* and the *put-get equality pattern*. State-of-the-art attacks against the query equality pattern were described in [102]. These attacks are query inference attacks that require the adversary to know the user's query distribution. The put-get equality can lead to adaptive injection attacks [30, 178] assuming the server can inject data/messages into the encrypted structure which, here, is the encrypted box. As with all leakage attacks, evaluating the real-world impact of attacks in practice is quite nuanced so we refer the reader to the papers and to the survey [103] for a more complete discussion.

Impact of correlation leakage. Although tigro$[\pi^*_{\text{bas}}]$ reveals correlations, they are less problematic in our setting than they might initially appear. This is because, in the context of tigro, the correlation leakage is restricted to the oids. So, in the worst case, any leakage attack would recover the oid and could potentially use it to infer information about the associated object. However, it's important to note that the objects themselves are typically public (e.g., digital flyers, social network profiles, etc.), and in some cases, they might even be adversarially created and sent to the client. What is crucial to understand is that tigro does not use any of the encrypted structures to index the annotations themselves so, as a result, none of the leakage produced by the protocol pertains to the annotations. In other words, *regardless of how* tigro *is instantiated, it will not leak any information about the annotations.*

Impact of injection attacks. Similar to the case of correlation leakage, injection attacks are not as problematic in our setting as they might seem at first glance. This is, again, because tigro does not index the annotations and does not support searching over them. Therefore, even if an adversary manages to inject annotations into a

box, it will not be able use this to learn any meaningful information about the annotations themselves.

Impact of social network and metadata leakage. In some settings, leakage of the social network topology could pose a risk, as it may reveal, for example, that certain users are highly connected within the community. While their identities remain protected, an adversary could exploit this to target well-connected users' boxes through denial-of-service (DoS) or other attacks. Similarly, metadata about box access could be leveraged to target more active boxes.

For environments where even network topology and metadata leakage may be too revealing, Appendix K discusses how to instantiate tigro with the recently proposed metadata-hiding encrypted multi-map from [24]. This results in an instantiation that conceals such information at the cost of additional complexity and overhead.

8.1 Empirical Evaluation

To determine the baseline performance of tigro in practice, we implemented tigro$[\pi^*_{\text{bas}}]$ using the Tor network [123] as the anonymous channel $\mathcal{F}_{\text{Anon}}$. Concretely, we implemented the tigro client in Python 3.11 and used the `stem` Python library which provides Tor networking primitives and the `arca` Python library [66] which provides cryptographic primitives for π^*_{bas}. For the tigro server, we use Redis as the backing storage layer. We evaluated three different networking implementations of the tigro client:[3]

- *Base:* baseline implementation with no Tor networking.
- *Sing:* the tigro client uses a single Tor circuit for all requests.
- *CPB* (circuit per box): the tigro client uses a unique Tor circuit for each box.

We single-threaded our client implementation, though we note that for *CPB*, it is theoretically possible to concurrently execute all of the necessary Tor requests at once. We additionally staggered our evaluation across multiple days to minimize the concurrent load our evaluation imposed on the Tor network.

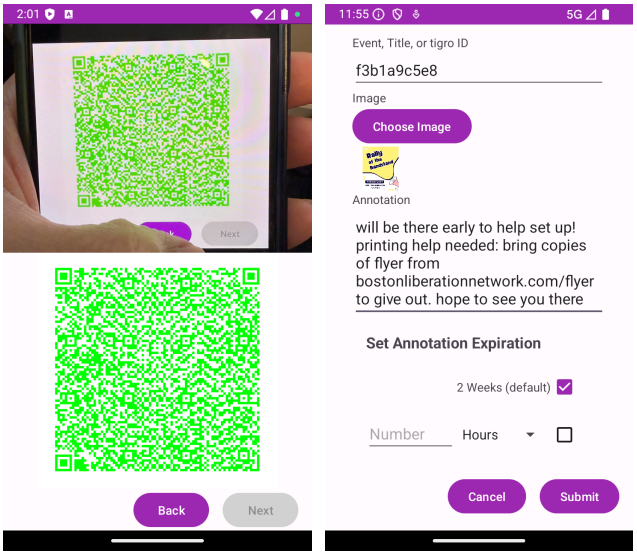
Experimental setup. Our experiments run on Amazon EC2. The tigro server and its Redis storage engine run on a single `m6g.4xlarge` instance. The tigro clients run on a `m6g.4xlarge` instance in the same zone. In each experiment, we set up 25 tigro clients and simulate Handshakes (normally in person) between all of the clients. Each tigro client's contact list therefore contains 24 contacts.

First contact. In Figure 3 (a), we measure the median "first contact" execution time of Annotate operations—the time that it takes for a client to Annotate to a box that they have not yet accessed in the tigro client session. We find that first contact Annotate operations on *Sing* take 0–14× longer than Annotate operations on the baseline *Base* implementation. This is expected as the use of Tor imposes significant latency on network requests. We additionally find that first contact Annotate operations are 2× slower on *CPB* than in *Sing*, which is because *CPB*'s first contact operations have to build new Tor circuits for the newly accessed box.

After first contact. In Figure 3 (b–d), we measure the execution times of Annotate, GetAnnotation, and DeleteAnnotation after the client has already accessed the involved boxes previously in the tigro client session. For Annotate and DeleteAnnotation, we change the size of authlist (the number of boxes affected by the operation) and record the execution time by authlist size. For GetAnnotation, the runtime stays relatively constant regardless of the number of retrieved annotations as the execution time is dominated by the size of a tigro client's contact list. As a result, we instead report the median GetAnnotation time in Figure 3 (b).

Across all operations, we find that *Sing* is roughly 9 – 16× slower than *Base*. This is expected due to the latency imposed by Tor. We also find that *CPB* is up to 33× slower than *Sing*. This is because the number of round trips required by *CPB* is linear in the number of boxes accessed in the given operation, whereas the use of a single Tor circuit in *Sing* means that all of the information needed for box access can be sent in a single round trip across the one circuit.

[3]We considered other Tor variants (e.g. generating new circuits more frequently to minimize correlation leakage). However, we did not evaluate these to avoid imposing high load on the Tor network. See "Ethics Considerations" for details.



(a) People meet in person to exchange ephemeral public keys via QR codes and phone cameras. **(b)** tigro**'s "share" functionality allows participants to annotate digital artifacts to known** oids**.**

Figure 4: Example screenshots from the tigro **application.**

8.2 Application Prototype

We implement the tigro$[\pi^*_{\text{bas}}]$ efficient construction as the tigro application prototype. The tigro client prototype was built in Android Studio using Kotlin as the primary programming language, while the server was implemented using C++ and Protobuf. The client prototype can be run on an emulator or a physical Android device. Figure 4 depicts example screenshots of the application. The client prototype includes additional usability features such as locally-stored contact lists, note-taking functionality, sharing, and group annotations. We discuss further considerations for the application prototype in Appendix C and ethical considerations in Appendix L.

Acknowledgments

Leah Namisa Rosenbloom was supported materially and spiritually by comrades near and far, who continue to inspire this work. They did this work in part at Brown University. John Wilkinson was supported by funding from the Taubman Center for American Politics and Policy and the Center for Technological Responsibility, Reimagination, and Redesign at Brown University. The remaining authors were not supported by any other grants from funding agencies in the public, commercial, or not-for-profit sectors.

References

[1] Ruba Abu-Salma, Kat Krol, Simon Parkin, Victoria Koh, Kevin Kwan, Jazib Mahboob, Zahra Traboulsi, and M Angela Sasse. 2017. The security blanket of the chat world: An analytic evaluation and a user study of telegram. Internet Society.

[2] Ruba Abu-Salma, M Angela Sasse, Joseph Bonneau, Anastasia Danilova, Alena Naiakshina, and Matthew Smith. 2017. Obstacles to the adoption of secure communication tools. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 137–153.

[3] Archita Agarwal and Zachary Espiritu. 2025. Sequentially Consistent Concurrent Encrypted Multimaps. In *2025 IEEE 10th European Symposium on Security and Privacy (EuroS&P)*. IEEE, IEEE.

[4] Archita Agarwal, Seny Kamara, and Tarik Moataz. 2025. Concurrent Encrypted Multimaps. In *Advances in Cryptology – ASIACRYPT 2024*, Kai-Min Chung and Yu Sasaki (Eds.). Springer Nature Singapore, Singapore, 169–201.

[5] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. 2004. Order preserving encryption for numeric data. In *ACM SIGMOD International Conference on Management of Data*. 563–574.

[6] Martin R. Albrecht, Jorge Blasco, Rikke Bjerg Jensen, and Lenka Mareková. 2021. Collective Information Security in Large-Scale Urban Protests: the Case of Hong Kong. (Aug. 2021), 3363–3380. https://www.usenix.org/conference/usenixsecurity21/presentation/albrecht

[7] Martin R Albrecht, Jorge Blasco, Rikke Bjerg Jensen, and Lenka Mareková. 2021. Mesh messaging in large-scale protests: Breaking Bridgefy. In *Cryptographers' Track at the RSA Conference*. Springer, 375–398.

[8] Nezar AlSayyad and Muna Guvenc. 2015. Virtual uprisings: On the interaction of new social media, traditional media coverage and urban space during the 'Arab Spring'. *Urban Studies* 52, 11 (2015), 2018–2034.

[9] Joël Alwen, Sandro Coretti, and Yevgeniy Dodis. 2019. The double ratchet: security notions, proofs, and modularization for the signal protocol. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 129–158.

[10] Ghous Amjad, Seny Kamara, and Tarik Moataz. 2019. Breach-Resistant Structured Encryption. In *Proceedings on Privacy Enhancing Technologies (Po/PETS '19)*.

[11] Ghous Amjad, Seny Kamara, and Tarik Moataz. 2019. Forward and Backward Private Searchable Encryption with SGX. In *Proceedings of the 12th European Workshop on Systems Security* (Dresden, Germany) *(EuroSec '19)*. Association for Computing Machinery, New York, NY, USA, Article 4, 6 pages. https://doi.org/10.1145/3301417.3312496

[12] Ghous Amjad, Seny Kamara, and Tarik Moataz. 2023. Injection-secure structured and searchable symmetric encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 232–262.

[13] Ghous Amjad, Sarvar Patel, Giuseppe Persiano, Kevin Yeo, and Moti Yung. 2021. Dynamic Volume-Hiding Encrypted Multi-Maps with Applications to Searchable Encryption. *Proceedings of Privacy Enhancing Technologies* 2023 (2021), 417–436. Issue 1. https://doi.org/10.56553/popets-2023-0025

[14] Megumi Ando, Miranda Christ, Anna Lysyanskaya, and Tal Malkin. 2022. Poly onions: Achieving anonymity in the presence of churn. In *Theory of Cryptography Conference*. Springer, 715–746.

[15] Megumi Ando, Anna Lysyanskaya, and Eli Upfal. 2018. Practical and Provably Secure Onion Routing. 107 (2018), 144:1–144:14. https://doi.org/10.4230/LIPIcs.ICALP.2018.144

[16] Megumi Ando, Anna Lysyanskaya, and Eli Upfal. 2024. Bruisable Onions: Anonymous Communication in the Asynchronous Model. In *Theory of Cryptography Conference*. Springer, 476–507.

[17] Eva Anduiza, Camilo Cristancho, and José M Sabucedo. 2014. Mobilization through online social networks: the political protest of the indignados in Spain. *Information, communication & society* 17, 6 (2014), 750–764.

[18] Miriyam Aouragh, Seda Gürses, Jara Rocha, and Femke Snelting. 2015. FCJ-196 Let's First Get Things Done! On Division of Labour and Techno-political Practices of Delegation in Times of Crisis. *The Fibreculture Journal* 26 2015: Entanglements–Activism and Technology (2015).

[19] Chinmayi Arun. 2019. On WhatsApp, rumours, lynchings, and the Indian Government. *Economic & Political Weekly* 54, 6 (2019).

[20] Gilad Asharov, Ilan Komargodski, Wei-Kai Lin, Kartik Nayak, Enoch Peserico, and Elaine Shi. 2020. OptORAMa: optimal oblivious RAM. In *Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part II 30*. Springer, 403–432.

[21] Tina Askanius and Julie Uldam. 2011. Online social media for radical politics: climate change activism on YouTube. *International journal of electronic governance* 4, 1-2 (2011), 69–84.

[22] Léonard Assouline and Brice Minaud. 2023. Weighted Oblivious RAM, with Applications to Searchable Symmetric Encryption. In *Advances in Cryptology – EUROCRYPT 2023*, Carmit Hazay and Martijn Stam (Eds.). Springer Nature Switzerland, Cham, 426–455.

[23] Evronia Azer, G Harindranath, and Yingqin Zheng. 2019. Revisiting leadership in information and communication technology (ICT)-enabled activism: A study of Egypt's grassroots human rights groups. *New Media & Society* 21, 5 (2019), 1141–1169.

[24] Amine Bahi, Seny Kamara, Tarik Moataz, and Guevara Noubir. 2024. Subliminal Encrypted Multi-Maps and Black-Box Leakage Absorption. Cryptology ePrint Archive, Paper 2024/1708. https://eprint.iacr.org/2024/1708

[25] L. Ballard, S. Kamara, and F. Monrose. 2005. Achieving Efficient Conjunctive Keyword Searches over Encrypted Data. In *Seventh International Conference on Information and Communication Security (ICICS '05) (Lecture Notes in Computer Science, Vol. 3783)*, S. Qing, W. Mao, J. Lopez, and G. Wang (Eds.). Springer, 414–426.

[26] M. Bellare, A. Boldyreva, and A. O'Neill. 2007. Deterministic and Efficiently Searchable Encryption. In *Advances in Cryptology – CRYPTO '07 (Lecture Notes in Computer Science)*, A. Menezes (Ed.). Springer, 535–552.

[27] Ruha Benjamin. 2019. Race after technology: Abolitionist tools for the new jim code. *Social Forces* (2019).

[28] Richard Bieringa, Abijith Radhakrishnan, Tavneet Singh, Sophie Vos, Jesse Donkervliet, and Alexandru Iosup. 2021. An empirical evaluation of the performance of video conferencing systems. In *Companion of the ACM/SPEC International Conference on Performance Engineering*. 65–71.

[29] Vincent Bindschaedler, Paul Grubbs, David Cash, Thomas Ristenpart, and Vitaly Shmatikov. 2018. The tao of inference in privacy-protected databases. *Proceedings of the VLDB Endowment* 11, 11 (2018), 1715–1728.

[30] Laura Blackstone, Seny Kamara, and Tarik Moataz. 2020. Revisiting Leakage Abuse Attacks. In *Network and Distributed System Security Symposium (NDSS '20)*.

[31] Tetyana Bohdanova. 2014. Unexpected revolution: the role of social media in Ukraine's Euromaidan uprising. *European View* 13, 1 (2014), 133–142.

[32] Glencora Borradaile. 2021. *Defend Dissent.* Oregon State University Corvallis.

[33] Glencora Borradaile, Kelsy Kretschmer, Michele Gretes, and Alexandria LeClerc. 2021. The Motivated Can Encrypt (Even with PGP). *Proceedings of Privacy Enhancing Technologies* 2021 (2021), 1–21. Issue 3. https://doi.org/10.2478/popets-2021-0000

[34] Jules Boykoff. 2007. Limiting dissent: The mechanisms of state repression in the USA. *Social Movement Studies* 6, 3 (2007), 281–310.

[35] Simone Browne. 2015. *Dark matters: On the surveillance of blackness*. Duke University Press.

[36] Théophile Brézot and Chloé Hébant. 2024. Findex: A Concurrent and Database-Independent Searchable Encryption Scheme. https://eprint.iacr.org/2024/1541 Cryptography ePrint Archive, Paper 2024/1541.

[37] Joy Buolamwini and Timnit Gebru. 2018. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*. PMLR, 77–91.

[38] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart. 2015. Leakage-Abuse Attacks against Searchable Encryption. In *ACM Conference on Communications and Computer Security (CCS '15)*. ACM, 668–679.

[39] David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, Marcel Rosu, and Michael Steiner. 2014. Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation. In *Network and Distributed System Security Symposium (NDSS '14)*.

[40] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu, and M. Steiner. 2013. Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries. In *Advances in Cryptology - CRYPTO '13*. Springer.

[41] Y. Chang and M. Mitzenmacher. 2005. Privacy Preserving Keyword Searches on Remote Encrypted Data. In *Applied Cryptography and Network Security (ACNS '05) (Lecture Notes in Computer Science, Vol. 3531)*. Springer, 442–455.

[42] M. Chase and S. Kamara. 2010. Structured Encryption and Controlled Disclosure. In *Advances in Cryptology - ASIACRYPT '10 (Lecture Notes in Computer Science, Vol. 6477)*. Springer, 577–594.

[43] David Chaum. 1988. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of cryptology* 1 (1988), 65–75.

[44] David L Chaum. 1981. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM* 24, 2 (1981), 84–90.

[45] Henrik Serup Christensen. 2011. Political activities on the Internet: Slacktivism or political participation by other means? *First Monday* (2011).

[46] Ward Churchill and Jim Vander Wall. 1990. The COINTELPRO papers. *Boston: South End* (1990).

[47] Jennifer Cobbe. 2021. Algorithmic censorship by social platforms: Power and resistance. *Philosophy & Technology* 34, 4 (2021), 739–766.

[48] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. 2020. A formal security analysis of the signal messaging protocol. *Journal of Cryptology* 33 (2020), 1914–1983.

[49] Ted M Coopman. 2011. Networks of dissent: Emergent forms in media based collective action. *Critical studies in media communication* 28, 2 (2011), 153–172.

[50] Henry Corrigan-Gibbs and Bryan Ford. 2010. Dissent: accountable anonymous group messaging. In *Proceedings of the 17th ACM conference on Computer and communications security*. 340–350.

[51] Sasha Costanza-Chock. 2020. *Design justice: Community-led practices to build the worlds we need*. The MIT Press.

[52] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. 2006. Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. In *ACM Conference on Computer and Communications Security (CCS '06)*. ACM, 79–88.

[53] Alaa Daffalla, Lucy Simko, Tadayoshi Kohno, and Alexandru G Bardas. 2021. Defensive technology use by political activists during the Sudanese revolution. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 372–390.

[54] Jakub Dalek, Philipp Winter, Andrei Dranka, Masashi Crete-Nishihata, and Adam Senft. 2014. Asia Chats: Update on Line, KakaoTalk, and FireChat in China. *The Citizen Lab* (2014).

[55] Tyler DeAtley. 2019. Mobile Ambivalence at Standing Rock: surveillance, antagonism, and mobility at the Dakota Access Pipeline protests. *Journal of Resistance Studies* 2 (2019).

[56] Sergej Dechand, Alena Naiakshina, Anastasia Danilova, and Matthew Smith. 2019. In encryption we don't trust: The effect of end-to-end encryption to the masses on user perception. In *2019 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 401–415.

[57] Michael Ann DeVito. 2022. How transfeminine TikTok creators navigate the algorithmic trap of visibility via folk theorization. *Proceedings of the ACM on Human-Computer Interaction* 6, CSCW2 (2022), 1–31.

[58] Roger Dingledine, Nick Mathewson, Paul F Syverson, et al. 2004. Tor: The second-generation onion router.. In *USENIX security symposium*, Vol. 4. 303–320.

[59] F Betül Durak, Thomas M DuBuisson, and David Cash. 2016. What else is revealed by order-revealing encryption?. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 1155–1166.

[60] Jennifer Earl and Katrina Kimport. 2011. *Digitally enabled social change: Activism in the internet age*. Mit Press.

[61] Ksenia Ermoshina, Harry Halpin, and Francesca Musiani. 2017. Can Johnny build a protocol? Co-ordinating developer and user intentions for privacy-enhanced secure messaging protocols. In *European Workshop on Usable Security*. 1–13.

[62] Zachary Espiritu, Marilyn George, Seny Kamara, and Lucy Qin. 2024. Synq: Public Policy Analytics Over Encrypted Data. In *IEEE Symposium on Security and Privacy (SP)*. 146–165. https://doi.org/10.1109/SP54263.2024.00085

[63] Zachary Espiritu, Seny Kamara, and Tarik Moataz. 2025. Bayesian Leakage Analysis. *IACR Comm. Crypto.* 2, 1 (2025). https://doi.org/10.62056/a36c0lmol

[64] Zachary Espiritu, Seny Kamara, Tarik Moataz, and Valentin Ogier. 2025. Leafblower: a Leakage Attack Against TEE-Based Encrypted Databases. In *IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 129–148. https://doi.org/10.1109/SP63933.2026.00008

[65] Zachary Espiritu, Seny Kamara, Tarik Moataz, and Andrew Park. 2025. PolySys: an Algebraic Leakage Attack Engine. In *34th USENIX Security Symposium (USENIX Security 25)*. 3357–3376.

[66] Zachary Espiritu, Evangelia Anna Markatou, and Roberto Tamassia. 2022. Time- and Space-Efficient Aggregate Range Queries over Encrypted Databases. *Proceedings on Privacy Enhancing Technologies* 2022, 2022.4 (2022).

[67] Virginia Eubanks. 2018. *Automating inequality: How high-tech tools profile, police, and punish the poor*. St. Martin's Press.

[68] S. Faber, S. Jarecki, H. Krawczyk, Q. Nguyen, M. Rosu, and M. Steiner. 2015. Rich queries on encrypted data: Beyond exact matches. In *European Symposium on Research in Computer Security (ESORICS '15). Lecture Notes in Computer Science*, Vol. 9327. 123–145.

[69] Francesca Falzon, Evangelia Anna Markatou, Zachary Espiritu, and Roberto Tamassia. 2022. Range Search over Encrypted Multi-Attribute Data. *Proceedings of the VLDB Endowment* 16, 4 (Dec. 2022), 587–600. https://doi.org/10.14778/3574245.3574247

[70] Stefano Ferretti, Mirko Zichichi, and Julian Sparber. 2021. Blockchain-based end-to-end encryption for Matrix instant messaging. (2021).

[71] S. Garg, P. Mohassel, and C. Papamanthou. 2016. TWORAM: Efficient Oblivious RAM in Two Rounds with Applications to Searchable Encryption. In *Advances in Cryptology - CRYPTO 2016*. 563–592. https://doi.org/10.1007/978-3-662-53015-3_20

[72] Marilyn George, Seny Kamara, Tarik Moataz, and Zachary Espiritu. 2025. Structured Encryption and Distribution-aware Leakage Suppression. In *Asiacrypt '25*.

[73] Marilyn George, Seny Kamra, and Tarik Moataz. 2021. Structured Encryption and Dynamic Leakage Suppression. In *Advances in Cryptology - EUROCRYPT 2021*.

[74] Sucheta Ghoshal and Amy Bruckman. 2019. The role of social computing technologies in grassroots movement building. *ACM Transactions on Computer-Human Interaction (TOCHI)* 26, 3 (2019), 1–36.

[75] Sucheta Ghoshal, Rishma Mendhekar, and Amy Bruckman. 2020. Toward a grassroots culture of technology practice. *Proceedings of the ACM on Human-Computer Interaction* 4, CSCW1 (2020), 1–28.

[76] Homero Gil de Zúñiga, Alberto Ardèvol-Abreu, and Andreu Casero-Ripollés. 2021. WhatsApp political discussion, conventional participation and activism: exploring direct, indirect and generational effects. *Information, communication & society* 24, 2 (2021), 201–218.

[77] E-J. Goh. 2003. *Secure Indexes*. Technical Report 2003/216. IACR ePrint Cryptography Archive. See http://eprint.iacr.org/2003/216.

[78] O. Goldreich and R. Ostrovsky. 1996. Software protection and simulation on oblivious RAMs. *J. ACM* 43, 3 (1996), 431–473.

[79] Paul Grubbs, Marie-Sarah Lacharité, Brice Minaud, and Kenneth G. Paterson. 2018. Pump up the Volume: Practical Database Reconstruction from Volume Leakage on Range Queries. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang (Eds.). ACM, 315–331. https://doi.org/10.1145/3243734.3243864

[80] Paul Grubbs, Marie-Sarah Lacharité, Brice Minaud, and Kenneth G. Paterson. 2019. Learning to Reconstruct: Statistical Learning Theory and Encrypted Database Attacks. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*. IEEE, 1067–1083. https://doi.org/10.1109/SP.2019.00030

[81] P. Grubbs, K. Sekniqi, V. Bindschaedler, M. Naveed, and T. Ristenpart. 2017. Leakage-Abuse Attacks against Order-Revealing Encryption. In *IEEE Symposium on Security and Privacy (S&P '17)*.

[82] Zichen Gui, Oliver Johnson, and Bogdan Warinschi. 2019. Encrypted Databases: New Volume Attacks against Range Queries. In *Proceedings of the 2019 ACM Conference on Computer and Communications Security* (London, United Kingdom) *(CCS '19)*. ACM, New York, NY, USA, 361–378. https://doi.org/10.1145/3319535.3363210

[83] Seda Gürses, Arun Kundnani, and Joris Van Hoboken. 2016. Crypto and empire: The contradictions of counter-surveillance advocacy. *Media, Culture & Society* 38, 4 (2016), 576–590.

[84] Gulizar Haciyakupoglu and Weiyu Zhang. 2015. Social media and trust during the Gezi protests in Turkey. *Journal of computer-mediated communication* 20, 4 (2015), 450–466.

[85] Summer Harlow. 2012. Social media and social movements: Facebook and an online Guatemalan justice movement that moved offline. *New media & society* 14, 2 (2012), 225–243.

[86] Samantha Hautea, Perry Parks, Bruno Takahashi, and Jing Zeng. 2021. Showing they care (or don't): Affective publics and ambivalent climate activism on TikTok. *Social media+ society* 7, 2 (2021), 20563051211012344.

[87] Eyako Heh and Joel Wainwright. 2022. No privacy, no peace: Urban surveillance and the movement for Black lives. *Journal of Race, Ethnicity and the City* 3, 2 (2022), 121–141.

[88] Philip N Howard, Aiden Duffy, Deen Freelon, Muzammil M Hussain, Will Mari, and Marwa Maziad. 2011. Opening closed regimes: what was the role of social media during the Arab Spring? *Available at SSRN 2595096* (2011).

[89] Philip N Howard and Muzammil M Hussain. 2013. *Democracy's fourth wave?: digital media and the Arab Spring*. Oxford University Press.

[90] Hypothesis. 2023. Collaborate with anyone, anywhere. (2023). https://web.hypothes.is/

[91] M. Saiful Islam, M. Kuzu, and M. Kantarcioglu. 2012. Access Pattern disclosure on Searchable Encryption: Ramification, Attack and Mitigation. In *Network and Distributed System Security Symposium (NDSS '12)*.

[92] Takanori Isobe and Ryoma Ito. 2021. Security analysis of end-to-end encryption for zoom meetings. *IEEE access* 9 (2021), 90677–90689.

[93] Farnaz Jahanbakhsh, Amy X Zhang, and David R Karger. 2022. Leveraging structured trusted-peer assessments to combat misinformation. *Proceedings of the ACM on Human-Computer Interaction* 6, CSCW2 (2022), 1–40.

[94] Jakob Bjerre Jakobsen and Claudio Orlandi. 2015. A practical cryptanalysis of the Telegram messaging protocol. *Aarhus university* (2015).

[95] Tim Jenkin. 1995. *Tim Jenkin: Talking with Vula*. Mayibuye.

[96] Hayley Johnson. 2017. #NoDAPL: Social media, empowerment, and civic participation at Standing Rock. *Library Trends* 66, 2 (2017), 155–175.

[97] Mireya Jurado, Catuscia Palamidessi, and Geoffrey Smith. 2021. A Formal Information-Theoretic Leakage Analysis of Order-Revealing Encryption. In *2021 IEEE 34th Computer Security Foundations Symposium (CSF)*. 1–16. https://doi.org/10.1109/CSF51468.2021.00046

[98] Mireya Jurado and Geoffrey Smith. 2019. Quantifying Information Leakage of Deterministic Encryption. In *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop* (London, United Kingdom) *(CCSW'19)*. Association for Computing Machinery, New York, NY, USA, 129–139. https://doi.org/10.1145/3338466.3358915

[99] Sina Kamali and Diogo Barradas. 2025. Anix: Anonymous Blackout-Resistant Microblogging with Message Endorsing. In *2025 IEEE Symposium on Security and Privacy (SP)*. 1381–1399. https://doi.org/10.1109/SP61157.2025.00015

[100] Seny Kamara. 2020. *COINTELPRO*. Algorithms for the People. http://algosforthepeople.org/posts/cointelpro/

[101] Seny Kamara. 2020. *Crypto for the People Invited Talk*. The International Association for Cryptologic Research. https://www.youtube.com/watch?v=Ygq9ci0GFhA

[102] Seny Kamara, Abdelkarim Kati, Tarik Moataz, Jamie DeMaria, Andrew Park, and Amos Treiber. 2024. MAPLE: MArkov Process Leakage attacks on Encrypted Search. *Proceedings on Privacy Enhancing Technologies* (2024).

[103] Seny Kamara, Abdelkarim Kati, Tarik Moataz, Thomas Schneider, Amos Treiber, and Michael Yonli. 2022. SoK: Cryptanalysis of Encrypted Search with LEAKER – A framework for LEakage AttacK Evaluation on Real-world data. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*. 90–108. https://doi.org/10.1109/EuroSP53844.2022.00014

[104] S. Kamara and T. Moataz. 2019. Computationally Volume-Hiding Structured Encryption. In *Advances in Cryptology - Eurocrypt' 19*.

[105] Seny Kamara and Tarik Moataz. 2023. Design and Analysis of a Stateless Encrypted Document Database. https://www.mongodb.com/collateral/stateless-document-database-encryption-scheme.

[106] Seny Kamara, Tarik Moataz, and Olya Ohrimenko. 2018. Structured Encryption and Leakage Suppression. In *Advances in Cryptology - CRYPTO '18*.

[107] Seny Kamara, Tarik Moataz, Andrew Park, and Lucy Qin. 2021. A decentralized and encrypted national gun registry. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1520–1537.

[108] G. Kellaris, G. Kollios, K. Nissim, and A. O' Neill. 2016. Generic Attacks on Secure Outsourced Databases. In *ACM Conference on Computer and Communications Security (CCS '16)*.

[109] Os Keyes. 2018. The misgendering machines: Trans/HCI implications of automatic gender recognition. *Proceedings of the ACM on human-computer interaction* 2, CSCW (2018), 1–22.

[110] Os Keyes. 2019. Counting the Countless: Why data science is a profound threat for queer people. *Real Life* 2 (2019).

[111] Evgenios M. Kornaropoulos, Nathaniel Moyer, Charalampos Papamanthou, and Alexandros Psomas. 2022. Leakage Inversion. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. ACM. https://doi.org/10.1145/3548606.3560593

[112] Evgenios M Kornaropoulos, Charalampos Papamanthou, and Roberto Tamassia. 2020. The state of the uniform: Attacks on encrypted databases beyond the uniform query distribution. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1223–1240.

[113] Evgenios M Kornaropoulos, Charalampos Papamanthou, and Roberto Tamassia. 2021. Response-hiding encrypted ranges: Revisiting security via parametrized leakage-abuse attacks. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1502–1519.

[114] Marie-Sarah Lacharité, Brice Minaud, and Kenneth G. Paterson. 2018. Improved Reconstruction Attacks on Encrypted Data Using Range Query Leakage. In *2018 IEEE Symposium on Security and Privacy (SP)*. 297–314. https://doi.org/10.1109/SP.2018.00002

[115] Simon Langowski, Sacha Servan-Schreiber, and Srinivas Devadas. 2022. Trellis: Robust and scalable metadata-private anonymous broadcast. In *Network and Distributed System Security Symposium*. 18 pages. https://doi.org/10.14722/ndss.2023.23088

[116] Andrei Lapets, Frederick Jansen, Kinan Dak Albab, Rawane Issa, Lucy Qin, Mayank Varia, and Azer Bestavros. 2018. Accessible privacy-preserving web-based data analysis for assessing and addressing economic inequalities. In *Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies*. 1–5.

[117] Francis LF Lee and Joseph Man Chan. 2016. Digital media activities and mode of participation in a protest campaign: A study of the Umbrella Movement. *Information, Communication & Society* 19, 1 (2016), 4–22.

[118] Francis LF Lee, Michael Chan, and Hsuan-Ting Chen. 2020. Social media and protest attitudes during movement abeyance: A study of Hong Kong university students. *International Journal of Communication* 14 (2020), 20.

[119] Ada Lerner, Helen Yuxun He, Anna Kawakami, Silvia Catherine Zeamer, and Roberto Hoyle. 2020. Privacy and activism in the transgender community. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

[120] Ada Lerner, Eric Zeng, and Franziska Roesner. 2017. Confidante: Usable encrypted email: A case study with lawyers and journalists. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 385–400.

[121] Yonatan Levy and Uri Yechiali. 1976. An M/M/s queue with servers' vacations. *INFOR: Information Systems and Operational Research* 14, 2 (1976), 153–163.

[122] Hai Liang and Francis LF Lee. 2023. Opinion leadership in a leaderless movement: discussion of the anti-extradition bill movement in the 'LIHKG'web forum. *Social movement studies* 22, 5-6 (2023), 670–688.

[123] Karsten Loesing, Steven J. Murdoch, and Roger Dingledine. 2010. A Case Study on Measuring Statistical Data in the Tor Anonymity Network. In *Proceedings of the Workshop on Ethics in Computer Security Research (WECSR 2010)* (Tenerife, Canary Islands, Spain) *(LNCS)*. Springer.

[124] Tetyana Lokot. 2018. Be safe or be seen? How Russian activists negotiate visibility and security in online resistance practices. *Surveillance & Society* 16, 3 (2018), 332–346.

[125] Eva Luvison, Sylvain Chatel, Justinas Sukaitis, Vincent Graf Narbel, Carmela Troncoso, and Wouter Lucks. 2025. A Low-Cost Privacy-Preserving Digital Wallet for Humanitarian Aid Distribution. In *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1234–1250.

[126] Robertas Maleckas, Kenneth G Paterson, and Martin R Albrecht. 2023. Practically-exploitable Vulnerabilities in the Jitsi Video Conferencing System. *Cryptology ePrint Archive* (2023).

[127] Evangelia Anna Markatou, Francesca Falzon, Zachary Espiritu, and Roberto Tamassia. 2023. Attacks on Encrypted Response-Hiding Range Search Schemes in Multiple Dimensions. *Proceedings on Privacy Enhancing Technologies* 2023, 4 (Oct. 2023), 204–223. https://doi.org/10.56553/popets-2023-0106

[128] Alice E Marwick and Danah Boyd. 2011. I tweet honestly, I tweet passionately: Twitter users, context collapse, and the imagined audience. *New media & society* 13, 1 (2011), 114–133.

[129] Alexandra Mateescu, Douglas Brunton, Alex Rosenblat, Desmond Patton, Zachary Gold, and Danah Boyd. 2015. Social media surveillance and law enforcement. *Data Civ Rights* 27 (2015), 2015–2027.

[130] Damon McCoy, Kevin Bauer, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. 2008. Shining light in dark places: Understanding the Tor network. In *Privacy Enhancing Technologies: 8th International Symposium, PETS 2008 Leuven, Belgium, July 23-25, 2008 Proceedings 8*. Springer, 63–76.

[131] MongoDB. 2023. Queryable Encryption. https://www.mongodb.com/docs/manual/core/queryable-encryption/.

[132] Luke P Morrison, Brian Team, Brian Nguyen, Senthil Kannan, Nathan Ray, and Gregory C Lewin. 2017. AirChat: Ad hoc network monitoring with drones. In *2017 Systems and Information Engineering Design Symposium (SIEDS)*. IEEE, 38–43.

[133] Marcia Mundt, Karen Ross, and Charla M Burnett. 2018. Scaling social movements through social media: The case of Black Lives Matter. *Social Media+ Society* 4, 4 (2018), 2056305118807911.

[134] Steven J Murdoch and George Danezis. 2005. Low-cost traffic analysis of Tor. In *2005 IEEE Symposium on Security and Privacy (S&P'05)*. IEEE, 183–195.

[135] Legostaeva Natalya and Svetlov Kirill. 2022. The QR code as a symbol of protest: analysis of social network VKontakte. *International Journal of Open Information Technologies* 10, 11 (2022), 41–46.

[136] M. Naveed, S. Kamara, and C. V. Wright. 2015. Inference Attacks on Property-Preserving Encrypted Databases. In *ACM Conference on Computer and Communications Security (CCS)* (Denver, Colorado, USA) *(CCS '15)*. ACM, 644–655. https://doi.org/10.1145/2810103.2813651

[137] Andrew Neef. 2019. Infrared Aerial Surveillance Used at Standing Rock to Monitor and Track Protesters. (2019). https://unicornriot.ninja/2019/infrared-aerial-surveillance-used-at-standing-rock-to-monitor-and-track-protesters/

[138] Rasmus Kleis Nielsen. 2009. The labors of Internet-assisted activism: Overcommunication, miscommunication, and communicative overload. *Journal of Information Technology & Politics* 6, 3-4 (2009), 267–280.

[139] Rasmus Kleis Nielsen. 2011. Mundane internet tools, mobilizing practices, and the coproduction of citizenship in political campaigns. *New Media & Society* 13, 5 (2011), 755–771.

[140] Rasmus Kleis Nielsen. 2013. Mundane internet tools, the risk of exclusion, and reflexive movements—Occupy Wall Street and political uses of digital networked technologies. *The Sociological Quarterly* 54, 2 (2013), 173–177.

[141] Safiya Umoja Noble. 2018. *Algorithms of oppression*. New York University Press.

[142] Rodrigo Ochigame and James Holston. 2016. FILTERING DISSENT Social Media and Land Struggles in Brazil. (2016).

[143] Stephen Owen. 2017. Monitoring social media and protest movements: Ensuring political order through surveillance and surveillance discourse. *Social Identities* 23, 6 (2017), 688–700.

[144] Simon Oya and Florian Kerschbaum. 2021. Hiding the Access Pattern is Not Enough: Exploiting Search Pattern Leakage in Searchable Encryption.. In *USENIX Security Symposium*. 127–142.

[145] Simon Oya and Florian Kerschbaum. 2022. IHOP: Improved Statistical Query Recovery against Searchable Symmetric Encryption through Quadratic Optimization. In *Proceedings of the 31st USENIX Security Symposium*. 2407–2424.

[146] V. Pappas, F. Krell, B. Vo, V. Kolesnikov, T. Malkin, S.-G. Choi, W. George, A. Keromytis, and S. Bellovin. 2014. Blind seer: A scalable private dbms. In *Security and Privacy (SP), 2014 IEEE Symposium on*. IEEE, 359–374.

[147] Sarvar Patel, Giuseppe Persiano, Kevin Yeo, and Moti Yung. 2019. Mitigating Leakage in Secure Cloud-Hosted Data Structures: Volume-Hiding for Multi-Maps via Hashing. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz (Eds.). ACM, 79–93. https://doi.org/10.1145/3319535.3354213
[148] The Tor Project. 2023. Tor Specifications: Circuits. (2023). https://spec.torproject.org/dos-spec/overview.html#circuits
[149] Kevin Purdy. 2025. *Russia-aligned hackers are targeting Signal users with device-linking QR codes.* Available at https://arstechnica.com/information-technology/2025/02/russia-aligned-hackers-are-targeting-signal-users-with-device-linking-qr-codes/.
[150] Anjana Rajan, Lucy Qin, David W Archer, Dan Boneh, Tancrede Lepoint, and Mayank Varia. 2018. Callisto: A cryptographic approach to detecting serial perpetrators of sexual misconduct. In *Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies*. 1–4.
[151] Phillip Rogaway. 2015. The moral character of cryptographic work. *Cryptology ePrint Archive* (2015).
[152] Kevin Roose. 2020. Social media giants support racial justice. Their products undermine it. *The New York Times* (2020).
[153] Leah Rosenbloom. 2021. Toward Secure Social Networks for Activists. In *Moving technology ethics at the forefront of society, organisations and governments*. Universidad de La Rioja, 491–502.
[154] Leah Namisa Rosenbloom. 2022. Activists Want Better, Safer Technology. *arXiv preprint arXiv:2209.01273* (2022).
[155] Leah Namisa Rosenbloom. 2024. Cryptography and Collective Power. *Cryptology ePrint Archive* (2024).
[156] Leah Namisa Rosenbloom. 2025. Cryptography and collective power. In *International Conference on Cryptology and Information Security in Latin America*. Springer, 3–39.
[157] Scott Ruoti, Jeff Andersen, Daniel Zappala, and Kent Seamons. 2015. Why Johnny still, still can't encrypt: Evaluating the usability of a modern PGP client. *arXiv preprint arXiv:1510.08555* (2015).
[158] John F Shortle, James M Thompson, Donald Gross, and Carl M Harris. 2018. *Fundamentals of queueing theory*. Vol. 399. John Wiley & Sons.
[159] Marko M Skoric, Nathaniel D Poor, Youqing Liao, and Stanley Wei Hong Tang. 2011. Online organization of an offline protest: From social to traditional media and back. In *2011 44th Hawaii International Conference on System Sciences*. IEEE, 1–8.
[160] D. Song, D. Wagner, and A. Perrig. 2000. Practical Techniques for Searching on Encrypted Data. In *IEEE Symposium on Research in Security and Privacy*. IEEE Computer Society, 44–55.
[161] Amory Starr, Luis A Fernandez, Randall Amster, Lesley J Wood, and Manuel J Caro. 2008. The impacts of state surveillance on political assembly and association: A socio-legal analysis. *Qualitative Sociology* 31 (2008), 251–270.
[162] E. Stefanov, M. van Dijk, E. Shi, C. Fletcher, L. Ren, X. Yu, and S. Devadas. 2013. Path ORAM: An Extremely Simple Oblivious RAM Protocol. In *ACM Conference on Computer and Communications Security (CCS '13)*.
[163] Tomáš Sušánka and Josef Kokeš. 2017. Security analysis of the Telegram IM. In *Proceedings of the 1st Reversing and Offensive-oriented Trends Symposium*. 1–8.
[164] Hideaki Takagi. 1994. M/G/1//N queues with server vacations and exhaustive service. *Operations Research* 42, 5 (1994), 926–939.
[165] Yannis Theocharis, Will Lowe, Jan W Van Deth, and Gema García-Albacete. 2015. Using Twitter to mobilize protest action: online mobilization patterns and action repertoires in the Occupy Wall Street, Indignados, and Aganaktismenoi movements. *Information, Communication & Society* 18, 2 (2015), 202–220.
[166] Iolanda Tortajada, Cilia Willem, R Lucas Platero Méndez, and Núria Araüna. 2021. Lost in transition? Digital trans activism on Youtube. *Information, Communication & Society* 24, 8 (2021), 1091–1107.
[167] Mark Tremayne. 2016. Anatomy of protest in the digital era: A network analysis of Twitter and Occupy Wall Street. In *Social Networks and Social Movements*. Routledge, 110–126.
[168] Emiliano Treré. 2015. Reclaiming, proclaiming, and maintaining collective identity in the #YoSoy132 movement in Mexico: An examination of digital frontstage and backstage activism through social media and instant messaging platforms. *Information, Communication & Society* 18, 8 (2015), 901–915.
[169] Emiliano Treré. 2020. The banality of WhatsApp: On the everyday politics of backstage activism in Mexico and Spain. *First Monday* 25 (2020).
[170] Zeynep Tufekci and Christopher Wilson. 2012. Social media and the decision to participate in political protest: Observations from Tahrir Square. *Journal of communication* 62, 2 (2012), 363–379.
[171] Temple Uwalaka, Scott Rickard, and Jerry Watkins. 2018. Mobile social networking applications and the 2012 Occupy Nigeria protest. *Journal of African Media Studies* 10, 1 (2018), 3–19.
[172] Sebastián Valenzuela. 2013. Unpacking the use of social media for protest behavior: The roles of information, opinion expression, and activism. *American behavioral scientist* 57, 7 (2013), 920–942.
[173] Kandrea Wade, Jed R Brubaker, and Casey Fiesler. 2021. Protest privacy recommendations: An analysis of digital surveillance circumvention advice during black lives matter protests. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–6.
[174] Hue Watson, Eyitemi Moju-Igbene, Akanksha Kumari, and Sauvik Das. 2020. " We Hold Each Other Accountable": Unpacking How Social Groups Approach Cybersecurity and Privacy Together. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–12.
[175] Alma Whitten and J Doug Tygar. 1999. Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0.. In *USENIX security symposium*, Vol. 348. 169–184.
[176] Lai Yan-ho and Ming Sing. 2020. Solidarity and Implications of a Leaderless Movement in Hong Kong. *Communist and Post-Communist Studies* 53, 4 (2020), 41–67.
[177] Samson Yuen and Gary Tang. 2023. Instagram and social capital: youth activism in a networked movement. *Social Movement Studies* 22, 5-6 (2023), 706–727.
[178] Y. Zhang, J. Katz, and C. Papamanthou. 2016. All Your Queries Are Belong to Us: The Power of File-Injection Attacks on Searchable Encryption. In *USENIX Security Symposium*.
[179] Sacha Zyto, David Karger, Mark Ackerman, and Sanjoy Mahajan. 2012. Successful classroom deployment of a social document annotation system. In *Proceedings of the sigchi conference on human factors in computing systems*. 1883–1892.

A Related Work (Continued)

Among the most widely studied are social media platforms like Facebook [85, 96, 133], Twitter [128, 165, 167], YouTube [21, 124, 166], TikTok [86], LIHKG [6, 54, 122, 176], and Instagram [177], blogs [88, 89, 124], end-to-end encrypted messaging protocols and applications like Pretty Good Privacy (PGP) and Off The Record (OTR) [33, 61, 120, 157, 175], WhatsApp [19, 76, 169], Signal [9, 48], Telegram [1, 94, 163] (which has limited encryption and some social media-like features), video conferencing platforms like Zoom [28, 92] and Jitsi [28, 126], and networking protocols like Tor [130, 134], Bridgefy [7], and Firechat [54, 132]. Other organizing-related ICTs with varying degrees of privacy and security claims such as specific virtual private networks (VPNs) [53], LINE and KakaoTalk [54], Keybase [120], Matrix [70], Session, Briar, ChatSecure, and Wire [61] are less well-studied, if at all.

One distinction between older protocols like PGP and OTR and modern-day applications like Signal and Telegram are that the former are *platform independent protocols*, while the latter are *platform-dependent applications* that require all users to adopt the same platform and vet the security of each on an independent basis. While many early studies of end-users' experience with PGP foreground the lack of usability of secure communication tools [157, 175], more contemporary studies have identified new obstacles such as "fragmented user bases and lack of interoperability" [2, 61], a sense of privacy nihilism or futility [2, 56, 154], and an incomplete or inaccurate understanding of privacy guarantees and end-to-end encryption in general [2, 6, 56, 120].

Finally, there is a wide gap between extant privacy-preserving technologies and specific, important tasks that grassroots organizers need to perform for safety, visibility, or "backstage" organizational reasons, causing them to adopt riskier options. For example, participants in the anti-ELAB movement needed to ensure they would be able to wipe one another's devices remotely in the event of device seizure; this led them to use a combination of Telegram and Life360, despite the fact that Life360 is stalkerware and not at all privacy-preserving from a standard security perspective [6].

B Design Considerations

In this section, we review related work in the humanities and social sciences which discusses relevant historical context at the intersection of technology and social movements and describe how this context leads us to the three core design goals of the tigro protocol.

Historical Context. Activists have used digital technologies extensively to facilitate grassroots organizing, as documented in studies of the following: anti-apartheid activism in South Africa [95, 101], political organizing in the U.S. [138–140], anti-censorship protests in Singapore [159], the Arab Spring [8, 88, 89, 170], the YoSoy132 movement in Mexico [168], political organizing in Guatemala [85], the Occupy Wall Street [167] and inspired [17, 165, 171] movements, education and energy policy activism in Chile [172], the Euromaidan Uprising in Ukraine [31], the Dakota Access Pipeline protests at Standing Rock [55, 96], the Gezi protests in Turkey [84], the Umbrella [117] and anti-Extradition Law Amendment Bill (anti-ELAB) [6, 177] Movements in Hong Kong [118], Anti-Corruption Foundation activism in Russia [124], the Black Lives Matter movement [87, 133, 154, 173], Science for the People-Atlanta [74], Southern Movement Assembly [75], and transgender rights [119] activists in the U.S., the Sudanese revolution [53], and more general studies [33, 49, 60, 76].

Our formal insights into activists' relationships with digital technologies come from the testimony in the above studies, as well as from expert analyses of the historical modes of suppression of grassroots movements [32, 34, 143, 161] that have been reproduced, reinforced, and expanded in the digital age. For example, the Federal Bureau of Investigation (FBI) used the COINTELPRO operation to illegally and extensively surveil, incarcerate, and assassinate Black civil rights activists from 1956-1971 [46, 100]. Edward Snowden revealed in 2013 that U.S. military and intelligence agencies, with support from U.S. technology corporations, had widely adapted and deployed these techniques in digital spaces—and have since wielded them against marginalized activists [18, 87, 133, 137, 143]. Simone Browne [35], Ruha Benjamin [27], and many others have discussed the systematic ways in which surveillance technologies are disproportionately wielded against People of Color and people with intersections of multiply-marginalized identities, for instance Black women [37, 141], working-class people [67], and trans and queer people [109, 110].

Design Goal #1: Digital Trust = Collective Physical Trust. While there is some debate about whether meaningful collective action can take place digitally without a physical component [45, 60], the vital role of shared physicality, or "copresence," in grassroots movements is undeniable [6, 49, 60, 84, 85, 154, 159]. To quote a participant in the anti-ELAB movement, "standing on the front line together is very important for trust" [6].

Participants in grassroots movements often navigate between physical and digital realms, and root their trust of digital objects (a digital identity, text, image, video, etc.) in the perceived object creator's trustworthiness within a shared community [6, 53, 133, 154]. When direct physical interaction is not possible, participants seek a "trusted first-hand source" [53]—someone who has directly witnessed the trustworthiness of the information "on-the-ground" and who can vouch for its legitimacy [6, 53, 154].

"Ground trust" is the fundamental building block around which we construct our digital trust infrastructure for grassroots organizing. Participants exchange ephemeral key material in a shared physical location, then use the key material to compute a unique shared secret, or *ground key*. The ground key is not tied to either participant's individual identity, but rather to the participants' *meeting*, or pairwise occupation of the same physical space. Participants can use the ground key in covert ways to identify one another and privately share information online, while remaining anonymous to everyone else.

The ground trust key exchange is similar to the PGP key signing ceremony and the Signal key verification procedure in that it involves exchanging information in person. However, in order to provide essential security functions for participants in grassroots movements, formation of the ground key differs from existing key exchange paradigms in an important way: participants do not need to share any personally identifiable or digitally linkable information such as long-term public keys or phone numbers to form a ground key [6]. Instead, participants exchange *ephemeral* key material which they use to form a pairwise *symmetric* key. They can then use the key to seed various cryptographic protocols.

In the case of tigro, each pair of contacts sets up a shared *grounded private post office (PO) box* on the tigro server. The content inside of the box is encrypted using the ground key, such that only the trusted pair is able to decrypt it. Communication via the box is asynchronous in the traditional sense—both parties do not have to be online at the same time to access content—but is also asynchronous in a stronger sense. Unlike an end-to-end encrypted messenger, where content is free-flowing and relatively disorganized, one participant in the tigro protocol "drops" information in the box with a *label*, allowing the other participant to retrieve the information only when it is relevant. In the next section, we discuss how participants can drop and retrieve *digital annotations* pertaining to various people, events, and digital artifacts into the box in order to facilitate *information sharing on a need-to-know basis*.

Design Goal #2: Information on a Need-to-Know Basis. By implementing the tigro post office box functionality using content-based, label-associated "drop" and "retrieve" operations (rather than the unlabeled messenger-style "send" and "receive"), we constrain annotation sharing to the case in which the receiver already knows about the associated labeled content, and needs further information in order to engage with it safely.

The quality of "need-to-know" is revocable—information sharing on the tigro platform is meant to be (digitally) ephemeral. The "retrieve" operation ensures that only necessary annotations will reach the memory of the participant's device, which may be compromised in the event of an arrest or device seizure. Once the participant reads the annotation, the vetter's impression of the content will remain in the participant's physical memory (their mind); as the annotation is not meant to invite response or sharing, there is no need for the annotation to remain on the participant's device. As we will discuss in the future work section, the physical erasure of this content from application memory is left to the implementation layer.

While clearly a priority for anti-ELAB movement participants, the problem of remote deletion is extremely delicate cryptographically speaking, as it requires someone other than the device owner to maintain administrative access to the device, as well as copies of the owner's secret keys or PIN (if the device is encrypted). We recognize the importance and difficulty of implementing this feature in both theory and practice, and hope to address it in future iterations of the project. For now, our construction provides a more basic capability of deletion (which allows for schedule deletion), similar to what exists on the popular end-to-end encrypted messaging platforms Signal and WhatsApp. Our version of deletion is slightly more expressive in that either participant can delete content from the shared PO box, regardless of who sent the content (in Signal, for example, only the content sender can delete the content "for everyone").

Design Goal #3: Think Toward Deployment. As with the erasure of fetched content discussed above, there are many security and usability details that we feel are best left to the implementation layer. We discuss these as future work in Appendix C.

C Future Work

We hope to integrate the following features into the tigro application. Although we do not handle properties in this section explicitly or formally (in the provable security sense), they are important considerations for "big-picture security" [155], a realistic model in which cryptographic primitives are taken in the context of larger sociotechnical systems. In the future we hope there will be provably secure, practical and universally composable versions of all of the (technological) primitives—but even then, working toward full compromise security must take into account the fluctuating state of human trust relationships, threat models, and co-education practices, code repositories, device-layer security, and the law. We think toward (and leave room for) the following features in our design.

Application Prototype. In the application prototype shown in Figure 4, oids can be auto-generated based on digital artifacts' URLs; they can also be randomly assigned or user-generated. Although discoverability was not one of our initial design goals (as we focused on mitigating information overload and context collapse), the tigro infrastructure can be leveraged as-is to achieve discoverability based on participant-defined keywords. For example, using special pre-defined terms for the oid such as "FYI," "URGENT," or "MUTUAL AID," tigro can take on a content-based publish-subscribe functionality similar to that of social media, where participants can "post" digital artifacts as bulletins or announcements to their contacts as annotations. This affords digital artifacts that are priorities to participants added discoverability, and may be especially useful for aggregating disparate social media spaces or sharing social media artifacts with participants who cannot otherwise access them (such as people who do not have Facebook or Instagram accounts).

Key Storage. All tigro protocol keys will be stored and computed upon inside trusted platform modules (TPMs).

Low-Tech Access. Low- or no-tech fallbacks in the event of network outages and in no- or low-service geographic areas, as well as physical locations for communal information technology (IT) services and information sharing, are of essential importance to grassroots movements all over the world [6, 31, 53, 88, 96]. While we recognize that a centralized server is a roadblock to effective use of the tigro protocol during a network outage, the embedding of participant's ephemeral key material into QR codes allows them to exchange keys without an internet connection via their local device cameras or by printing them out. This allows participants to defer digitization until after sensitive physical gatherings, during which participants might want to leave digital devices turned off or at home [32, 173]. In the future, we hope to make a decentralized version of the tigro protocol.

Ephemeral Fetching. When the participant's device "retrieves" and decrypts content, the content is loaded into application memory and displayed. Once the participant navigates away from that display, the application memory pertaining to that content will be overwritten with 0s.

DDoS Protection. Given the setting of anonymous clients, the tigro server will be as robust as possible against DDoS attacks.

Group Sharing. Ground keys in the tigro system are currently shared between two participants only. Group sharing will therefore be an application-layer feature that breaks down a group or "authorization" list into point-to-point connections. While this is less efficient than establishing a group key, the "groundedness" of the connections inherently limits the number of participants' shared contacts—it may be sufficient for "grassroots optimization" [155]. In future iterations of the tigro protocol we hope to create a "grounded group key" which is the result of two or more participants physically meeting up to exchange ephemeral key material.

Browser and Application Overlay. Similar to PGP and OTR, tigro is a protocol rather than a designated platform or application—content from anywhere can be assigned a tigro identifier and "annotated." As such, we look toward an interoperable implementation such as a browser or application overlay [93, 179] that seamlessly integrates the tigro protocol with existing organizing technology like social media. For example, a participant might see a post on social media and wish to vet the trustworthiness of the post with respect to their organizing network; the tigro browser overlay would scan the content, search the participant's shared boxes for annotations, and display a small tigro icon that would expand the annotation (and the annotation's originator) upon click.

Digital Security Trainings. Borradaile, Kretschmer, Gretes, and LeClerc found that hands-on training workshops can produce sustained use of secure communication tools for grassroots organizing, even when the tool (PGP) is notoriously difficult to navigate [33]. The language and explanations surrounding the tigro protocol (such as the post office box analogy and concept of annotations) are designed with future collaborative training workshops in mind, and we intend to work directly with organizers on developing and deploying the first iteration of the tigro protocol.

D Notation

The set of all binary strings of length n is denoted as $\{0,1\}^n$, and the set of all finite binary strings as $\{0,1\}^*$. $[n]$ is the set of integers $\{1,\ldots,n\}$. $a := b$ means that a is set to b. The output y of a probabilistic algorithm $\mathcal{A}$ on input x is denoted by $y \leftarrow \mathcal{A}(x)$.

Functionality $\mathcal{F}_{\text{anon}}$

The functionality interacts with a client $\mathbf{C}_a$, the server $\mathbf{S}$ and an adversary $\mathcal{A}$:

- $\text{Send}_{\mathbf{C}_a,\mathbf{S}}(m;\perp)$:
  (1) set $\text{leak}_{\mathcal{A}} := m$;
  (2) output $(\perp; m; \text{leak}_{\mathcal{A}})$;

Figure 5: Anonymous channel

The output y of a deterministic algorithm $\mathcal{A}$ on input x is denoted by $y := \mathcal{A}(x)$. If S is a set then $x \leftarrow\!\$\ S$ denotes sampling from S uniformly at random. Given a sequence $\mathbf{s}$ of n elements, we refer to its ith element as s_i. If S is a set then $\#S$ refers to its cardinality. Throughout, k will denote the security parameter and we assume 1^k is implicitly provided as input to every algorithm and protocol. The symbol $\perp_A$ is a special abort symbol that causes any party to abort. Given elements a and b from a totally ordered set, we write $\text{sort}(ab)$ to denote a concatenated with b if $a < b$ or b concatenated with a if $b > a$.

E Anonymous Channel Functionality

The anonymous channel functionality is given in Figure 5.

F Structured Encryption

Structured encryption. Let $\mathscr{T}$ be an abstract data type that supports a set of operations $(\text{op}\langle 1\rangle, \ldots, \text{op}\langle m\rangle)$. A multi-client structured encryption scheme $\Sigma_{\mathscr{T}}$ for $\mathscr{T}$, consists of $m+1$ protocols $(\text{Op}\langle 0\rangle, \ldots \text{Op}\langle m\rangle)$ executed between n clients $\mathbf{C}_1, \ldots, \mathbf{C}_n$ and a server $\mathbf{S}$ such that for all $0 \le i \le m$:

$$\left(\left(\text{st}_a, \text{out}_a\right)_{a\in I_{\mathbb{C}}}; \text{st}_{\mathbf{S}}, \text{out}_{\mathbf{S}}\right) \leftarrow \Sigma_{\mathscr{T}}.\text{Op}\langle\omega_i\rangle_{\mathbb{C},\mathbf{S}}\left(\left(\text{st}_a, \text{in}_a\right)_{a\in I_{\mathbb{C}}}; \text{st}_{\mathbf{S}}, \text{in}_{\mathbf{S}}\right)$$

where $\mathbb{C} \subseteq \{\mathbf{C}_1 \ldots, \mathbf{C}_n, \perp\}$ and $I_{\mathbb{C}} = \{x \in [n] : \mathbf{C}_x \in \mathbb{C}\}$. We assume that the input of every operation of every party includes a globally unique structure identifier stid.

We will refer to $\text{Op}\langle 0\rangle$ as the initialization operation Init for which $\text{st}_{\mathbb{C}} = \text{st}_{\mathbf{S}} = \perp$ and $\text{in}_a = \text{params}$ for some $a \in I_{\mathbb{C}}$ are the initialization parameters of the data structure.

F.1 Leakage Graphs

We model leakage using leakage graphs, as introduced in [105]. A leakage profile is a function $\mathcal{L}$ that takes as input the data structure and a sequence of operations and outputs a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ that captures all the leakage. More precisely, consider an STE scheme Σ, and a sequence of operations $\mathbf{op} = (\text{op}_0, \ldots, \text{op}_n)$. $\mathcal{L}$ is defined as a *stateful* function that, given an operation op_i, for $i \ge 1$, returns a vertex and set of edges $(\text{vx}_i, \mathbf{E}_i)$, where $\mathbf{E}_i$ is over $\bigcup_{j=1}^{i-1} \text{vx}_j$. Intuitively, one can view $(\text{vx}_i, \mathbf{E}_i)$ as the leakage of operation op_i.

Public/secret vertices. Vertices in leakage graphs have a special structure which consists of a *public* component and of a *secret* component. The public component of a vertex is visible to the adversary/simulator but the secret component is not. In a leakage graph, we describe a vertex vx with the notation $\langle\!\langle\ P \mid S\ \rangle\!\rangle$, where P is the public component and S is the secret component. Intuitively speaking, the public component of a vertex is a set that stores information about the operation that is leaked. The secret component, on the other hand, is a set used by the leakage profile to store information about the operation that will help it define/construct adversarially-visible edges between vertices. Intuitively, these edges capture correlations between different operations. We note that correlations between vertices/operations can exist due to elements of the public component but, by definition, these correlations are public so they are not explicitly modeled by edges; though they can be recovered by just observing the public components of vertices. The purpose of the edges is to capture correlations about *secret* information. Of course, this does not mean that correlations due to public components is not important, especially since it could be combined with correlations due to secret components.

Vertex filters. Given a leakage graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, we write $\mathbf{V}(\text{ pred } \mid \cdot) \subseteq \mathbf{V}$ to denote the set of vertices in $\mathbf{V}$ whose public component satisfies the CNF predicate

$$\text{pred} = \left(x_{1,1} \vee \cdots \vee x_{1,m}\right) \wedge \cdots \wedge \left(x_{n,1} \vee \cdots \vee x_{n,m}\right).$$

More precisely,

$$\mathbf{V}(\text{ pred } \mid \cdot) = \left\{\text{vx} \in \mathbf{V} : \bigwedge_{i=1}^{n}\bigvee_{j=1}^{m} x_{i,j} \in \text{pub}(\text{vx})\right\}$$

where $\text{pub}(\text{vx})$ denotes the public component of vx.

Black-box leakage. Leakage graphs capture the leakage of a given scheme Σ. It is common, however, that a scheme Σ is used as a building block of another scheme Ω and that Ω's leakage depends on the leakage of Σ. This relationship can be captured as follows. If Ω makes use of Σ, then Ω's leakage graph $\mathbf{G}_\Omega = (\mathbf{V}_\Omega, \mathbf{E}_\Omega)$ is composed of *meta-vertices* which have the form

$$\check{\text{vx}} = [\![\ \Omega, t_\Omega\ ;\ \text{vx}_1, \ldots, \text{vx}_n\ ]\!],$$

where Ω and t_Ω is meta-data that captures the fact that the vertex resulted from an Ω operation, t_Ω denotes that operation's time or rank and $\text{vx}_1, \ldots, \text{vx}_n$ are the $\mathbf{G}_\Sigma$-vertices that resulted from the operation. Note that for visual clarity we will omit the time/rank's subscript and just write t instead of t_Ω. The scheme that the time/rank applies to should be clear from the context. Note that the meta-vertices $\check{\text{vx}} \in \mathbf{V}_\Omega$ of $\mathbf{G}_\Omega$ can be viewed as hyperedges over $\mathbf{V}_\Sigma$ with additional meta-data. Edges between meta-vertices can result from Ω itself but are also induced by the edges in $\mathbf{E}_\Sigma$. More precisely, if an edge $(\text{vx}_1, \text{vx}_2) \in \mathbf{E}_\Sigma$ is such that vx_1 and vx_2 are in different $\mathbf{G}_\Omega$ meta-vertices $\check{\text{vx}}_1$ and $\check{\text{vx}}_2$ then $\mathbf{E}_\Omega$ will contain an edge $(\check{\text{vx}}_1, \check{\text{vx}}_2)$.

G Grounded Dictionaries

The pseudocode for the grounded dictionary functionality is in Figure 6, for the grounded encrypted dictionary scheme is in Figure 7 and for the grounded dictionary leakage is in Figure 8. The security of the scheme is captured in the following Theorem.

Theorem 1. *If Σ_{DX} is $\mathcal{L}^{\delta}_{\text{DX}}$-secure and* KeyAgree *is a secure key agreement protocol, Σ_{GDX} as described in Figure 7 is $\mathcal{L}^{\delta}_{\text{GDX}}$-secure in the $(\mathcal{F}_{\text{prox}}, \mathcal{F}_{\text{anon}})$-hybrid model.*

Proof. Let $\mathcal{S}_{\text{DX}}$ be the simulator guaranteed to exist by the $\mathcal{L}^{\delta}_{\text{DX}}$-security of Σ_{DX} and $\mathcal{S}_{\text{KA}}$ be the simulator guaranteed to exist by the

Functionality $\mathcal{F}^{\mathcal{L}}_{\mathsf{GDX}}$

The functionality is parameterized with a leakage profile $\mathcal{L}$, a distance $\delta \in \mathbb{R}_{\geq 0}$, identification functions (φ, γ) and interacts with n clients $\mathbf{C}_1, \ldots, \mathbf{C}_n$, a trusted party $\mathbf{T}$, a server $\mathbf{S}$ and an ideal adversary $\mathcal{S}$. It stores and manages a set of dictionaries and a location function λ using the following operations:

$\mathsf{SetLoc}_{\mathbf{T}}(\lambda)$:
(1) set $\lambda := \lambda$;

$\mathsf{Init}_{\mathbf{C}_a,\mathbf{C}_b,\mathbf{S}}(\varphi_\beta, \mathsf{dxid}, \mathbb{L}, \mathbb{V}; \varphi_\alpha; \perp)$:
(1) set $\mathsf{leak}_{\mathcal{S}} := \mathcal{L}(\mathsf{init}, \lambda, \mathtt{i}_a, \mathtt{i}_b, \varphi_\alpha, \varphi_\beta, \mathsf{dxid}, \mathbb{L}, \mathbb{V})$;
(2) if $\mathcal{D}(\lambda_a, \lambda_b) \leq \delta$, $\varphi_b = \varphi_\beta$ and $\varphi_a = \varphi_\alpha$,
  (a) set $\mathsf{grounded}_{\mathsf{dxid}} := \{\mathtt{i}_a, \mathtt{i}_b\}$;
  (b) initialize $\mathsf{DX}_{\mathsf{dxid}} : \mathbb{L} \to \mathbb{V}$;
  (c) output $(\perp; \mathsf{dxid}; \perp; \mathsf{leak}_{\mathcal{S}})$;
(3) else, output $(\perp_A; \perp_A; \perp_A; \mathsf{leak}_{\mathcal{S}})$;

$\mathsf{Put}_{\mathbf{C}_a,\mathbf{S}}(\mathsf{dxid}, \ell, v; \perp)$:
(1) set $\mathsf{leak}_{\mathcal{S}} := \mathcal{L}(\mathsf{put}, \lambda, \mathtt{i}_a, \mathsf{dxid}, \ell, v)$;
(2) if $\mathtt{i}_a \in \mathsf{grounded}_{\mathsf{dxid}}$,
  (a) set $\mathsf{DX}_{\mathsf{dxid}}[\ell] := v$;
  (b) output $(\perp; \perp; \mathsf{leak}_{\mathcal{S}})$;
(3) else, output $(\perp_A; \perp_A; \mathsf{leak}_{\mathcal{S}})$;

$\mathsf{Get}_{\mathbf{C}_a,\mathbf{S}}(\mathsf{dxid}, \ell; \perp)$:
(1) set $\mathsf{leak}_{\mathcal{S}} := \mathcal{L}(\mathsf{get}, \lambda, \mathtt{i}_a, \mathsf{dxid}, \ell)$;
(2) if $\mathtt{i}_a \in \mathsf{grounded}_{\mathsf{dxid}}$,
  (a) compute $v := \mathsf{DX}_{\mathsf{dxid}}[\ell]$;
  (b) output $(v; \perp; \mathsf{leak}_{\mathcal{S}})$;
(3) else, output $(\perp_A; \perp_A; \mathsf{leak}_{\mathcal{S}})$;

$\mathsf{Delete}_{\mathbf{C}_a,\mathbf{S}}(\mathsf{dxid}, \ell; \perp)$:
(1) set $\mathsf{leak}_{\mathcal{S}} := \mathcal{L}(\mathsf{delete}, \lambda, \mathtt{i}_a, \mathsf{dxid}, \ell)$;
(2) if $\mathtt{i}_a \in \mathsf{grounded}_{\mathsf{dxid}}$,
  (a) compute $\mathsf{DX}_{\mathsf{dxid}} - \ell$;
  (b) output $(\perp; \perp; \mathsf{leak}_{\mathcal{S}})$;
(3) else, output $(\perp_A; \perp_A; \mathsf{leak}_{\mathcal{S}})$;

Figure 6: Grounded Dictionary Functionality

security of KeyAgree. Recall that δ is a public parameter. Consider the simulator $\mathcal{S}$ that simulates $\mathcal{A}$ and that works as follows:

- simulating Init: given $(\check{\mathsf{vx}}, \check{\mathbf{E}})$, parse $\check{\mathsf{vx}}$ as

  $$[\![\, \Sigma_{\mathsf{GDX}}, t, \mathsf{init}, \lambda_a, \varphi_a, \varphi_\alpha, \lambda_b, \varphi_b, \varphi_\beta, \mathsf{dxid}\,;\, \mathsf{vx} \,]\!]$$

  and $\check{\mathbf{E}}$ as $\mathbf{E}$. Simulate the key agreement between $\mathbf{C}_a$ and $\mathbf{C}_b$ by computing

  $$(K_1, K_2; K_1, K_2; \mathsf{leak}_{\mathcal{A}}) \leftarrow \mathcal{F}_{\mathsf{prox}}[\varphi_\beta, \varphi_\alpha] \circ \mathcal{S}_{\mathsf{KA}}(1^{2k})$$

  using λ_a and λ_b as the locations of $\mathbf{C}_a$ and $\mathbf{C}_b$ and send $\mathsf{leak}_{\mathcal{A}}$ to $\mathcal{A}$. If the proximity channel aborts, send $\perp_A$ to $\mathbf{S}$ via $\mathcal{F}_{\mathsf{anon}}$ and abort. Otherwise, use $\mathcal{F}_{\mathsf{anon}} \circ \mathcal{S}_{\mathsf{DX}}(\mathsf{init}, \mathsf{vx}, \mathbf{E})$ to simulate a $\mathcal{F}_{\mathsf{anon}} \circ \Sigma_{\mathsf{DX}}.\mathsf{Init}_{\mathbf{C}_a,\mathbf{S}}$ execution with $\mathcal{A}$. Compute $\mathsf{ct}_a \leftarrow \mathsf{Enc}_{K_2}\left(0^{|\mathsf{dxst}_0|}\right)$, where $|\mathsf{dxst}_0|$ is the size of an empty dictionary state, and send ct_a to $\mathbf{C}_b$ via

  $$\mathcal{F}_{\mathsf{prox}}.\mathsf{Send}_{\mathbf{C}_a,\mathbf{C}_b}(\mathsf{ct}_a; \varphi_a).$$

  Finally, output whatever $\mathcal{A}$ outputs.

Grounded Encrypted Dictionary

Given an encrypted dictionary scheme $\Sigma_{\mathsf{DX}} = (\mathsf{KeyGen}, \mathsf{Init}, \mathsf{Put}, \mathsf{Get}, \mathsf{Delete})$ and a key agreement protocol KeyAgree, construct the grounded encrypted dictionary $\Sigma_{\mathsf{GDX}} = (\mathsf{Init}, \mathsf{Put}, \mathsf{Get}, \mathsf{Delete})$ where Put, Get and Delete are as in Σ_{DX}, with the exception that all messages are sent to the server are via $\mathcal{F}_{\mathsf{anon}}$, and Init is as follows:

$\mathsf{Init}_{\mathbf{C}_a,\mathbf{C}_b,\mathbf{S}}(\varphi_\beta, \mathsf{dxid}, \mathbb{L}, \mathbb{V}; \varphi_\alpha; \perp)$:
(1) $\mathbf{C}_a$ and $\mathbf{C}_b$ execute
  $(\mathsf{out}'_a; \mathsf{out}'_b) \leftarrow \mathcal{F}_{\mathsf{prox}}[\varphi_\beta, \varphi_\alpha] \circ \mathsf{KeyAgree}_{\mathbf{C}_a,\mathbf{C}_b}(1^{2k}; 1^{2k})$;
(2) if $\mathsf{out}'_a = \mathsf{out}'_b = \perp_A$,
  (a) $\mathbf{C}_a$ sends $\perp_A$ to $\mathbf{S}$ via $\mathcal{F}_{\mathsf{anon}}$;
  (b) output $(\perp_A; \perp_A; \perp_A)$;
(3) else,
  (a) $\mathbf{C}_a$ parses out'_a as (K_1, K_2);
  (b) $\mathbf{C}_a$ and $\mathbf{S}$ execute
  $(\mathsf{dxst}; \mathsf{EDX}) \leftarrow \mathcal{F}_{\mathsf{anon}} \circ \Sigma_{\mathsf{DX}}.\mathsf{Init}_{\mathbf{C}_a,\mathbf{S}}(K_1, \mathsf{dxid}, \mathbb{L}, \mathbb{V}; \perp)$;
  (c) $\mathbf{C}_a$ computes $\mathsf{ct}_a \leftarrow \mathsf{Enc}_{K_2}(\mathsf{dxst})$;
  (d) $\mathbf{C}_a$ and $\mathbf{C}_b$ execute
  $(\perp; \mathsf{ct}_a) \leftarrow \mathcal{F}_{\mathsf{prox}}.\mathsf{Send}_{\mathbf{C}_a,\mathbf{C}_b}(\mathsf{ct}_a; \varphi_\alpha)$
  (e) $\mathbf{C}_b$ parses out'_b as (K_1, K_2);
  (f) $\mathbf{C}_b$ computes $\mathsf{dxst} := \mathsf{Dec}_{K_2}(\mathsf{ct}_a)$;
  (g) output $(\mathsf{dxst}, K_1; \mathsf{dxst}, K_1; \mathsf{EDX})$;

Figure 7: Grounded Encrypted Dictionary

Leakage $\mathcal{L}^{\delta}_{\mathsf{GDX}}$

$\mathcal{L}^{\delta}_{\mathsf{GDX}}(\mathsf{op})$:
(1) if $\mathsf{op} = (\mathsf{init}, \lambda, \mathtt{i}_a, \mathtt{i}_b, \varphi_\alpha, \varphi_\beta, \mathsf{dxid}, \mathbb{L}, \mathbb{V})$,
  (a) set $t := 0$;
  (b) initialize a dictionary sDX;
  (c) compute $(\mathsf{vx}, \mathbf{E}) := \mathcal{L}_{\mathsf{DX}}(\mathsf{init}, \mathsf{dxid}, \mathbb{L}, \mathbb{V})$;
  (d) set $\check{\mathsf{vx}} := [\![\, \Sigma_{\mathsf{GDX}}, \mathsf{init}, t, \lambda_a, \varphi_a, \varphi_\alpha, \lambda_b, \varphi_b, \varphi_\beta, \mathsf{dxid}\,;\, \mathsf{vx} \,]\!]$;
  (e) set $\check{\mathbf{E}} := \emptyset$;
(2) if $\mathsf{op} = (\mathsf{put}, \lambda, \mathtt{i}_a, \mathsf{dxid}, \ell, v)$,
  (a) set $t := t + 1$;
  (b) compute $(\mathsf{vx}, \mathbf{E}) := \mathcal{L}_{\mathsf{DX}}(\mathsf{put}, \mathsf{dxid}, \ell, v)$;
  (c) set $\check{\mathsf{vx}} := [\![\, \Sigma_{\mathsf{GDX}}, \mathsf{put}, t, \mathsf{dxid}\,;\, \mathsf{vx} \,]\!]$;
  (d) set $\check{\mathbf{E}} := \mathbf{E}$;
(3) if $\mathsf{op} = (\mathsf{get}, \lambda, \mathsf{dxid}, \mathtt{i}_a, \ell)$,
  (a) set $t := t + 1$;
  (b) compute $(\mathsf{vx}, \mathbf{E}) := \mathcal{L}_{\mathsf{DX}}(\mathsf{get}, \mathsf{dxid}, \ell)$;
  (c) set $\check{\mathsf{vx}} := [\![\, \Sigma_{\mathsf{GDX}}, \mathsf{get}, t, \mathsf{dxid}\,;\, \mathsf{vx} \,]\!]$;
  (d) set $\check{\mathbf{E}} := \mathbf{E}$;
(4) if $\mathsf{op} = (\mathsf{delete}, \lambda, \mathsf{dxid}, \mathtt{i}_a, \ell)$,
  (a) set $t := t + 1$;
  (b) compute $(\mathsf{vx}, \mathbf{E}) := \mathcal{L}_{\mathsf{DX}}(\mathsf{delete}, \mathsf{dxid}, \ell)$;
  (c) set $\check{\mathsf{vx}} := [\![\, \Sigma_{\mathsf{GDX}}, \mathsf{delete}, t, \mathsf{dxid}\,;\, \mathsf{vx} \,]\!]$;
  (d) set $\check{\mathbf{E}} := \mathbf{E}$;
(5) output $(\check{\mathsf{vx}}, \check{\mathbf{E}})$;

Figure 8: Leakage $\mathcal{L}^{\delta}_{\mathsf{GDX}}$

- simulating Put: given $(\check{\mathsf{vx}}, \check{\mathbf{E}})$, parse $\check{\mathsf{vx}}$ as

  $$[\![\, \Sigma_{\mathsf{GDX}}, \mathsf{put}, t, \mathsf{dxid}\,;\, \mathsf{vx} \,]\!]$$

and $\check{\mathbf{E}}$ as $\mathbf{E}$ and use $\mathcal{F}_{\text{anon}} \circ \mathcal{S}_{\text{DX}}(\text{put}, \text{vx}, \mathbf{E})$ to simulate a $\mathcal{F}_{\text{anon}} \circ \Sigma_{\text{DX}}.\text{Put}_{\text{C}_a,\text{S}}$ execution and output whatever it outputs.

- simulating Get: given $(\check{\text{vx}}, \check{\mathbf{E}})$, parse $\check{\text{vx}}$ as

$$[\![\Sigma_{\text{GDX}}, \text{get}, t, \text{dxid} ; \text{vx}]\!]$$

and $\check{\mathbf{E}}$ as $\mathbf{E}$ and use $\mathcal{F}_{\text{anon}} \circ \mathcal{S}_{\text{DX}}(\text{get}, \text{vx}, \mathbf{E})$ to simulate a $\mathcal{F}_{\text{anon}} \circ \Sigma_{\text{DX}}.\text{Get}_{\text{C}_a,\text{S}}$ execution and output whatever it outputs.

- simulating Delete: given $(\check{\text{vx}}, \check{\mathbf{E}})$, parse $\check{\text{vx}}$ as

$$[\![\Sigma_{\text{GDX}}, \text{delete}, t, \text{dxid} ; \text{vx}]\!]$$

and $\check{\mathbf{E}}$ as $\mathbf{E}$ and use $\mathcal{F}_{\text{anon}} \circ \mathcal{S}_{\text{DX}}(\text{delete}, \text{vx}, \mathbf{E})$ to simulate a $\mathcal{F}_{\text{anon}} \circ \Sigma_{\text{DX}}.\text{Delete}_{\text{C}_a,\text{S}}$ execution and output whatever it outputs.

It remains to show that for all p.p.t. adversaries $\mathcal{A}$, all p.p.t. environments $\mathcal{Z}$, all $z \in \{0,1\}^k$ and all $\delta \in \mathbb{R}_{\geq 0}$,

$$\Big| \Pr[\, \mathcal{Z}(\mathcal{A}, \mathbf{T}, \mathbf{C}_1, \ldots, \mathbf{C}_n :: \Sigma_{\text{GDX}}) = 1 \,] - \Pr[\, \mathcal{Z}(\mathcal{S}, \mathbf{T}, \mathbf{C}_1, \ldots, \mathbf{C}_n :: \mathcal{F}_{\text{GDX}}^{\mathcal{L}}) = 1 \,] \Big| \leq \text{negl}(k).$$

We do this using the following sequence of games:

- Game_0: is a $\mathcal{Z}(\mathcal{A}, \mathbf{T}, \mathbf{C}_1, \ldots, \mathbf{C}_n :: \Sigma_{\text{GDX}})$ experiment;
- Game_1: is the same as Game_0 except for the following. Given leakage

$$\check{\text{vx}} = [\![\Sigma_{\text{GDX}}, t, \text{init}, \lambda_a, \varphi_a, \varphi_\alpha, \lambda_b, \varphi_b, \varphi_\beta, \text{dxid} ; \text{vx}]\!]$$

the $\mathcal{F}_{\text{prox}}[\varphi_\beta, \varphi_\alpha]$ functionality is simulated using λ_a and λ_b, and φ_a and φ_b as $\mathbf{C}_a$ and $\mathbf{C}_b$'s locations and physical identities, respectively. Note that the views of $\mathcal{Z}$ and $\mathcal{A}$ in Game_1 are exactly the same as in Game_0, from which it follows that the probabilities that Game_0 and Game_1 output 1 are the same.
- Game_2: is the same as Game_1 except that the KeyAgree execution is replaced with an execution of $\mathcal{S}_{\text{KA}}\left(1^{2k}\right)$. Note that the security of KeyAgree guarantees that the views of $\mathcal{Z}$ and $\mathcal{A}$ in Game_1 and in Game_2 are computationally indistinguishable which implies that the probabilities that Game_1 and Game_2 output 1 are negligibly-close.
- Game_3: is the same as Game_2 except that ct_a is replaced with $\text{Enc}_{K_2}\left(0^{|\text{dxst}_0|}\right)$. Note that the security of SKE guarantees that the views of $\mathcal{Z}$ and $\mathcal{A}$ in Game_2 and Game_3 are computationally indistinguishable which implies that the probabilities that Game_2 and Game_3 output 1 are negligibly-close.
- Game_4: is the same as Game_3 except that
  - $\mathcal{F}_{\text{anon}} \circ \Sigma_{\text{DX}}.\text{Init}_{\text{C}_a,\text{S}}(K_1, \text{dxid}, \mathbb{L}, \mathbb{V}; \perp)$ is replaced with $\mathcal{F}_{\text{anon}} \circ \mathcal{S}_{\text{DX}}(\text{init}, \text{vx}, \mathbf{E})$, where $(\text{vx}, \mathbf{E}) := \mathcal{L}_{\text{DX}}(\text{init}, \text{dxid}, \mathbb{L}, \mathbb{V})$;
  - $\mathcal{F}_{\text{anon}} \circ \Sigma_{\text{DX}}.\text{Put}_{\text{C}_a,\text{S}}(K_1, \text{dxid}, \ell, v; \text{EDX})$ executions are replaced with $\mathcal{F}_{\text{anon}} \circ \mathcal{S}_{\text{DX}}(\text{put}, \text{vx}, \mathbf{E})$ executions where $(\text{vx}, \mathbf{E}) := \mathcal{L}_{\text{DX}}(\text{put}, \text{dxid}, \ell, v)$;
  - $\mathcal{F}_{\text{anon}} \circ \Sigma_{\text{DX}}.\text{Get}_{\text{C}_a,\text{S}}(K_1, \text{dxid}, \ell; \text{EDX})$ executions are replaced with $\mathcal{F}_{\text{anon}} \circ \mathcal{S}_{\text{DX}}(\text{get}, \text{vx}, \mathbf{E})$ executions where $(\text{vx}, \mathbf{E}) := \mathcal{L}_{\text{DX}}(\text{get}, \text{dxid}, \ell)$;
  - $\mathcal{F}_{\text{anon}} \circ \Sigma_{\text{DX}}.\text{Delete}_{\text{C}_a,\text{S}}(K_1, \text{dxid}, \ell; \text{EDX})$ executions are replaced with $\mathcal{F}_{\text{anon}} \circ \mathcal{S}_{\text{DX}}(\text{delete}, \text{vx}, \mathbf{E})$ executions where $(\text{vx}, \mathbf{E}) := \mathcal{L}_{\text{DX}}(\text{delete}, \text{dxid}, \ell)$;

Note that the $\mathcal{L}_{\text{DX}}$-security of Σ_{DX} guarantees that the views of $\mathcal{Z}$ and $\mathcal{A}$ in Game_3 and Game_4 are computationally indistinguishable which implies that the probabilities that they output 1 are negligibly-close probability.

The Theorem follows by a standard probabilistic argument.

■

H Encrypted Box Bank

The pseudocode for the box bank functionality is in Figure 9, for the encrypted box bank scheme is in Figures 10 and 11, and for the box bank leakage is in Figure 12. The security of the scheme is captured in the following Theorem.

Theorem 2. *If Σ_{GDX} is $\mathcal{L}_{\text{GDX}}^{\delta}$-secure, then Σ_{BXB} as described in Figures 10 and 11 is $\mathcal{L}_{\text{BXB}}^{\delta}$-secure in the $(\mathcal{F}_{\text{anon}}, \mathcal{F}_{\text{id}})$-hybrid model.*

Proof. Let $\mathcal{S}_{\text{GDX}}$ be the simulator guaranteed to exist by the $\mathcal{L}_{\text{GDX}}^{\delta}$-security of Σ_{GDX} and consider the simulator that simulates $\mathcal{A}$ as follows:

- simulating Init: given $(\check{\text{vx}}, \check{\mathbf{E}})$, initialize a dictionary sDX and set bxrank := 0;
- simulating CreateBox: given $(\check{\text{vx}}, \check{\mathbf{E}})$, parse $\check{\text{vx}}$ as

$$[\![\Sigma_{\text{BXB}}, \text{crtbx}, t, \text{bxrank}, \varphi_a, \varphi_b ; \text{vx}]\!],$$

and $\check{\mathbf{E}}$ as $\mathbf{E}$. Simulate a

$$(\text{bxid}; \text{bxid}; \text{bxid}; \text{leak}_{\mathcal{A}}) \leftarrow \mathcal{F}_{\text{id}}.\text{GetId}_{\text{C}_a,\text{C}_b,\text{S}}(1^k, \Sigma_{\text{BXB}}; 1^k, \Sigma_{\text{BXB}}; \perp)$$

execution, compute bxrank := bxrank+1 and set sDX[bxrank] := bxid. Then, use $\mathcal{S}_{\text{GDX}}(\text{init}, \text{vx}, \mathbf{E})$ to simulate a $\Sigma_{\text{GDX}}.\text{Init}$ execution with the server S and output whatever $\mathcal{A}$ outputs.

- simulating DropMail: given $(\text{vx}, \mathbf{E})$, parse vx as

$$[\![\Sigma_{\text{BXB}}, \text{drpmail}, t, \text{bxrank} ; \text{vx}]\!],$$

and $\check{\mathbf{E}}$ as $\mathbf{E}$. Compute bxid := sDX[bxrank] and send bxid to S via $\mathcal{F}_{\text{anon}}$. Use $\mathcal{S}_{\text{GDX}}(\text{put}, \text{vx}, \mathbf{E})$ to simulate a $\Sigma_{\text{GDX}}.\text{Put}$ execution with S and output whatever $\mathcal{A}$ outputs.

- simulating GetMail: given $(\check{\text{vx}}, \check{\mathbf{E}})$, parse $\check{\text{vx}}$ as

$$[\![\Sigma_{\text{BXB}}, \text{gtmail}, t, \text{bxrank} ; \text{vx}]\!],$$

and $\check{\mathbf{E}}$ as $\mathbf{E}$. Compute bxid := sDX[bxrank] and send bxid to S via $\mathcal{F}_{\text{anon}}$. Use $\mathcal{S}_{\text{GDX}}(\text{get}, \text{vx}, \mathbf{E})$ to simulate a $\Sigma_{\text{GDX}}.\text{Get}$ execution with S and output whatever $\mathcal{A}$ outputs.

- simulating DeleteMail: given $(\check{\text{vx}}, \check{\mathbf{E}})$, parse $\check{\text{vx}}$ as

$$[\![\Sigma_{\text{BXB}}, \text{delmail}, t, \text{bxrank} ; \text{vx}]\!],$$

and $\check{\mathbf{E}}$ as $\mathbf{E}$. Compute bxid := sDX[bxrank] and send bxid to S via $\mathcal{F}_{\text{anon}}$. Use $\mathcal{S}_{\text{GDX}}(\text{delete}, \text{vx}, \mathbf{E})$ to simulate a $\Sigma_{\text{GDX}}.\text{Delete}$ execution with S and output whatever $\mathcal{A}$ outputs.

It remains to show that for all p.p.t. adversaries $\mathcal{A}$, all p.p.t. environments $\mathcal{Z}$, all $z \in \{0,1\}^k$ and all $\delta \in \mathbb{R}_{\geq 0}$,

$$\Big| \Pr[\, \mathcal{Z}(\mathcal{A}, \mathbf{T}, \mathbf{C}_1, \ldots, \mathbf{C}_n :: \Sigma_{\text{BXB}}) = 1 \,] - \Pr[\, \mathcal{Z}(\mathcal{S}, \mathbf{T}, \mathbf{C}_1, \ldots, \mathbf{C}_n :: \mathcal{F}_{\text{BXB}}^{\mathcal{L}}) = 1 \,] \Big| \leq \text{negl}(k). \quad (1)$$

We do this using the following sequence of games:

Functionality $\mathcal{F}_{\mathsf{BXB}}^{\mathcal{L}}$

The functionality is parameterized with a leakage profile $\mathcal{L}$, a distance $\delta \in \mathbb{R}_{\geq 0}$ and identification functions (φ, γ). It interacts with n clients $\mathbf{C}_1, \ldots, \mathbf{C}_n$, a trusted party $\mathbf{T}$, a server $\mathbf{S}$ and an ideal adversary $\mathcal{S}$. It stores and manages a dictionary BXB and a location function λ using the following operations:

$\mathsf{SetLoc}_{\mathbf{T}}(\lambda)$:
(1) set $\lambda := \lambda$;

$\mathsf{Init}_{\mathbf{S}}(\perp)$:
(1) initialize a dictionary BXB;
(2) initialize a dictionary sDX;
(3) set $\mathsf{leak}_{\mathcal{S}} := \mathcal{L}(\mathsf{init}, \lambda)$;
(4) output $(\perp; \mathsf{leak}_{\mathcal{S}})$;

$\mathsf{CreateBox}_{\mathbf{C}_a,\mathbf{C}_b,\mathbf{S}}(\varphi_\beta, \mathbb{L}, \mathbb{V}; \varphi_\alpha; \perp)$:
(1) sample bxid $\leftarrow\$ \{0,1\}^k$;
(2) set $\mathsf{leak}_{\mathcal{S}} := \mathcal{L}(\mathsf{crtbx}, \lambda, \mathsf{i}_a, \mathsf{i}_b, \varphi_\beta, \varphi_\alpha, \mathsf{bxid}, \mathbb{L}, \mathbb{V})$;
(3) if $\mathcal{D}(\lambda_a, \lambda_b) \leq \delta$, $\varphi_\beta = \varphi_b$ and $\varphi_\alpha = \varphi_a$,
  (a) initialize an empty dictionary $\mathsf{DX} : \mathbb{L} \to \mathbb{V}$;
  (b) set $\mathsf{sDX}[\mathsf{bxid}] := \{\mathsf{i}_a, \mathsf{i}_b\}$;
  (c) set $\mathsf{BXB}[\mathsf{bxid}] := \mathsf{DX}$;
  (d) output $(\mathsf{bxid}; \mathsf{bxid}; \mathsf{bxid}; \mathsf{leak}_{\mathcal{S}})$;
(4) else, set output $(\perp_A; \perp_A; \perp_A; \mathsf{leak}_{\mathcal{S}})$;

$\mathsf{DropMail}_{\mathbf{C}_a,\mathbf{S}}(\mathsf{bxid}, \ell, m; \perp)$:
(1) set $\mathsf{leak}_{\mathcal{S}} := \mathcal{L}(\mathsf{drpmail}, \lambda, \mathsf{i}_a, \mathsf{bxid}, \ell, m)$;
(2) if $\mathsf{i}_a \in \mathsf{sDX}[\mathsf{bxid}]$ and $\mathsf{BXB}[\mathsf{bxid}] \neq \perp$,
  (a) compute $\mathsf{DX} := \mathsf{BXB}[\mathsf{bxid}]$;
  (b) set $\mathsf{DX}[\ell] := m$;
  (c) set $\mathsf{BXB}[\mathsf{bxid}] := \mathsf{DX}$;
(3) output $(\perp; \perp; \mathsf{leak}_{\mathcal{S}})$;

$\mathsf{GetMail}_{\mathbf{C}_a,\mathbf{S}}(\mathsf{bxid}, \ell; \perp)$:
(1) set $\mathsf{leak}_{\mathcal{S}} := \mathcal{L}(\mathsf{gtmail}, \lambda, \mathsf{i}_a, \mathsf{bxid}, \ell)$;
(2) if $\mathsf{i}_a \in \mathsf{sDX}[\mathsf{bxid}]$ and $\mathsf{BXB}[\mathsf{bxid}] \neq \perp$,
  (a) compute $\mathsf{DX} := \mathsf{BXB}[\mathsf{bxid}]$;
  (b) compute $m := \mathsf{DX}[\ell]$;
  (c) output $(m; \perp; \mathsf{leak}_{\mathcal{S}})$;
(3) else, output $(\perp; \perp; \mathsf{leak}_{\mathcal{S}})$;

$\mathsf{DeleteMail}_{\mathbf{C}_a,\mathbf{S}}(\mathsf{bxid}, \ell; \perp)$:
(1) set $\mathsf{leak}_{\mathcal{S}} := \mathcal{L}(\mathsf{delmail}, \lambda, \mathsf{i}_a, \mathsf{bxid}, \ell)$;
(2) if $\mathsf{i}_a \in \mathsf{sDX}[\mathsf{bxid}]$ and $\mathsf{BXB}[\mathsf{bxid}] \neq \perp$,
  (a) set $\mathsf{DX} := \mathsf{BXB}[\mathsf{bxid}]$;
  (b) compute $\mathsf{DX} - \ell$;
  (c) set $\mathsf{BXB}[\mathsf{bxid}] := \mathsf{DX}$;
(3) output $(\perp; \perp; \mathsf{leak}_{\mathcal{S}})$

Figure 9

Encrypted Box Bank (part 1)

Given an encrypted grounded dictionary scheme $\Sigma_{\mathsf{GDX}} = (\mathsf{Init}, \mathsf{Put}, \mathsf{Get}, \mathsf{Delete})$, construct an encrypted box bank scheme $\Sigma_{\mathsf{BXB}} = (\mathsf{Init}, \mathsf{CreateBox}, \mathsf{DropMail}, \mathsf{GetMail}, \mathsf{DeleteMail})$ as follows:

$\mathsf{Init}_{\mathsf{S}}(\perp)$:
(1) S initializes a dictionary EBB;
(2) output EBB

$\mathsf{CreateBox}_{\mathbf{C}_a,\mathbf{C}_b,\mathbf{S}}(\varphi_\beta, \mathbb{L}, \mathbb{V}; \varphi_\alpha; \mathsf{EBB})$:
(1) $\mathbf{C}_a$ and $\mathbf{C}_b$ execute
$(\mathsf{bxid}; \mathsf{bxid}; \mathsf{bxid}) \leftarrow \mathcal{F}_{\mathsf{id}}.\mathsf{GetId}_{\mathbf{C}_a,\mathbf{C}_b,\mathbf{S}}(1^k, \Sigma_{\mathsf{BXB}}; 1^k, \Sigma_{\mathsf{BXB}}; 1^k)$;
(2) $\mathbf{C}_a$, $\mathbf{C}_b$ and S run
$(\mathsf{out}_a; \mathsf{out}_b; \mathsf{out}_{\mathsf{S}}) \leftarrow \Sigma_{\mathsf{GDX}}.\mathsf{Init}_{\mathbf{C}_a,\mathbf{C}_b,\mathbf{S}}(\varphi_\beta, \mathsf{bxid}, \mathbb{L}, \mathbb{V}; \varphi_\alpha; \perp)$;
(3) if $\mathsf{out}_a = \mathsf{out}_b = \mathsf{out}_{\mathsf{S}} = \perp_A$,
  (a) output $(\perp_A; \perp_A; \perp_A)$;
(4) else,
  (a) $\mathbf{C}_a$ parses out_a as $(\mathsf{dxst}, K_{\mathsf{GDX}})$;
  (b) $\mathbf{C}_a$ sets $\mathsf{bxst}_a := (\mathsf{dxst}, \mathsf{bxid})$;
  (c) $\mathbf{C}_a$ sets $K_{\mathsf{BXB}} := K_{\mathsf{GDX}}$;
  (d) $\mathbf{C}_b$ parses out_b as $(\mathsf{dxst}, K_{\mathsf{GDX}})$;
  (e) $\mathbf{C}_b$ sets $\mathsf{bxst}_b := (\mathsf{dxst}, \mathsf{bxid})$;
  (f) $\mathbf{C}_b$ sets $K_{\mathsf{BXB}} := K_{\mathsf{GDX}}$;
  (g) S parses $\mathsf{out}_{\mathsf{S}}$ as $(\mathsf{EDX}, \mathsf{dxid})$;
  (h) S sets $\mathsf{bxid} := \mathsf{dxid}$;
  (i) S sets $\mathsf{EBB}[\mathsf{bxid}] := \mathsf{EDX}$;
  (j) output $(\mathsf{bxst}_a, K_{\mathsf{BXB}}, \mathsf{bxid}; \mathsf{bxst}_b, K_{\mathsf{BXB}}, \mathsf{bxid}; \mathsf{EBB}, \mathsf{bxid})$;

Figure 10

- Game_0: is the same as a $\mathcal{Z}(\mathcal{A}, \mathbf{T}, \mathbf{C}_1, \ldots, \mathbf{C}_n :: \Sigma_{\mathsf{BXB}})$ experiment.
- $\mathsf{Game}_{i \in [n]}$: is the same as Game_{i-1} except that the session of the ith box is replaced with a simulated session generated by applying $\mathcal{S}_{\mathsf{GDX}}$ on the $\mathcal{L}_{\mathsf{GDX}}$ leakage of each operation of that session.

We argue that for Game_i, if there exists a p.p.t. adversary $\mathcal{A}$, a p.p.t. environment $\mathcal{Z}$, a string $z \in \{0,1\}^*$ and a distance $\delta \in \mathbb{R}_{\geq 0}$ such that the probability that Game_{i-1} and Game_i output 1 is non-negligibly close, then there exists a p.p.t. adversary $\mathcal{A}_{\mathsf{GDX}}$ such that for all p.p.t. simulators $\mathcal{S}_{\mathsf{GDX}}$, there exists a p.p.t. environment $\mathcal{Z}_{\mathsf{GDX}}$, a string $z_{\mathsf{GDX}} \in \{0,1\}^*$ and a distance $\delta_{\mathsf{GDX}} \in \mathbb{R}_{\geq 0}$ such that the $\mathcal{L}_{\mathsf{GDX}}^{\delta_{\mathsf{GDX}}}$-security of Σ_{GDX} is violated. Let $\mathcal{A}_{\mathsf{GDX}}$ be the adversary that forwards messages back and forth, $z_{\mathsf{GDX}} = z$ and $\delta_{\mathsf{GDX}} = \delta$. We denote by χ the party that is either $\mathcal{A}_{\mathsf{GDX}}$ or $\mathcal{S}_{\mathsf{GDX}}$. $\mathcal{Z}_{\mathsf{GDX}}$ simulates $\mathcal{Z}$ and $\mathcal{A}$ and does the following:

(1) when $\mathcal{Z}$ outputs (init, λ), $\mathcal{Z}_{\mathsf{GDX}}$ does nothing;
(2) when $\mathcal{Z}$ outputs $(\varphi, \gamma, \lambda, \mathbf{C}_a, \varphi_\beta, \mathbf{C}_b, \varphi_\alpha, \mathsf{crtbx}, \mathbb{L}, \mathbb{V})$, $\mathcal{Z}_{\mathsf{GDX}}$ simulates
$(\mathsf{bxid}; \mathsf{bxid}; \mathsf{bxid}) \leftarrow \mathcal{F}_{\mathsf{id}}.\mathsf{GetId}_{\mathbf{C}_a,\mathbf{C}_b,\mathbf{S}}(1^k, \Sigma_{\mathsf{BXB}}; 1^k, \Sigma_{\mathsf{BXB}}; 1^k)$
and outputs $(\varphi, \gamma, \lambda, \mathbf{C}_a, \varphi_\beta, \mathbf{C}_b, \varphi_\alpha, \mathsf{init}, \mathsf{dxid}, \mathbb{L}, \mathbb{V})$, where $\mathsf{dxid} := \mathsf{bxid}$. It then sends $(\mathsf{bxid}; \mathsf{bxid}; \mathsf{bxid})$ back to $\mathcal{Z}_{\mathsf{BXB}}$ while forwarding messages back and forth between χ and $\mathcal{A}$ and $\mathcal{A}$ and $\mathcal{Z}$.
(3) when $\mathcal{Z}$ outputs $(\mathbf{C}_a, \mathsf{drpmail}, \mathsf{bxid}, \ell, m)$, $\mathcal{Z}_{\mathsf{GDX}}$ sends bxid to S via $\mathcal{F}_{\mathsf{anon}}$ and outputs $(\mathbf{C}_a, \mathsf{put}, \mathsf{dxid}, \ell, m)$ where $\mathsf{dxid} := \mathsf{bxid}$, while forwarding messages back and forth between χ and $\mathcal{A}$ and $\mathcal{A}$ and $\mathcal{Z}$. It then sends $(\perp, \perp)$ to $\mathcal{Z}$.
(4) when $\mathcal{Z}$ outputs $(\mathbf{C}_a, \mathsf{gtmail}, \mathsf{bxid}, \ell)$, $\mathcal{Z}_{\mathsf{GDX}}$ sends bxid to S via $\mathcal{F}_{\mathsf{anon}}$ and outputs $(\mathbf{C}_a, \mathsf{get}, \mathsf{dxid}, \ell)$, where $\mathsf{dxid} := \mathsf{bxid}$ and forwards messages between χ and $\mathcal{A}$ and $\mathcal{A}$ and $\mathcal{Z}$. After receiving m it sends $(m; \perp)$ to

Encrypted Box Bank (part 2)

$\mathsf{DropMail}_{\mathbf{C}_a,\mathbf{S}}(\mathsf{bxst}, K_{\mathsf{BXB}}, \ell, m; \mathsf{EBB})$;
(1) $\mathbf{C}_a$ parses bxst as (dxst, bxid);
(2) $\mathbf{C}_a$ parses K_{BXB} as K_{GDX};
(3) $\mathbf{C}_a$ sends bxid to S via $\mathcal{F}_{\mathsf{anon}}$;
(4) if $\mathsf{EBB}[\mathsf{bxid}] \neq \perp$,
(a) S computes EDX := EBB[bxid];
(b) $\mathbf{C}_a$ and S run
$(\mathsf{dxst}; \mathsf{EDX}) \leftarrow \Sigma_{\mathsf{GDX}}.\mathsf{Put}_{\mathbf{C}_a,\mathbf{S}}(\mathsf{dxst}, K_{\mathsf{GDX}}, \mathsf{bxid}, \ell, m; \mathsf{EDX})$
(c) S sets EBB[bxid] := EDX;
(d) $\mathbf{C}_a$ sets bxst := (dxst, bxid);
(e) output (bxst; EBB);
(5) else, output $(\perp; \perp)$;

$\mathsf{GetMail}_{\mathbf{C}_a,\mathbf{S}}(\mathsf{bxst}, K_{\mathsf{BXB}}, \ell; \mathsf{EBB})$:
(1) $\mathbf{C}_a$ parses bxst as (dxst, bxid);
(2) $\mathbf{C}_a$ parses K_{BXB} as K_{GDX};
(3) $\mathbf{C}_a$ sends bxid to S via $\mathcal{F}_{\mathsf{anon}}$;
(4) if $\mathsf{EBB}[\mathsf{bxid}] \neq \perp$,
(a) S computes EDX := EBB[bxid];
(b) $\mathbf{C}_a$ and S run
$(\mathsf{dxst}, m; \mathsf{EDX}) \leftarrow \Sigma_{\mathsf{GDX}}.\mathsf{Get}_{\mathbf{C}_a,\mathbf{S}}(\mathsf{dxst}_{ab}, K_{\mathsf{GDX}}, \mathsf{bxid}, \ell; \mathsf{EDX})$
(c) S sets EBB[bxid] := EDX;
(d) $\mathbf{C}_a$ sets bxst := (dxst, bxid);
(e) output (bxst, m; EBB);
(5) else, output $(\perp; \perp)$;

$\mathsf{DeleteMail}_{\mathbf{C}_a,\mathbf{S}}(\mathsf{bxst}, K_{\mathsf{BXB}}, \ell; \mathsf{EBB})$:
(1) $\mathbf{C}_a$ parses bxst as (dxst, bxid);
(2) $\mathbf{C}_a$ parses K_{BXB} as K_{GDX};
(3) $\mathbf{C}_a$ sends bxid to S via $\mathcal{F}_{\mathsf{anon}}$;
(4) if $\mathsf{EBB}[\mathsf{bxid}] \neq \perp$,
(a) S computes EDX := EBB[bxid];
(b) $\mathbf{C}_a$ and S run
$(\mathsf{dxst}; \mathsf{EDX}) \leftarrow \Sigma_{\mathsf{GDX}}.\mathsf{Delete}_{\mathbf{C}_a,\mathbf{S}}(\mathsf{dxst}, K_{\mathsf{GDX}}, \mathsf{bxid}, \ell; \mathsf{EDX})$
(c) S sets EBB[bxid] := EDX;
(d) C sets bxst := (dxst, bxid);
(e) output (bxst; EBB);
(5) else, output $(\perp; \perp)$;

Figure 11

(5) when $\mathcal{Z}$ outputs $(\mathbf{C}_a, \mathsf{delmail}, \mathsf{bxid}, \ell)$, $\mathcal{Z}_{\mathsf{GDX}}$ sends bxid to S via $\mathcal{F}_{\mathsf{anon}}$ and outputs $(\mathbf{C}_a, \mathsf{delete}, \mathsf{dxid}, \ell)$, where dxid := bxid while forwarding all messages back and forth between χ and $\mathcal{A}$ and $\mathcal{A}$ and $\mathcal{Z}$. It then sends $(\perp; \perp)$ to $\mathcal{Z}$.

Note that if $\mathcal{Z}_{\mathsf{GDX}}$ is in a $\mathcal{Z}_{\mathsf{GDX}}\left(\mathcal{A}_{\mathsf{GDX}}, \mathbf{T}, \mathbf{C}_1, \ldots, \mathbf{C}_n :: \Sigma_{\mathsf{GDX}}\right)$ experiment then $\mathcal{Z}$ and $\mathcal{A}$'s views are as in Game_{i-1}. On the other hand, if $\mathcal{Z}_{\mathsf{GDX}}$ is in a $\mathcal{Z}_{\mathsf{GDX}}\left(\mathcal{S}_{\mathsf{GDX}}, \mathbf{C}_1, \ldots, \mathbf{C}_n :: \mathcal{F}_{\mathsf{GDX}}^{\mathcal{L}_{\mathsf{GDX}}}\right)$ experiment, then $\mathcal{Z}$ and $\mathcal{A}$'s views are as in Game_i. It follows by our initial assumption that $\mathcal{Z}_{\mathsf{GDX}}$ and $\mathcal{A}_{\mathsf{GDX}}$ violate the $\mathcal{L}_{\mathsf{GDX}}^{\delta_{\mathsf{GDX}}}$-security of Σ_{GDX}.

Towards finishing the proof, note that Game_n is equivalent to a $\mathcal{Z}\left(\mathcal{S}, \mathbf{C}_1, \ldots, \mathbf{C}_n :: \mathcal{F}_{\mathsf{BXB}}^{\mathcal{L}_{\mathsf{BXB}}}\right)$ experiment. And since n is a constant and is independent of k, Equation 1 follows by a standard probabilistic argument.

■

Leakage $\mathcal{L}_{\mathsf{BXB}}^{\delta}$

$\mathcal{L}_{\mathsf{BXB}}^{\delta}(\mathsf{op})$:
(1) if op = (init, λ),
(a) set $t := 0$ and bxrank := 0;
(b) initialize a dictionary sDX;
(c) set $\check{\mathsf{vx}} := [\![\, \Sigma_{\mathsf{BXB}}, \mathsf{init}, t\,;\, \perp \,]\!]$;
(d) set $\check{\mathbf{E}} := \emptyset$;
(2) if op = $(\mathsf{crtbx}, \lambda, \mathbf{i}_a, \mathbf{i}_b, \varphi_\beta, \varphi_\alpha, \mathsf{bxid}, \mathbb{L}, \mathbb{V})$,
(a) set $t := t + 1$ and bxrank := bxrank + 1;
(b) set sDX[bxid] := bxrank;
(c) compute
$(\mathsf{vx}, \mathbf{E}) := \mathcal{L}_{\mathsf{GDX}}(\mathsf{init}, \lambda, \mathbf{i}_a, \mathbf{i}_b, \varphi_\alpha, \varphi_\beta, \mathsf{bxid}, \mathbb{L}, \mathbb{V})$;
(d) set
$\check{\mathsf{vx}} := [\![\, \Sigma_{\mathsf{BXB}}, \mathsf{crtbx}, t, \mathsf{bxrank}, \varphi_a, \varphi_b, \mathsf{bxid}\,;\, \mathsf{vx} \,]\!]$;
(e) set $\check{\mathbf{E}} := \mathbf{E}$;
(3) if op = $(\mathsf{drpmail}, \lambda, \mathbf{i}_a, \mathsf{bxid}, \ell, m)$,
(a) set $t := t + 1$;
(b) compute
$(\mathsf{vx}, \mathbf{E}) := \mathcal{L}_{\mathsf{GDX}}(\mathsf{put}, \lambda, \mathbf{i}_a, \mathsf{bxid}, \ell, m)$;
(c) compute bxrank := sDX[bxid];
(d) set $\check{\mathsf{vx}} := [\![\, \Sigma_{\mathsf{BXB}}, \mathsf{drpmail}, t, \mathsf{bxrank}\,;\, \mathsf{vx} \,]\!]$;
(e) set $\check{\mathbf{E}} := \mathbf{E}$;
(4) if op = $(\mathsf{gtmail}, \lambda, \mathbf{i}_a, \mathsf{bxid}, \ell)$,
(a) set $t := t + 1$;
(b) compute $(\mathsf{vx}, \mathbf{E}) := \mathcal{L}_{\mathsf{GDX}}(\mathsf{get}, \lambda, \mathbf{i}_a, \mathsf{bxid}, \ell)$;
(c) compute bxrank := sDX[bxid];
(d) set $\check{\mathsf{vx}} := [\![\, \Sigma_{\mathsf{BXB}}, \mathsf{gtmail}, t, \mathsf{bxrank}\,;\, \mathsf{vx} \,]\!]$;
(e) set $\check{\mathbf{E}} := \mathbf{E}$;
(5) if op = $(\mathsf{delmail}, \lambda, \mathbf{i}_a, \mathsf{bxid}, \ell)$,
(a) set $t := t + 1$;
(b) compute
$(\mathsf{vx}, \mathbf{E}) := \mathcal{L}_{\mathsf{GDX}}(\mathsf{delete}, \lambda, \mathbf{i}_a, \mathsf{bxid}, \ell)$;
(c) compute bxrank := sDX[bxid];
(d) set $\check{\mathsf{vx}} := [\![\, \Sigma_{\mathsf{BXB}}, \mathsf{delmail}, t, \mathsf{bxrank}\,;\, \mathsf{vx} \,]\!]$;
(e) set $\check{\mathbf{E}} := \mathbf{E}$;
(6) output $(\check{\mathsf{vx}}, \check{\mathbf{E}})$;

Figure 12

I The tigro Functionality and Leakage

The pseudocode for the annotation functionality is in Figure 13 and for the tigro leakage is in Figure 14. The security of the tigro protocol is captured in the following Theorem.

Theorem 3. *If Σ_{BXB} is $\mathcal{L}_{\mathsf{BXB}}^{\delta}$-secure, then* tigro *as described in Figure 2 is $\mathcal{L}_{\mathsf{tigro}}^{\delta}$-secure in the $\mathcal{F}_{\mathsf{anon}}$-hybrid model.*

Proof. Let $\mathcal{S}_{\mathsf{BXB}}$ be the simulator guaranteed to exist by the $\mathcal{L}_{\mathsf{BXB}}$-security of Σ_{BXB} and consider the simulator that simulates $\mathcal{A}$ and works as follows:

- simulating Init: given $(\check{\mathsf{vx}}, \check{\mathbf{E}})$ parse $\check{\mathsf{vx}}$ as
$$[\![\, \mathsf{tigro}, \mathsf{init}, t\,;\, \mathsf{vx} \,]\!],$$
and $\check{\mathbf{E}}$ as $\mathbf{E}$.
- simulating Handshake: given $(\check{\mathsf{vx}}, \check{\mathbf{E}})$, parse $\check{\mathsf{vx}}$ as
$$[\![\, \mathsf{tigro}, \mathsf{hshk}, t, \varphi_a, \varphi_b\,;\, \mathsf{vx} \,]\!],$$

Functionality $\mathcal{F}^{\mathcal{L}}_{\mathsf{anno}}$

The functionality is parameterized with a leakage profile $\mathcal{L}$, a distance $\delta \in \mathbb{R}_{\geq 0}$ and identification functions (φ, γ). It interacts with n clients $\mathbf{C}_1, \ldots, \mathbf{C}_n$, a trusted party $\mathbf{T}$, a server $\mathbf{S}$ and an ideal adversary $\mathcal{S}$. It stores and manages a two-dimensional multi-map 2MM and a location function λ using the following operations:

$\mathsf{SetLoc}_{\mathbf{T}}(\lambda)$:
(1) set $\lambda := \lambda$;

$\mathsf{Init}_{\mathbf{S}}(\perp)$:
(1) initialize a two-dimensional dictionary 2DX;
(2) set $T := \emptyset$;
(3) set grounded := $\emptyset$;
(4) output $(\perp)$;

$\mathsf{Handshake}_{\mathbf{C}_a,\mathbf{C}_b,\mathbf{S}}(\tau_a, \varphi_\beta; \tau_b, \varphi_\alpha; \perp)$:
(1) set $\mathsf{leak}_{\mathcal{S}} := \mathcal{L}(\mathsf{hshk}, \lambda, \mathsf{i}_a, \tau_a, \mathsf{i}_b, \tau_b)$;
(2) if $\mathcal{D}(\lambda_a, \lambda_b) \leq \delta$ and $\varphi_b = \varphi_\beta$ and $\varphi_a = \varphi_\alpha$,
  (a) set $T := T \cup \{\tau_a, \tau_b\}$;
  (b) set grounded := grounded $\cup \big\{\{\tau_a, \tau_b\}\big\}$;
  (c) output $(\tau_b; \tau_a; \perp; \mathsf{leak}_{\mathcal{S}})$;
(3) else, set $(\perp_A; \perp_A; \perp_A; \mathsf{leak}_{\mathcal{S}})$;

$\mathsf{Annotate}_{\mathbf{C}_a,\mathbf{S}}(\tau_a, \mathsf{oid}, \mathsf{anno}, \mathsf{authlist}; \perp)$:
(1) set $\mathsf{leak}_{\mathcal{S}} := \mathcal{L}(\mathsf{ann}, \lambda, \mathsf{i}_a, \mathsf{oid}, \mathsf{anno}, \mathsf{authlist})$;
(2) for all $\tau_b \in$ authlist,
  (a) if $\{\tau_a, \tau_b\} \in$ grounded, set $\mathsf{2DX}[\mathsf{sort}(\tau_a, \tau_b), \mathsf{oid}] :=$ anno;
(3) output $(\perp; \perp; \mathsf{leak}_{\mathcal{S}})$;

$\mathsf{GetAnnotation}_{\mathbf{C}_a,\mathbf{S}}(\tau_a, \mathsf{oid}; \perp)$:
(1) set $\mathsf{leak}_{\mathcal{S}} := \mathcal{L}(\mathsf{gtanno}, \lambda, \mathsf{i}_a, \mathsf{oid})$;
(2) for all $\tau_b \in T \setminus \tau_a$,
  (a) if $\{\tau_a, \tau_b\} \in$ grounded,
    (i) set anno := $\mathsf{2DX}\big[\mathsf{sort}(\tau_a, \tau_b), \mathsf{oid}\big]$;
    (ii) set **anno** := (**anno**, anno);
(3) output (**anno**; $\perp$; $\mathsf{leak}_{\mathcal{S}}$);

$\mathsf{DeleteAnnotation}_{\mathbf{C}_a,\mathbf{S}}(\tau_a, \mathsf{oid}, \mathsf{authlist}; \perp)$:
(1) set $\mathsf{leak}_{\mathcal{S}} := \mathcal{L}(\mathsf{delanno}, \lambda, \mathsf{i}_a, \mathsf{oid}, \mathsf{authlist})$;
(2) for all $\tau_b \in$ authlist,
  (a) if $\{\tau_a, \tau_b\} \in$ grounded, set $\mathsf{2DX} - \big(\mathsf{sort}(\tau_a, \tau_b), \mathsf{oid}\big)$;
(3) output $(\perp; \perp; \mathsf{leak}_{\mathcal{S}})$;

Figure 13

Leakage $\mathcal{L}^{\delta}_{\mathsf{tigro}}$

$\mathcal{L}^{\delta}_{\mathsf{tigro}}(\mathsf{op})$:
(1) if op = (init, λ),
  (a) set $t := 0$;
  (b) initialize a multi-map sMM;
  (c) initialize a dictionary sDX;
  (d) set $\check{\mathsf{vx}} := [\![\, \mathsf{tigro}, \mathsf{init}, t\,;\, \perp \,]\!]$;
  (e) set $\check{\mathbf{E}} := \emptyset$;
(2) if op = $(\mathsf{hshk}, \lambda, \mathsf{i}_a, \tau_a, \varphi_\beta, \mathsf{i}_b, \tau_b, \varphi_\alpha)$,
  (a) set $t := t + 1$;
  (b) sample tid $\leftarrow_{\$} \{0,1\}^k$;
  (c) set $\mathsf{sDX}\big[\mathsf{sort}(\tau_a, \tau_b)\big] := \mathsf{tid}$;
  (d) set $\mathsf{sMM}[\tau_a] \stackrel{+}{=} \tau_b$ and $\mathsf{sMM}[\tau_b] \stackrel{+}{=} \tau_a$;
  (e) compute
  $$(\mathsf{vx}, \mathbf{E}) := \mathcal{L}_{\mathsf{BXB}}(\mathsf{crtbx}, \lambda, \mathsf{i}_a, \mathsf{i}_b, \varphi_\alpha, \varphi_\beta, \mathsf{tid}, \mathbb{L}, \mathbb{V});$$
  (f) set $\check{\mathsf{vx}} := [\![\, \mathsf{tigro}, \mathsf{hshk}, t, \varphi_a, \varphi_b\,;\, \mathsf{vx} \,]\!]$;
  (g) set $\check{\mathbf{E}} := \mathbf{E}$;
(3) if op = $(\mathsf{ann}, \lambda, \tau_a, \mathsf{oid}, \mathsf{anno}, \mathsf{authlist})$,
  (a) set $t := t + 1$;
  (b) parse authlist as $(\tau_1, \ldots, \tau_m)$;
  (c) for all $i \in [m]$,
    (i) set $\mathsf{tid}_i := \mathsf{sDX}\big[\mathsf{sort}(\tau_a, \tau_i)\big]$;
    (ii) compute
    $$(\mathsf{vx}_i, \mathbf{E}_i) := \mathcal{L}_{\mathsf{BXB}}(\mathsf{drpmail}, \lambda, \mathsf{i}_a, \mathsf{tid}_i, \mathsf{oid}, \mathsf{anno});$$
  (d) set $\check{\mathsf{vx}} := [\![\, \mathsf{tigro}, \mathsf{ann}, t\,;\, \mathsf{vx}_1, \ldots, \mathsf{vx}_m \,]\!]$;
  (e) set $\check{\mathbf{E}} := (\mathbf{E}_1, \ldots, \mathbf{E}_m)$;
(4) if op = $(\mathsf{gtanno}, \lambda, \tau_a, \mathsf{oid})$,
  (a) set $t := t + 1$;
  (b) compute $(\tau_1, \ldots, \tau_m) := \mathsf{sMM}[\tau_a]$;
  (c) for all $i \in [m]$,
    (i) set $\mathsf{tid}_i := \mathsf{sDX}\big[\mathsf{sort}(\tau_a, \tau_i)\big]$;
    (ii) compute $(\mathsf{vx}_i, \mathbf{E}_i) := \mathcal{L}_{\mathsf{BXB}}(\mathsf{gtmail}, \lambda, \mathsf{i}_a, \mathsf{tid}_i, \mathsf{oid})$;
  (d) set $\check{\mathsf{vx}} := [\![\, \mathsf{tigro}, \mathsf{gtanno}, t\,;\, \mathsf{vx}_1, \ldots, \mathsf{vx}_m \,]\!]$;
  (e) set $\check{\mathbf{E}} := (\mathbf{E}_1, \ldots, \mathbf{E}_m)$;
(5) if op = $(\mathsf{delanno}, \lambda, \tau_a, \mathsf{oid}, \mathsf{authlist})$,
  (a) set $t := t + 1$;
  (b) parse authlist as $(\tau_1, \ldots, \tau_m)$;
  (c) for all $i \in [m]$,
    (i) set $\mathsf{tid}_i := \mathsf{sDX}\big[\mathsf{sort}(\tau_a, \tau_i)\big]$;
    (ii) compute $(\mathsf{vx}_i, \mathbf{E}_i) := \mathcal{L}_{\mathsf{BXB}}(\mathsf{delmail}, \lambda, \mathsf{i}_a, \mathsf{tid}_i, \mathsf{oid})$;
  (d) set $\check{\mathsf{vx}} := [\![\, \mathsf{tigro}, \mathsf{delanno}, t\,;\, \mathsf{vx}_1, \ldots, \mathsf{vx}_m \,]\!]$;
  (e) set $\check{\mathbf{E}} := (\mathbf{E}_1, \ldots, \mathbf{E}_m)$;
(6) output $(\check{\mathsf{vx}}, \check{\mathbf{E}})$;

Figure 14

and $\check{\mathbf{E}}$ as $\mathbf{E}$. Use $\mathcal{S}_{\mathsf{BXB}}(\mathsf{crtbx}, \mathsf{vx}, \mathbf{E})$ to simulate a Σ_{BXB}.CreateBox execution with $\mathcal{A}$ and output whatever it outputs.

- simulating Annotate: given $(\check{\mathsf{vx}}, \check{\mathbf{E}})$, parse $\check{\mathsf{vx}}$ as
$$[\![\, \mathsf{tigro}, \mathsf{ann}, t\,;\, \mathsf{vx}_1, \ldots, \mathsf{vx}_m \,]\!],$$
and $\check{\mathbf{E}}$ as $(\mathbf{E}_1, \ldots, \mathbf{E}_m)$. For all $i \in [m]$, use $\mathcal{S}_{\mathsf{BXB}}(\mathsf{drpmail}, \mathsf{vx}_i, \mathbf{E}_i)$ to simulate a Σ_{BXB}.DropMail execution with $\mathcal{A}$ and output whatever it outputs.
- simulating GetAnnotation: given $(\check{\mathsf{vx}}, \check{\mathbf{E}})$, parse $\check{\mathsf{vx}}$ as
$$[\![\, \mathsf{tigro}, \mathsf{gtanno}, t\,;\, \mathsf{vx}_1, \ldots, \mathsf{vx}_m \,]\!],$$
and $\check{\mathbf{E}}$ as $(\mathbf{E}_1, \ldots, \mathbf{E}_m)$. For all $i \in [m]$, use $\mathcal{S}_{\mathsf{BXB}}(\mathsf{gtmail}, \mathsf{vx}_i, \mathbf{E}_i)$ to simulate a Σ_{BXB}.GetMail execution with $\mathcal{A}$ and output whatever it outputs.
- simulating DeleteMail: given $(\check{\mathsf{vx}}, \check{\mathbf{E}})$, parse $\check{\mathsf{vx}}$ as
$$[\![\, \mathsf{tigro}, \mathsf{delanno}, t\,;\, \mathsf{vx}_1, \ldots, \mathsf{vx}_m \,]\!],$$
and $\check{\mathbf{E}}$ as $(\mathbf{E}_1, \ldots, \mathbf{E}_m)$. For all $i \in [m]$, use $\mathcal{S}_{\mathsf{BXB}}(\mathsf{delmail}, \mathsf{vx}_i, \mathbf{E}_i)$ to simulate a Σ_{BXB}.DeleteMail execution with $\mathcal{A}$ and output whatever it outputs.

It remains to show that for all p.p.t. adversaries $\mathcal{A}$, all p.p.t. environments $\mathcal{Z}$, all $z \in \{0,1\}^*$ and all $\delta \in \mathbb{R}_{\geq 0}$,

$$\left|\Pr\left[\,\mathcal{Z}(\mathcal{A}, \mathbf{T}, \mathbf{C}_1, \ldots, \mathbf{C}_n :: \text{tigro}) = 1\,\right] - \Pr\left[\,\mathcal{Z}(\mathcal{S}, \mathbf{T}, \mathbf{C}_1, \ldots, \mathbf{C}_n :: \mathcal{F}_{\text{anno}}^{\mathcal{L}_{\text{tigro}}}) = 1\,\right]\right| \leq \text{negl}(k). \tag{2}$$

We argue that if there exists a p.p.t. adversary $\mathcal{A}$, a p.p.t. environment $\mathcal{Z}$, a string $z \in \{0,1\}^*$ and a distance $\delta \in \mathbb{R}_{\geq 0}$ such that the probability that $\mathcal{Z}(\mathcal{A}, \mathbf{T}, \mathbf{C}_1, \ldots, \mathbf{C}_n ::$ tigro) and $\mathcal{Z}(\mathcal{S}, \mathbf{T}, \mathbf{C}_1, \ldots, \mathbf{C}_n :: \mathcal{F}_{\text{anno}}^{\mathcal{L}_{\text{tigro}}})$ output 1 is non-negligibly close, then there exists a p.p.t. adversary $\mathcal{A}_{\text{BXB}}$ such that then for all p.p.t. simulators $\mathcal{S}_{\text{BXB}}$, there exists a p.p.t. environment $\mathcal{Z}_{\text{BXB}}$, a string $z_{\text{BXB}} \in \{0,1\}^*$ and a distance $\delta_{\text{BXB}} \in \mathbb{R}_{\geq 0}$ such that the $\mathcal{L}_{\text{BXB}}$-security of Σ_{BXB} is violated.

Let $\mathcal{A}_{\text{BXB}}$ be the adversary that forwards messages back and forth and $z_{\text{BXB}} = z$ and $\delta_{\text{BXB}} = \delta$. We denote by χ the party that is either $\mathcal{A}_{\text{BXB}}$ or $\mathcal{S}_{\text{BXB}}$. $\mathcal{Z}_{\text{BXB}}$ simulates $\mathcal{Z}$ and $\mathcal{A}$ and does the following:

(1) when $\mathcal{Z}$ outputs (init, λ), $\mathcal{Z}_{\text{BXB}}$ outputs (init, λ). It also initializes a two-dimensional multi-map sMM.
(2) when $\mathcal{Z}$ outputs $(\varphi, \gamma, \lambda, \mathbf{C}_a, \varphi_\beta, \mathbf{C}_b, \varphi_\alpha, \text{hshk}, \mathbb{L}, \mathbb{V})$, $\mathcal{Z}_{\text{BXB}}$ outputs $(\varphi, \gamma, \lambda, \mathbf{C}_a, \varphi_\beta, \mathbf{C}_b, \varphi_\alpha, \text{crtbx}, \mathbb{L}, \mathbb{V})$. Upon receiving bxid, it sets $\text{sMM}\big[\text{sort}(\tau_a, \tau_b)\big] :=$ bxid and sends $(\tau_b; \tau_a)$ to $\mathcal{Z}$. $\mathcal{Z}_{\text{BXB}}$ also forwards messages back and forth between χ and $\mathcal{A}$ and $\mathcal{A}$ and $\mathcal{Z}$.
(3) when $\mathcal{Z}$ outputs $(\mathbf{C}_a, \text{ann}, \text{oid}, \text{anno}, \text{authlist})$ then, for all $\tau_b \in$ authlist, $\mathcal{Z}_{\text{BXB}}$ computes $\text{bxid} := \text{sMM}\big[\text{sort}(\tau_a, \tau_b)\big]$ and outputs $(\mathbf{C}_a, \text{drpmail}, \text{bxid}, \text{oid}, \text{anno})$ and sends $(\perp; \perp)$ to $\mathcal{Z}$. $\mathcal{Z}_{\text{BXB}}$ also forwards messages back and forth between χ and $\mathcal{A}$ and $\mathcal{A}$ and $\mathcal{Z}$.
(4) when $\mathcal{Z}$ outputs $(\mathbf{C}_a, \text{gtanno}, \text{oid})$, then, for all $\text{bxid} \in \text{sMM}\big[\text{sort}(\tau_a, \tau_b)\big]$, $\mathcal{Z}_{\text{BXB}}$ outputs $(\mathbf{C}_a, \text{gtmail}, \text{bxid}, \text{oid})$. Upon receiving m, it returns $(m; \perp)$ to $\mathcal{Z}$. $\mathcal{Z}_{\text{BXB}}$ also forwards messages back and forth between χ and $\mathcal{A}$ and $\mathcal{A}$ and $\mathcal{Z}$.
(5) when $\mathcal{Z}$ outputs $(\mathbf{C}_a, \text{delanno}, \text{oid}, \text{authlist})$ then, for all $\tau_b \in$ authlist, $\mathcal{Z}_{\text{BXB}}$ computes $\text{bxid} := \text{sMM}\big[\text{sort}(\tau_a, \tau_b)\big]$ and outputs $(\mathbf{C}_a, \text{delmail}, \text{bxid}, \text{oid})$ and sends $(\perp; \perp)$ to $\mathcal{Z}$. $\mathcal{Z}_{\text{BXB}}$ also forwards messages back and forth between χ and $\mathcal{A}$ and $\mathcal{A}$ and $\mathcal{Z}$.

Note that if $\mathcal{Z}_{\text{BXB}}$ is in a $\mathcal{Z}_{\text{BXB}}(\mathcal{A}_{\text{BXB}}, \mathbf{T}, \mathbf{C}_1, \ldots, \mathbf{C}_n :: \Sigma_{\text{BXB}},)$ experiment then $\mathcal{Z}$ and $\mathcal{A}$'s views are as in $\mathcal{Z}(\mathcal{A}, \mathbf{T}, \mathbf{C}_1, \ldots, \mathbf{C}_n ::$ tigro) experiment. On the other hand, if $\mathcal{Z}_{\text{BXB}}$ is in a

$$\mathcal{Z}_{\text{BXB}}\left(\mathcal{S}_{\text{BXB}}, \mathbf{T}, \mathbf{C}_1, \ldots, \mathbf{C}_n :: \mathcal{F}_{\text{BXB}}^{\mathcal{L}_{\text{BXB}}}\right)$$

experiment, then $\mathcal{Z}$ and $\mathcal{A}$'s views are as in a

$$\mathcal{Z}(\mathcal{S}, \mathbf{T}, \mathbf{C}_1, \ldots, \mathbf{C}_n :: \mathcal{F}_{\text{anno}}^{\mathcal{L}_{\text{tigro}}})$$

experiment. It follows by our initial assumption that $\mathcal{Z}_{\text{BXB}}$ and $\mathcal{A}_{\text{BXB}}$ violate the $\mathcal{L}_{\text{BXB}}$-security of Σ_{DX}, from which the Theorem follows.

■

J The Leakage of tigro$[\pi^*_{\text{bas}}]$

The leakage of tigro$[\pi^*_{\text{bas}}]$ is in Figures 15 and 16.

Leakage $\mathcal{L}_{\text{tigro}}[\pi^*_{\text{bas}}]^\delta$

$\mathcal{L}_{\text{tigro}}[\pi^*_{\text{bas}}]^\delta(\text{op})$:
(1) if op = (init, λ),
  (a) set $t := 0$;
  (b) initialize a multi-map sMM;
  (c) initialize a dictionary sDX;
  (d) set $\check{\text{vx}} := [\![\,\text{tigro}[\pi^*_{\text{bas}}], \text{init}, t\,;\,\perp\,]\!]$;
  (e) set $\check{\mathbf{E}} := \emptyset$;
(2) if op = $(\text{hshk}, \lambda, \mathbf{i}_a, \tau_a, \varphi_\beta, \mathbf{i}_b, \tau_b, \varphi_\alpha)$,
  (a) set $t := t + 1$;
  (b) sample $\text{tid} \leftarrow\$ \{0,1\}^k$;
  (c) set $\text{sDX}\big[\text{sort}(\tau_a, \tau_b)\big] := \text{tid}$;
  (d) set $\text{sMM}[\tau_a] \stackrel{+}{=} \tau_b$ and $\text{sMM}[\tau_b] \stackrel{+}{=} \tau_a$;
  (e) set $\check{\text{vx}}$ as
$[\![\text{tigro}[\pi^*_{\text{bas}}], \text{hshk}, t, \varphi_a, \varphi_b;$
$[\![\Sigma_{\text{BXB}}, \text{crtbx}, t, \text{bxrank}, \varphi_a, \varphi_b, \text{tid};$
$[\![\Sigma_{\text{GDX}}, \text{init}, t, \lambda_a, \varphi_a, \varphi_\alpha, \lambda_b, \varphi_b, \varphi_\beta, \text{tid};$
$\langle\!\langle\, \pi^*_{\text{bas}}, \text{init}, t, \text{tid}, \mathbb{L}, \mathbb{V} \mid \perp \,\rangle\!\rangle]\!]]\!]]\!]$
  (f) set $\check{\mathbf{E}} := \emptyset$;
(3) if op = $(\text{ann}, \lambda, \tau_a, \text{oid}, \text{anno}, \text{authlist})$,
  (a) set $t := t + 1$;
  (b) parse authlist as $(\tau_1, \ldots, \tau_m)$;
  (c) for all $i \in [m]$, set $\text{tid}_i := \text{sDX}\big[\text{sort}(\tau_a, \tau_i)\big]$;
  (d) set $\check{\text{vx}}$ as
$[\![\text{tigro}[\pi^*_{\text{bas}}], \text{ann}, t;$
$[\![\Sigma_{\text{BXB}}, \text{drpmail}, t, \text{bxrank};$
$[\![\Sigma_{\text{GDX}}, \text{put}, t, \text{tid}_i;$
$\langle\!\langle\, \pi^*_{\text{bas}}, \text{put}, t, \text{tid}_i, |\text{anno}| \mid \text{oid} \,\rangle\!\rangle]\!]_{i \in [m]}]\!]$
  (e) set $\check{\mathbf{E}}$ as
$\check{\text{vx}} \times \mathbf{V}\left(\pi^*_{\text{bas}} \wedge (\text{get} \vee \text{put} \vee \text{delete}) \wedge \bigvee_{i=1}^{m} \text{tid}_i \mid \text{oid}\right);$

Figure 15

K EXH-based tigro

In this section, we instantiate the tigro protocol using the EXH multi-map encryption scheme [24]. EXH leverages latency as a new paradigm to minimize the leakage. In particular, EXH has two modes of operations: a *low-leakage* mode, which leaks minimal information about the queries and the data, such as the expected response length, the operation arrival rate, and the size of the multi-map; and a *subliminal* mode that only leaks the size of the multi-map. The subliminal mode is more potent as it hides whether an operation was executed at the server. EXH's main technical idea is conceptually simple: it introduces a queue at the client that stores the client's operations, and there is a servicing mechanism that sends operations to the server following some fixed distribution. If the distribution is chosen independently of the data and the queries, then EXH operates in the subliminal mode; otherwise, it operates in the low-leakage mode.

In the remainder of this section, we briefly recall EXH, discuss the main technical challenges we face when instantiating tigro using EXH, describe its efficiency as well as the efficiency of the grounded

Leakage $\mathcal{L}_{\text{tigro}}[\pi^*_{\text{bas}}]^{\delta}$ (part 2)

4. if op = (gtanno, λ, τ_a, oid),
   (a) set $t := t + 1$;
   (b) compute $(\tau_1, \ldots, \tau_m) := \text{sMM}[\tau_a]$;
   (c) for all $i \in [m]$, set $\text{tid}_i := \text{sDX}\big[\text{sort}(\tau_a, \tau_i)\big]$;
   (d) set $\check{\text{vx}}$ as

$$[\![\text{tigro}[\pi^*_{\text{bas}}], \text{gtanno}, t;\ [\![\Sigma_{\text{BXB}}, \text{gtmail}, t, \text{bxrank};\ [\![\Sigma_{\text{GDX}}, \text{get}, t, \text{tid}_i;\ \langle\!\langle\ \pi^*_{\text{bas}}, \text{get}, t, \text{tid}_i \mid \text{oid}\ \rangle\!\rangle]\!]_{i\in[m]}]\!]$$

   (e) set $\check{\mathbf{E}}$ as

$$\check{\text{vx}} \times \mathbf{V}\left(\pi^*_{\text{bas}} \wedge (\text{get} \vee \text{put} \vee \text{delete}) \wedge \bigvee_{i=1}^{m} \text{tid}_i \mid \text{oid}\right);$$

5. if op = (delanno, λ, τ_a, oid, authlist),
   (a) set $t := t + 1$;
   (b) parse authlist as $(\tau_1, \ldots, \tau_m)$;
   (c) for all $i \in [m]$, set $\text{tid}_i := \text{sDX}\big[\text{sort}(\tau_a, \tau_i)\big]$;
   (d) set $\check{\text{vx}}$ as

$$[\![\text{tigro}[\pi^*_{\text{bas}}], \text{delanno}, t;\ [\![\Sigma_{\text{BXB}}, \text{delmail}, t, \text{bxrank};\ [\![\Sigma_{\text{GDX}}, \text{delete}, t, \text{tid}_i;\ \langle\!\langle\ \pi^*_{\text{bas}}, \text{delete}, t, \text{tid}_i \mid \text{oid}\ \rangle\!\rangle]\!]_{i\in[m]}]\!]$$

   (e) set $\check{\mathbf{E}}$ as

$$\check{\text{vx}} \times \mathbf{V}\left(\pi^*_{\text{bas}} \wedge (\text{get} \vee \text{put} \vee \text{delete}) \wedge \bigvee_{i=1}^{m} \text{tid}_i \mid \text{oid}\right);$$

6. output $(\check{\text{vx}}, \check{\mathbf{E}})$;

Figure 16

dictionary encryption scheme, the encrypted box bank scheme, and tigro. Finally, we describe the concrete leakage profile of tigro and all its building blocks. Note that some results below require background on queueing theory, and we recommend the reader to go over the preliminary section in [24] or for more details [158]. Moreover, we only focus on the query part in our analysis, but a similar analysis can be done for the updates and the deletes. We also consider the special case where the multi-map is a dictionary since this is sufficient for our needs.

EXH *details.* EXH is a multi-map encryption scheme with two client-side queues: a query queue $\mathbf{Q}_q$ and an update queue $\mathbf{Q}_u$. Whenever a client issues a query q, it adds the query to the queue. At the same time, there is an ongoing query process, QServicing, parametrized by a servicing distribution $\mathcal{D}$ that sends a query to the server at times sampled from $\mathcal{D}$. However, when the queue is empty, the client still has to send a *fake* query to the server to hide the real queries' arrival time and response length. The multi-map $\{\ell, \mathbf{v}\}_{\ell\in\mathbb{L}}$ is first transformed into a dictionary $\{\ell \| i_\ell, v_{i_\ell}\}_{\ell\in\mathbb{L}, i_\ell\in[\#\mathbf{v}]}$ following the same technical idea leveraged in the π_{bas} construction [39]. The dictionary is then encrypted using a leakage-free dictionary, LDX, which can be instantiated using an oblivious RAM [78]. Given that LDX is leakage-free, a server cannot distinguish between two real queries or between a real and fake query. One of the main technical challenges in EXH is quantifying the query (or update) latency or the queue size as a function of the arrival rate, the servicing rate, and the size of the response length distribution. We refer the reader to [24] for more details.

Technical challenges. The technical challenge when instantiating tigro using EXH is that the latency analysis does not work. In [24], EXH only considers a single user interacting with the data structure; in our case, however, two users can *concurrently* query or update the encrypted structure. In particular, if the two users concurrently send two operations, only one is executed. At the same time, the other one has to wait due to the locking mechanism. Such a lock substantially changes the latency analysis as, now, the operations can potentially remain in the queue longer than the single client setting.

K.1 Efficiency Analysis

In the following, we first study the efficiency of EXH when there is a locking mechanism on the entire encrypted structure. We then discuss the overall efficiency of the more high-level protocols.

Multi-class queue model. We focus on the query queue for our analysis, but a similar argument can be made for the update queue. Let $\mathbf{Q}^1_q$ and $\mathbf{Q}^2_q$ denote the query queue of the first and second users. Our first observation is that the existence of the lock enforces a fixed order on the operations in both $\mathbf{Q}^1_q$ and $\mathbf{Q}^2_q$ based on the time of arrival of the queries. As a result, we consider an abstract *shared* queue $\mathbf{Q}^{\text{sh}}_q$ that stores both elements in $\mathbf{Q}^1_q$ and $\mathbf{Q}^2_q$ ordered based on their arrival times. One can then observe that the expected latency as well as the expected size of the queue $\mathbf{Q}^{\text{sh}}_q$ is an upper bound on the expected latency and the expected queue size of both $\mathbf{Q}^1_q$ and $\mathbf{Q}^2_q$. Such a queue is referred to as a multi-class queue in queuing theory. The analysis of such a queue mainly depends on the arrival and servicing rates of the individual queues. Another contribution we make here is we also provide alternative modeling of the queueing system in [24], which has been made possible also since the response length is always equal to 1 in tigro as we are working with dictionaries instead of multi-maps. For our queue analysis, instead of going through the birth-and-death process using the transition rate matrix and then calculating the probability-generating function using the balance equations, we use the notion of queueing systems with vacation, which we describe below. In particular, we equate processing a fake element, which is required when the queue is empty in EXH, to the vacation period the server takes when a queue is empty. For our analysis, we assume that the arrival processes for both queues are a Poisson process with arrival rates λ_1 and λ_2 and exponential servicing rates μ_1 and μ_2.

Queues with vacation. Conceptually, one can equate the time spent processing a fake element, which is required when the queue is empty in EXH, to the vacation period the server takes when a queue is empty in a standard queue. In particular, a vacation is defined as the period between the queue is empty until an element arrives in the queue. Such a period is equivalent to the period needed to process a fake element. We denote such a system as $\text{M}/\overline{\text{M}}/1$ where consider a Poisson arrival process with rate λ, an exponential servicing rate with rate μ, and vacation time with rate

μ. The expected number of elements in the queue is equal to,

$$S = \frac{\rho}{1-\rho},$$

and the expected latency can be derived using Little's Law as

$$L = \frac{1}{\mu \cdot (1-\rho)}.$$

We refer the reader to [121] for more details. Now, we go back to our multi-class queuing model where the underlying queues are both $\mathsf{M}/\overline{\mathsf{M}}/1$ queueing systems with parameters λ_1, μ_1 and λ_2, μ_2. The resulting multi-class queuing system is a hyper-exponential $\mathsf{M}/\overline{\mathsf{H}}_2/1$ with vacation where the expected (shared) queue size is equal to

$$S_{\mathsf{sh}} = \frac{\lambda \cdot \mathsf{E}[S^2]}{2} \cdot \left(\frac{\lambda}{1-\rho} + \frac{1}{\mathsf{E}[S]}\right)$$

where

$$\mathsf{E}[S^2] = \frac{2\lambda_1}{\lambda \cdot \mu_1^2} + \frac{2\lambda_2}{\lambda \cdot \mu_2^2}, \quad \mathsf{E}[S] = \frac{\rho}{\lambda} \quad \text{and} \quad \rho = \frac{\lambda_1}{\mu_1} + \frac{\lambda_2}{\mu_2}$$

and $\lambda = \lambda_1 + \lambda_2$. The expected latency can be derived from Little's Law such that,

$$L_{\mathsf{sh}} = \frac{S_{\mathsf{sh}}}{\lambda}.$$

We refer the reader to [164] for more details. We are now ready to state our main efficiency Theorem.

Theorem K.1. *Given a $\mathsf{M}/\overline{\mathsf{M}}/1$ queueing system with arrival rate $\lambda \in \mathbb{R}_{>0}$, servicing rate $\mu \in \mathbb{R}_{>0}$, vacation rate μ,* EXH *with the locking mechanism has the following asymptotics: the expected size of the query queue is at most S_{sh}, the expected query latency is at most L_{sh}, the expected communication complexity is $O(\frac{\mu_i}{\lambda_i} \cdot \log N)$, and the round complexity is $O(\frac{\mu_i}{\lambda_i})$, for $i \in [2]$.*

The proof of the theorem follows directly from the analysis of the multi-class queueing system stated above. The communication complexity and the number of rounds are exactly equal to the ones stated in [24] with the assumption that LDX is instantiated using a constant client-side ORAM such as OptORAMa [20].

Efficiency of Σ_{GDX}, Σ_{BXB} and tigro. The Put and Get of Σ_{GDX} ,and GetMail and DropMail of Σ_{BXB} have the exact asymptotics as EXH stated in Theorem K.1 under the assumption of a Poisson arrival, exponential servicing and exponential vacation. The asymptotics of the GetAnnotation protocol of tigro, assuming that the for-loop over the contact list is executed in parallel, is summarized in the following corollary. Note that we assume that the cost of the anonymous network as a constant, and it does not show in the asymptotics.

Corollary K.2. *Given a $\mathsf{M}/\overline{\mathsf{M}}/1$ queueing system with arrival rate $\lambda \in \mathbb{R}_{>0}$, servicing rate $\mu \in \mathbb{R}_{>0}$, vacation rate μ,* GetAnnotation *in* tigro *has the following asymptotics: the expected size of the query queue is at most $\#\mathsf{Contacts}_a \cdot S_{\mathsf{sh}}$, the expected query latency is at most L_{sh}, the expected communication complexity is $O(\#\mathsf{Contacts}_a \cdot \frac{\mu_i}{\lambda_i} \cdot \log N)$, and the round complexity is $O(\frac{\mu_i}{\lambda_i})$, for $i \in [2]$,*

The proof follows directly from Theorem K.1 and the only change is that there are $\#\mathsf{Contacts}_a$ queues and query processes running in parallel. A similar result can be obtained for the Annotate protocol.

Leakage $\mathcal{L}^{\delta}_{\mathsf{tigro[EXH]}}$

$\mathcal{L}^{\delta}_{\mathsf{tigro[EXH]}}(\mathsf{op})$:
- (1) if $\mathsf{op} = (\mathsf{init}, \lambda)$,
  - (a) set $t := 0$;
  - (b) set $\check{\mathbf{vx}} := [\![\, \mathsf{tigro[EXH]}, \mathsf{init}, t\,;\, \perp \,]\!]$;
  - (c) set $\check{\mathbf{E}} := \emptyset$;
- (2) if $\mathsf{op} = (\mathsf{hshk}, \lambda, \mathbf{i}_a, \tau_a, \varphi_\beta, \mathbf{i}_b, \tau_b, \varphi_\alpha)$,
  - (a) set $t := t + 1$;
  - (b) sample $\mathsf{tid} \leftarrow\$ \{0,1\}^k$;
  - (c) set $\check{\mathbf{vx}}$ as

$$[\![\mathsf{tigro[EXH]}, \mathsf{hshk}, t, \varphi_a, \varphi_b;$$
$$[\![\Sigma_{\mathsf{BXB}}, \mathsf{crtbx}, t, \mathsf{bxrank}, \varphi_a, \varphi_b, \mathsf{tid};$$
$$[\![\Sigma_{\mathsf{GDX}}, \mathsf{init}, t, \lambda_a, \varphi_a, \varphi_\alpha, \lambda_b, \varphi_b, \varphi_\beta, \mathsf{tid};$$
$$\langle\!\langle\, \mathsf{EXH}, \mathsf{init}, t, \mathsf{tid}, \mathbb{L}, \mathbb{V} \mid \perp \,\rangle\!\rangle]\!]]\!]]\!]$$

  - (d) set $\check{\mathbf{E}} := \emptyset$;
- (3) if $\mathsf{op} = (\mathsf{ann}, \lambda, \tau_a, \mathsf{oid}, \mathsf{anno}, \mathsf{authlist})$,
  - (a) set $\check{\mathbf{vx}} := \emptyset$ and $\check{\mathbf{E}} := \emptyset$;
- (4) if $\mathsf{op} = (\mathsf{gtanno}, \lambda, \tau_a, \mathsf{oid})$,
  - (a) set $\check{\mathbf{vx}} := \emptyset$ and $\check{\mathbf{E}} := \emptyset$;
- (5) if $\mathsf{op} = (\mathsf{delanno}, \lambda, \tau_a, \mathsf{oid}, \mathsf{authlist})$,
  - (a) set $\check{\mathbf{vx}} := \emptyset$ and $\check{\mathbf{E}} := \emptyset$;
- (6) output $(\check{\mathbf{vx}}, \check{\mathbf{E}})$;

Figure 17

Remarks. The number of pairs, N, in EXH is an upper bound on the maximum number of pairs the dictionary will ever store. This upper bound is set at setup time and cannot be exceeded. Note that this is a limitation of using oblivious RAMs, as they are not resizable. A more practice-conscious decision uses the recursive variant of Path ORAM [162] instead of OptORAMa [20] as the former is significantly more practical. However, using the recursive variant of Path ORAM results in worse asymptotics. In particular, the communication complexity of EXH becomes $O(\frac{\mu_i}{\lambda_i} \cdot \log^2 N)$, and the round complexity is $O(\frac{\mu_i}{\lambda_i} \cdot \log N)$, for $i \in [2]$.

Concrete leakage. When using EXH in the subliminal mode, i.e., when the servicing rate is set independently of the arrival rate, the concrete leakage of tigro is almost null. First, notice that the leakage of the EXH-based encrypted grounded dictionary is only composed of the vertices of the init operation, and the leakage has no edges. Second, the concrete leakage of the EXH-based encrypted box bank comprises the vertices of the init and crtbx operations and has no edges. Finally, the concrete leakage of the EXH-based tigro is solely composed of the vertices of the init and hshk operations and also has no edges. In other words, EXH-based tigro only reveals the existence of a box but does not reveal who owns it, when it is queried or updated, or the number of boxes a given individual has access to. The detailed leakage profile Σ_{GDX}, Σ_{BXB} and tigro when instantiated with EXH are detailed in Figure 17.

K.2 Empirical Evaluation

As a first step towards evaluating the practicality of tigro[EXH], we implemented the concurrent multi-client version of EXH described above which outsources its state to the server and uses a coarse-grained lock to access it. As we will see in more detail below, this adds two round trips and additional communication needed to retrieve and upload the state, which significantly impacts EXH's performance—especially for larger structures. In fact, the performance of this EXH variant was poor enough that it ruled out tigro[EXH] as a practical solution for large mailboxes and we leave the design of an efficient subliminal multi-client and concurrent EMM as an important open problem.

Evaluation setup. We used the same experimental setup as in [24], which we briefly recall here for completeness. We use the Enron email dataset, which is a publicly available collection of emails from approximately 150 Enron employees. Our experiments are run on a machine with an AMD Ryzen 7 5700x processor and 32GB of RAM on a local network. First, we parse the Enron dataset and randomly generate three multi-maps of sizes 2^{12}, 2^{12}, and 2^{14}. We evaluated the scheme with two clients that execute queries concurrently. Their query arrivals follow a Poisson process, with arrival rate of 4 for the first client and 2 for the second. We define the service rate μ as the inverse of the time required by the server to process a query.

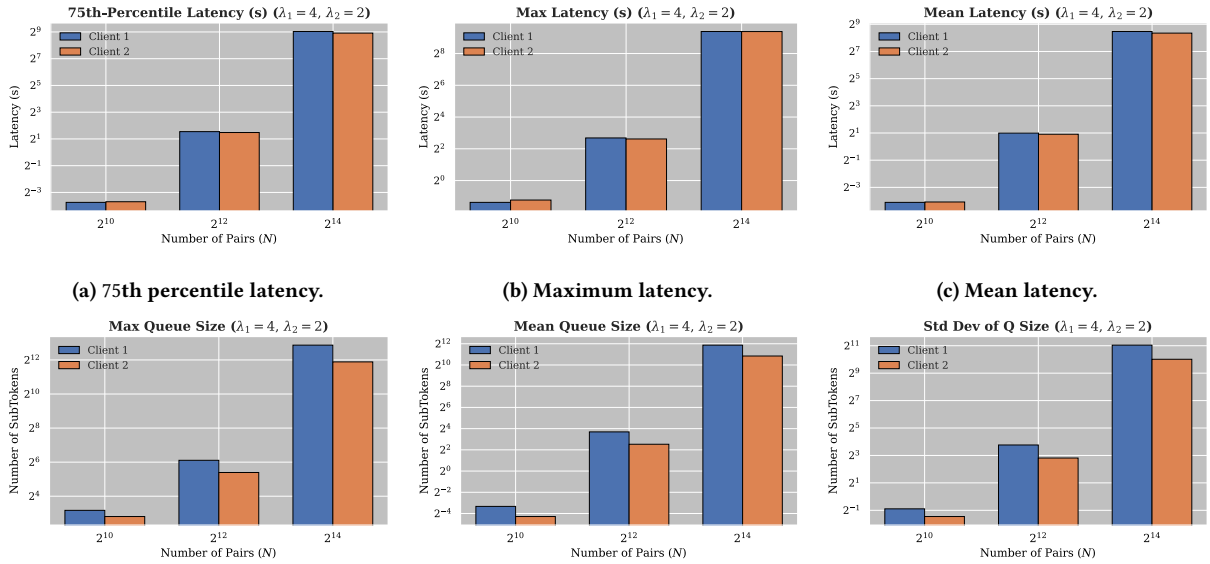
Query latency. Though we set the query rate of the first client to be twice that of the second, we observed only a small difference between the two clients. In the following, we report only the numbers for the first client, but we refer the reader to Figure 18 for more details. The 75th percentile of the query latency was 87 milliseconds (ms), while the mean and maximum values were 70 ms and 383 ms, respectively. The query latency increased significantly when we expanded the structure size to 2^{14}. Specifically, the 75th percentile, mean, and maximum query latencies were 633, 429, and 838 seconds, respectively. This is due to the large state, which impacts the servicing rate. We believe that smaller arrival rates could lead to slower growing queues but setting the service rate as a function of the arrival rate breaks the subliminality of EXH.

Queue size. We noticed almost no difference between the queues of the two clients. Figure 18 presents the maximum, mean, and standard deviation of the queue size. For $N = 2^{10}$, the mean, maximum, and standard deviation of the queue size were 0.08, 9, and 0.49, respectively—values that are small and align well with the low latency described earlier. However, when we increased N to 2^{14}, the queue size grew significantly. Specifically, the mean, maximum, and standard deviation of the queue size were 2, 120, 4, 506, and 1, 128, respectively, which again aligns with the slow latency described above.

L Ethical Principles

Our design goals are based on insights from numerous qualitative studies, historical artifacts, and personal organizing experience, and did not include any direct interactions with activists or their data. While we do intend to involve activists directly in qualitative implementation testing, grassroots optimization, and future work, such involvement will be carefully protected by IRB privacy and security protocols—we do not intend to release the tigro protocol as a mainstream software application for use outside of academic settings until resources are available for sufficient penetration testing and maintenance. When it is ready for a full release, we intend to develop educational materials and work closely with organizations to configure tigro to their needs, and continue to improve the accessibility and usability of the tigro application.

Our experiments involved running traffic through the Tor network. While our client set sizes were small, we wanted to further minimize the load our evaluation imposed on the Tor network. To do so, we avoided concurrent executions which would require clients to open many Tor circuits in parallel, and also staggered our evaluations across multiple days. This is consistent with Tor community standards and specifications, for instance the Circuits specification section, which notes that "excessive circuit creation can impact the entire path of that circuit" [148]. The C Tor implementation additionally rate-limits circuit creation per client IP address. In the future, we hope to work with folks at the Tor Project to better integrate Tor into the tigro protocol in a way that serves activists in the context of the broader Tor user base.



(a) 75th percentile latency. **(b) Maximum latency.** **(c) Mean latency.**

(d) Maximum size. **(e) Mean size.** **(f) Size standard deviation.**

Figure 18: Query latency in seconds and queue size, where the arrival rates of the first and second queues are $\lambda_1 = 4$ and $\lambda_2 = 2$, respectively. We varied the number of pairs N within the set $\{2^{10}, 2^{12}, 2^{14}\}$.