Abuse-Resistant Location Tracking: Balancing Privacy and Safety in the Offline Finding Ecosystem

Harry Eldridge* Gabrielle Beck* Matthew Green*
Nadia Heninger† Abhishek Jain*

Abstract

Location tracking accessories (or "tracking tags") such as those sold by Apple, Samsung, and Tile, allow owners to track the location of their property via offline finding networks. The tracking protocols were designed to ensure that no entity (including the vendor) can use a tag's broadcasts to surveil its owner. These privacy guarantees, however, seem to be at odds with the phenomenon of *tracker-based stalking*, where attackers use these very tags to monitor a target's movements. Numerous such criminal incidents have been reported, and in response, manufacturers have chosen to substantially weaken privacy guarantees in order to allow users to detect stalker tags. This compromise has been adopted in a recent IETF draft jointly proposed by Apple and Google.

We put forth the notion of *abuse-resistant offline finding protocols* that aim to achieve a better balance between user privacy and stalker detection. We present an efficient protocol that achieves stalker detection under realistic conditions without sacrificing honest user privacy. At the heart of our result, and of independent interest, is a new notion of *multi-dealer secret sharing* which strengthens standard secret sharing with novel privacy and correctness guarantees. We show that this primitive can be instantiated efficiently on edge devices using variants of Interleaved Reed-Solomon codes combined with new lattice-based decoding algorithms.

1 Introduction

Vendors such as Apple, Samsung, and Tile have recently begun to deploy large-scale *offline finding networks* to monitor network-disconnected devices. These systems employ short-distance communications networks such as Bluetooth Low Energy (BLE) or Ultra-Wideband (UWB) to transmit periodic advertisement messages to nearby receivers (such as smartphones). The receiving devices upload location reports to servers controlled by the service provider.

While offline finding networks can be used to locate relatively powerful devices like phones and laptops, the breakout product in this category is the *location tracking accessory* (LTA), colloquially known as a "tracking tag." Exemplified by Apple's AirTag and Tile Trackers, these tags embed a transceiver, microprocessor, and battery in a compact package that can be attached to physical objects. The popularity of tracking tags stems from their low cost (typically under $30 USD) as well as the availability of large volunteer-operated tracking networks that can detect them. As of this writing, the combined tag sales of Tile, Apple, and Samsung exceed 100 million units [41].

Privacy and stalking risks. The widespread deployment of offline finding networks exposes users to new privacy and safety risks. On the one hand, LTAs can undermine the privacy of individuals who carry them: if an LTA emits an unchanging identifier such as a static MAC address, then a *tracking adversary* such as the network operator or a third-party RF tracking network [50, 48] can easily monitor individuals' physical movements. In response to these privacy concerns, manufacturers deploy sophisticated countermeasures: for

*Johns Hopkins University, {hme,becgabri, mgreen,abhishek}@cs.jhu.edu
†University of California San Diego, nadiah@cs.ucsd.edu

Protocol	Epoch duration	Broadcasts per epoch	Stalker detection?	Tracking privacy	Continuous Proximity	Stalker detection
Apple FindMy [2] / IETF [34]:						
Near-owner mode	15 min	450	✗	n/a	n/a	
Separated mode	24 hrs	43,200	●	n/a	✗	15-60 min†
This work (§4):						
4-second epochs / 1-hour window	4 sec	1	●	$39-46$ min*	●	60 min
1-minute epochs / 1-hour window	60 sec	15	●	$41-47$ min*	●	60 min

Figure 1: We compare the detection and privacy guarantees of our scheme with various parameters to existing tag protocols. Our goal in this work is to design a scheme that maximizes tracking privacy within the stalker detection window. "Tracking privacy" indicates the duration that a tracking adversary may receive broadcasts without de-anonymizing an LTA. "Stalking detection" indicates the minimum length of time that broadcasts must be collected before a stalking LTA can be detected. Epoch duration indicates the time between changes to an LTA identifier. Broadcasts/epoch indicates the number of *repeated* broadcasts of the same identifier that an LTA will issue in this time. "Continuous Proximity" means that a tracking adversary must be in the presence of an LTA for the entire unlinkability duration in order to de-anonymize it. *The former number uses current BLE payload sizes, while the second assumes BLE v5. †Apple does not publish their methodology, so this is an estimate.

example, Apple's Find My system employs a cryptographic protocol that rotates pseudonymous identifiers and uses encrypted location reports to hide location information from third parties and from Apple itself [2].

On the other hand, the availability of inexpensive LTAs also enables *tracker-based stalking* [31], in which attackers surreptitiously place a tag on a targeted person or vehicle and then monitor the target's movements via the offline finding network. These devices have been used in hundreds of criminal incidents, including serious cases that culminated in physical assault and murder [8, 52, 25, 35, 16].

Privacy vs stalker detection. Apple, Tile, Samsung and Google have adopted stalker-detection countermeasures to alert users to the presence of an unrecognized tag that "moves with" a user for a pre-defined length of time [4, 18, 49, 34, 33]. Victims can typically trigger an audio alert from the tag, obtain the tag's serial number using Near-Field Communication (NFC), and then query the provider's servers to obtain partial account information.

In order to implement such countermeasures, manufacturers have adopted various compromises: for example, Apple AirTags rotate their identifier every 15 minutes when within range of their owner devices, but reduce this rate to once every 24 hours when out of range. This approach allows potential stalking victims to detect nearby AirTags, but at the cost of reducing privacy against tracking adversaries.[1] Apple and Google have jointly proposed an IETF draft [34] to standardize this approach.

As this solution illustrates, the goal of a stalker detection mechanism appears to be in direct conflict with the goal of preserving privacy against a tracking adversary. To defend against a tracking adversary, tags must routinely change identifiers: this ensures that a listener cannot link a series of broadcasts to a single emitting device. Yet, to detect stalkers, a potential victim must be able to determine that a series of broadcasts belongs to a single device. This raises the following question:

> *Is it possible to provide strong privacy protections against location tracking, while also enabling the detection of stalker abuse?*

Contributions. In this work, we answer the question in the affirmative, designing new protocols that offer strong privacy guarantees while ensuring that stalker tags can be reliably detected. Critically, our solutions *operate in the threat model of today's systems* and do not rely on the creation of new trusted parties or the placement of additional trust in the service provider itself.

1. **Abuse-resistant offline finding protocols:** We put forth the notion of *abuse-resistant offline finding protocols* that simultaneously achieve privacy against tracking adversaries and stalker detection by

[1] In practice, our experience shows that many users routinely carry AirTags in "separated mode". This is due at least in part to the fact that users often share physical property with other individuals, and AirTags cannot easily be paired with multiple owner devices.

victim devices. We propose an efficient instantiation that can operate under the constraints of existing systems.

2. **Multi-dealer secret sharing.** To obtain the above result, we define and construct a new cryptographic notion: *multi-dealer secret sharing* (MDSS). MDSS extends standard secret sharing to admit multiple dealers with different secrets while achieving new properties of unlinkability and multi-dealer correctness. We provide two constructions of MDSS, including a practical solution (with heuristic correctness) where the decoder can operate on consumer-grade devices like smartphones based on lattice-based list decoding techniques [19]. Finally, we propose some optimizations, including a new notion of *collision-aware PRFs* that enables a small-state streaming algorithm for MDSS dealers.

3. **Implementation and Evaluation.** We explore parameters for our constructions and demonstrate that existing offline finding networks can be made substantially more private using our protocols. Concretely, we propose parameters that allow users to remain private for up to 40 minutes while still detecting stalkers within an hour (see Figure 1). Finally, we implement our algorithms and demonstrate that they are efficient in practice when running on modest hardware. When using our most efficient parameters, our proposed stalker detection algorithm runs in approximately two seconds on a Raspberry Pi 3 B, and a tenth of a second on a modern laptop.

Ethics and human-subjects research. In the course of this work we conducted experiments to determine the density of Apple LTA (AirTag) broadcasts in several public areas. The data we collected in these efforts includes public BLE advertisement data combined with the approximate time and GPS location noted by our detector. Advertisement data includes a rotating MAC address. We did not engage in any effort to link addresses to non-ephemeral device properties, or through rotations, and the results we present in this work contain only aggregate statistics about tag broadcast rates. To eliminate the chance of future address linking, all addresses were anonymized[2], and all data was stored on an encrypted volume to be deleted subject to data retention policies.

Prior to conducting these experiments we presented our experiments to an IRB and received a determination that they do not constitute human subjects research.

1.1 Technical Overview

The core challenge we address in this work is that *the goal of a stalking victim is nearly identical to that of the tracking adversary.* As such, it seems intrinsically difficult to develop systems that allow the former to succeed without providing an advantage to the latter.

Separating stalking victims from tracking adversaries. Our work starts with the following insight: while tracking adversaries must solve a problem that is *similar* to that of a stalking victim, the two parties are *not* identical. Stalking victims are (by definition) guaranteed to be in close proximity to a stalking LTA for a relatively long period. In contrast, a tracking adversary may only have brief or intermittent access to a given tag's broadcasts (*e.g.,* as owners enter and leave the physical locations of tracking receivers). Figure 2 illustrates an example of the two different scenarios.

Our goal is to design a scheme that provides strong cryptographic privacy against a tracking adversary that sees a large subset of an LTA's broadcasts, and yet allows full linking (and thus stalker detection) when the receiver obtains a more complete series of broadcasts. Simultaneously, our scheme must handle broadcasts sent by unrelated tags in the vicinity, that is, be robust to substantial "noise." We refer to this goal as *abuse-resistant offline finding* (AROF).

Abuse-Resistant Offline Finding Protocols. Rather than broadcast a constant identifier, we propose to transmit frequently changing and unlinkable broadcast identifiers as a means to maximize privacy, where the rate of identifier turnover is a system configuration parameter. After a (potential) victim device has collected a list of broadcasts for some fixed amount of time, it can then run a detection algorithm to check if any

[2]MAC addresses were replaced by the HMAC-SHA256 of the address with a randomly sampled 32 byte key. The key was then deleted.

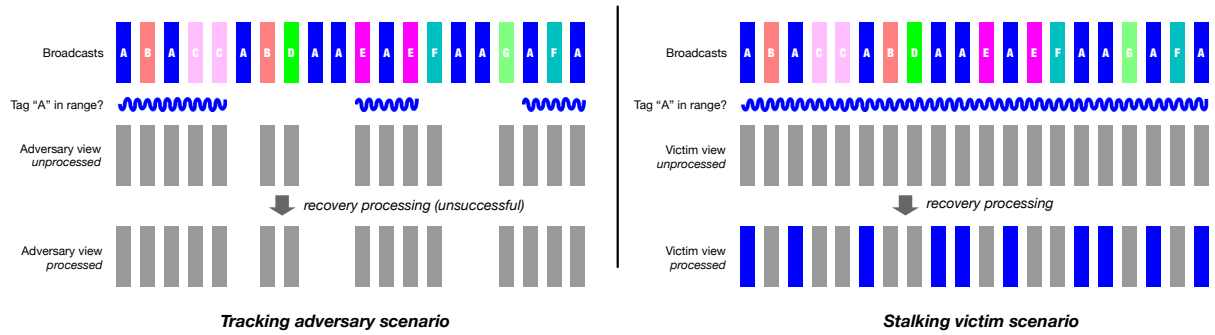

Figure 2: Illustration of two passive detection scenarios. **Left:** a static *tracking adversary* sees many broadcasts from different LTAs, including LTA "A" (blue). However, because "A" is mobile, the adversary is not in continuous range to receive its broadcasts and does not receive enough to enable tracking of the LTA. "A" broadcasts cannot be distinguished from any other broadcasts. **Right:** a *stalking victim* receives broadcasts from many LTAs including a *stalking LTA* "A". The victim is continuously within range of the stalking device and receives all of its broadcasts, which is enough to enable stalker detection.

stalkers are nearby and, if so, it can identify and *link* those broadcasts that come from the stalker. However, to a tracking adversary who has only seen some small number of broadcasts, not only will the detection algorithm fail to recover the user's tag, but it will also be unable to link any of the tag's broadcasts to one another. In fact, the distribution will be cryptographically indistinguishable from a series of broadcasts sent by many different tags.

A key desideratum in AROF is that detection should imply that an LTA has been in *continuous proximity* to a given target, not merely that they have seen a few related broadcasts in distant broadcast periods. More formally, we wish to ensure that a tracking adversary who receives an ϵ-fraction of broadcasts (for some choice of ϵ) during a detection window will have no advantage in linking those broadcasts, while a victim who receives only a slightly greater fraction can efficiently link and detect the stalker. We call the first property *tag indistinguishability* and the second *tag detectability*. The first property ensures that honest tag users are protected from a tracking adversary while the second ensures that stalkers are detected.

A Flawed Attempt. A simple but flawed approach to realizing AROF is to produce a cyclically repeating sequence of distinct broadcast identifiers. Such a construction provides privacy against tracking adversaries who receive broadcasts for a fixed duration, but is very fragile: a tracking adversary who receives only *two* broadcasts (*e.g.,* one broadcast from each cycle) would immediately link these broadcasts to the same LTA. We further discuss this proposal and and its weaknesses in Appendix H.

Our Approach: Multi-Dealer Secret Sharing. To obtain a better solution, we instead use secret sharing [51]. Recall that in classical t-out-of-n secret sharing, a dealer distributes a secret into n shares such that the original secret can be recovered from any subset of shares of size at least t. The privacy property of secret sharing guarantees that any subset of less than t shares (referred to as an *unauthorized* set) "hides" the secret.

Consider the following proposal for an AROF protocol: at periodic intervals, the LTA samples a fresh secret tag identifier. Then, at each epoch within this period, it outputs a pseudonym similar to the one used in Apple's scheme, *along with a secret share of the secret tag identifier.* By changing the pseudonym and using a different secret share at each epoch (and, critically, having volunteer devices keep these shares local without forwarding them to the service provider), we can hope to preserve the privacy of the tag against tracking adversaries who receive an unauthorized subset of the tag's emitted secret shares within the current period. At the same time, a stalking victim who receives a more complete set of shares can use the secret sharing reconstruction algorithm to recover the tag's identifier and communicate with the tag. The exact parameters to determine both privacy and the stalker-detection threshold can be determined by the chosen parameters of the secret sharing scheme, as well as estimates of likely broadcast traffic.

Unfortunately, this proposal runs into several challenges. Note that the broadcast traffic at any time may

consist of broadcasts from *multiple* LTAs in the vicinity of each other. For example, the broadcasts emitted by a stalking LTA might be intermingled with broadcasts from both ephemeral LTAs that a victim is only briefly in range of, as well as additional LTAs that remain persistently near the victim. In this setting, the standard privacy and correctness properties of secret sharing no longer suffice; instead, new properties are necessary in order to realize the above proposal:

1. **Unlinkability:** In order to achieve privacy against tracking adversaries, we require a new *unlinkability* property for secret sharing. Roughly speaking, this guarantees that given a mixed set of shares of different secrets broadcast by multiple dealers, an adversary cannot "link" any of the shares emitted by a dealer to one another as long as the shares comprise an unauthorized set. Note that this is *stronger* than the standard privacy property, which only guarantees that the adversary cannot recover the secret.
2. **MD-Correctness:** In order to detect stalker tags, we require a robust secret recovery mechanism in the presence of *multiple dealers*. In particular, we require that given a set of shares from multiple dealers, a receiver can reconstruct all secrets for which they have an authorized set (i.e., at least a threshold number) of shares. In the context of AROF, this means that a victim can identify potentially multiple stalker tags even in the vicinity of ephemeral tags.

We refer to a secret sharing scheme that satisfies these two properties as *multi-dealer secret sharing* (MDSS). While achieving either of the above two properties is non-trivial, it is especially challenging to achieve them simultaneously. Indeed, in prior uses of secret sharing (e.g., [9, 22, 24]), the problem of robust reconstruction in the presence of noisy shares is addressed by simply *identifying* (or labeling) all shares transmitted by a given dealer to identify which shares correspond to which set. This, however, is *not* an option in our setting since labeling of shares immediately breaks the unlinkability property.

Constructing MDSS. We present two constructions of MDSS. Our first construction is a variant of Shamir secret sharing where instead of using fixed evaluation points (which breaks unlinkability), we sample polynomial evaluation points uniformly at random from a super-polynomial sized field. In order to achieve MD-correctness, we leverage the connection between Shamir's secret sharing and Reed-Solomon (RS) codes[47] and utilize known list decoding algorithms [27] for RS codes. The main disadvantage of this construction is that the parameter restrictions of [27] cap the number of dealers in a way that renders the scheme impractical for our application to AROF.

To achieve better bounds, we employ a variant that uses multiple Shamir polynomials simultaneously, realizing an "interleaved" Reed-Solomon code (IRS) [12]. To achieve MD-correctness, we develop a novel lattice-based list-decoding algorithm building on the work of Cohn and Heninger [19]. This construction achieves provable unlinkability and heuristic correctness guarantees that we justify empirically. Crucially, this approach unlocks significantly better parameters that make efficient decoding achievable on highly-restricted devices. We remark that while efficient decoding algorithms for IRS codes exist in the literature [20, 12], they are defined for the random noise model which does not capture our multi-dealer setting.

Organization. We begin by formalizing the notion of abuse-resistant offline finding protocols in §2. Next, in §3, we give definitions and constructions for our main building block: multi-dealer secret sharing (MDSS). In §4, we show how to use MDSS to construct AROF schemes. In §5 we discuss how to tune MDSS parameters to optimize AROF deployments. In §6 we describe our implementation of MDSS schemes, and in §7 we provide benchmarks of our implementation using our suggested parameters. We discuss some limitations of our approach in §8, related work in §9, and conclude in §10.

2 Abuse-Resistant Offline Finding

An offline finding network, illustrated in Figure 4, is a crowdsourced system designed to locate lost or stolen devices. In these systems, users purchase *location-tracking accessories* (LTAs) that run a tracking protocol with the network and service provider. Deployed networks typically work as follows: to enroll the device, the user pairs the LTA to a client device (such as a smartphone or computer) and optionally registers the LTA

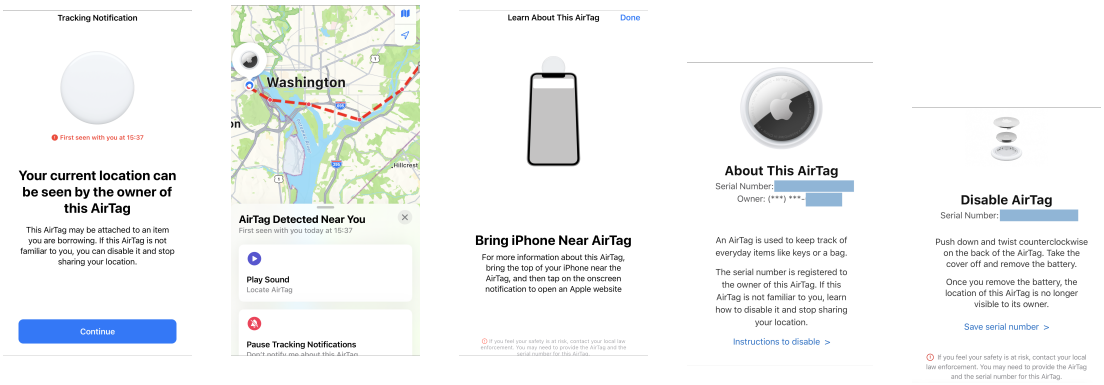

Figure 3: Apple FindMy alerts when an unrecognized AirTag is detected. The rightmost two images are presented on a website launched from the alert screen.

with a service provider (SP) that controls the network. LTAs typically operate in two modes: in *near-owner mode* the LTA is in range of the owner's device and communicates directly with it. The LTA switches to *separated mode* when it is out of range of the owner device. In this work we focus primarily on the behavior of devices in separated mode, which is the typical setting for stalking attacks. For more information on Apple's *FindMy* network, see Appendix A.

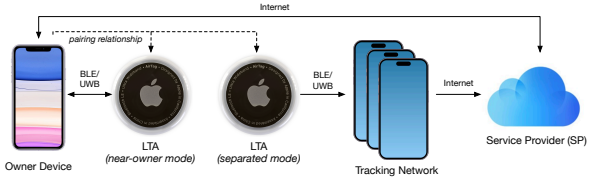

Figure 4: Components of a location-tracking network.

In separated mode, the LTA periodically emits RF-based broadcasts that can be detected by volunteer devices in the offline finding network. These volunteer devices construct *location reports* that combine the LTA broadcast data with the encrypted GPS coordinates of the volunteer device, then upload these reports to the service provider's servers. An owner device with the necessary identifiers (and other credentials) can query the service provider to obtain past and present location reports for a given LTA.

Abuse-Resistant Offline Finding Protocols. The devices in an offline finding network jointly conduct an offline finding protocol. We present algorithmic definitions and security notions for an *abuse-resistant offline finding protocol* that achieves privacy (formalized via a notion of tag indistinguishability) as well as detectability for malicious tags (formalized via a notion of tag detectability). Throughout this section, we use notation inspired by the work of Mayberry *et al.* [36].

While stalking is inherently a malicious activity, the LTA devices used in these attacks are typically *honest*, in the sense that they correctly execute the tracking network protocol. This is due to the fact that many attackers use legitimate (non-counterfeit) devices produced by the original manufacturer. In this work we will focus on the case where all LTAs honestly execute the protocol, and consider extensions to address counterfeit/malicious LTA devices, as well as other concerns, in Appendix G.

Definition 2.1. An *abuse-resistant offline-finding protocol* (AROF) is a tuple of algorithms (KeyGen, Beacon, GetTagID, GenReport, Detect), a protocol RetrieveReports as specified in Figure 5, and two predicates P_s and P_h.

To use an offline-finding protocol, each LTA executes the KeyGen algorithm with a set of deployment-specific parameters, and shares the resulting key with an owner device. The LTA then initializes an *anonymity*

Abuse-resistant Offline Finding Protocol

$\mathsf{KeyGen}(1^\lambda, \mathsf{cfg}) \to k_{\mathsf{tag}}$ given a set of implementation-specific scheme parameters cfg, generates a secret key k_{tag}

$\mathsf{Beacon}(k_{\mathsf{tag}}, i_{\mathsf{epoch}}, \mathsf{aux}) \to B$ on input the tag key, the index of the current *anonymity epoch*, and auxiliary data aux (*e.g.,* battery status), generates a broadcast message B

$\mathsf{GetTagID}(k_{\mathsf{tag}}, i_{\mathsf{epoch}}) \to \mathsf{id}_{\mathsf{tag}}$ is a helper algorithm used by the LTA and owner device to find the current identifier for the LTA.

$\mathsf{GenReport}(B, \mathsf{loc}) \to R$ on input a tag broadcast B and time/GPS coordinates loc, outputs a report R

$\mathsf{Detect}(\mathsf{cfg}, \{(B_1, \mathsf{loc}_1), \ldots, (B_n, \mathsf{loc}_n)\}) \to \{\mathsf{id}_{\mathsf{tag}_i}\}$ on input a set of broadcasts, outputs one or more identifiers $\mathsf{id}_{\mathsf{tag}}$

$\mathsf{RetrieveReports}(\mathsf{Owner}(k_{\mathsf{tag}}, i_{\mathsf{epoch}}), \mathsf{SP}(\mathcal{D}))$ a protocol executed between two parties. Owner provides a tag key k_{tag} and an epoch i_{epoch} and SP uses a database $\mathcal{D}$. The output to Owner is potentially a list of reports and SP's output is $\perp$.

Figure 5: Algorithms for abuse-resistant offline finding protocol.

epoch counter that increases monotonically whenever the identifier is to be rotated (we refer to this period as the *epoch duration.*) At the start of each epoch, the LTA executes the Beacon algorithm on the current epoch counter and transmits the output one or more times.[3] Periodically, after a *detectability period* of some number of anonymity epochs, the LTA may re-key so that very old beacons do not contribute to the tag being detectable. Volunteer devices collect the resulting broadcasts and use the $\mathsf{GenReport}$ algorithm to generate location reports for the service provider. An owner device executes the $\mathsf{RetrieveReports}$ protocol with the service provider to obtain reports collected by the network. Victim devices collect all received LTA broadcasts within a *detection window* and then use them in the Detect algorithm to detect the presence of nearby stalking LTAs.

An abuse-resistant tracking scheme must satisfy correctness and security properties. We first give informal descriptions of the desired properties, followed by the formal definitions.

For the purposes of our definitions, we will use the predicates P_s and P_h to describe specific access patterns that determine whether or not a tag should be *detectable* as a potential stalker, or should remain unlinkable, respectively.

Correctness. An authorized Owner should be able to obtain location information on their LTA, provided a volunteer device sees at least one of its emitted broadcasts.

Detectability. A stalking LTA should be detectable by a victim. Even when selected adversarially, any set of received broadcasts satisfying P_s should both reveal the LTA as a stalker, and result in the victim device accurately learning the stalker's unique identifier.

Tag Indistinguishability. A tracking adversary who receives a broadcast pattern that satisfies P_h should *not* be able to distinguish a given LTA's broadcast from those of other LTAs. We model this as an indistinguishability game, where an adversary may see broadcasts from one of two LTAs, and then must determine which LTA a challenge broadcast belongs to. The advesary wins if it can successfully determine the correct LTA while only having seen broadcast patterns that satisfy P_h.

[3]If the broadcast interval is longer than the epoch duration, then the LTA may transmit the same data multiple times.

Tag Detectability

$\{\{(i_{\ell,j}, \mathsf{aux}_{\ell,j}, \mathsf{loc}_{\ell,j})\}_{j\in[z_\ell]}\}_{\ell\in[L]} \leftarrow \mathcal{A}(1^\lambda)$
$\forall \ell \in [L] : k^\ell_{\mathsf{tag}} \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{cfg})$
$\forall \ell \in [L], j \in [z_\ell] : B_{\ell,j} \leftarrow \mathsf{Beacon}(k^\ell_{\mathsf{tag}}, i_{\ell,j}, \mathsf{aux}_{\ell,j})$
$\mathsf{out} \leftarrow \mathsf{Detect}(\{(B_{\ell,j}, \mathsf{loc}_{\ell,j})\})$
$Q \leftarrow \{i_{\ell,j}\}_{\ell\in[L],j\in[z_\ell]}$
$\mathcal{S} \leftarrow \{P_s(\mathsf{cfg}, k^\ell_{\mathsf{tag}}, Q, i_{\ell,j}) = 1 : \mathsf{GetTagID}(k_{\mathsf{tag}}, i_{\ell,j})\}$
Remove duplicates from $\mathcal{S}$
If $\mathcal{S} \subseteq \mathsf{out}$, output 1 else 0

Figure 6: Experiment $\mathsf{Exp}^{\mathsf{Det},P_s}_{\mathcal{A}}(\lambda, \mathsf{cfg})$.

Tag Indistinguishability

$\mathcal{O}_{k^0_{\mathsf{tag}},k^1_{\mathsf{tag}},Q}(\mathsf{id}, i, \mathsf{aux})$**:** Record (id, i) in table Q. Return $\mathsf{Beacon}(k^{\mathsf{id}}_{\mathsf{tag}}, i, \mathsf{aux})$

$k^i_{\mathsf{tag}} \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{cfg}), \forall i \in \{0, 1\}$
$(i^*_0, i^*_1, \mathsf{aux}) \leftarrow \mathcal{A}^{\mathcal{O}_{k^0_{\mathsf{tag}},k^1_{\mathsf{tag}},Q}(\cdot,\cdot,\cdot)}(1^\lambda)$
$B^* \leftarrow \mathsf{Beacon}(k^b_{\mathsf{tag}}, i^*_b, \mathsf{aux})$
$\hat{b} \leftarrow \mathcal{A}(B^*)$
$Q \leftarrow Q \cup \{(0, i^*_0), (1, i^*_1)\}$
If $P_h(\mathsf{cfg}, Q) = 1$,
output $\hat{b}$, else 0

Figure 7: Experiment $\mathsf{Exp}^{\mathsf{Tag},P_h,b}_{\mathcal{A}}(\lambda, \mathsf{cfg})$.

We can now proceed with the formal definitions.

Definition 2.2 (Correctness)**.** *A privacy preserving tracking protocol satisfies correctness if for all authorized owners* Owner *and compliant service providers* SP*,* $\forall \mathsf{loc}, \mathsf{aux}$ *and allowed anonymity epochs* i_{epoch}*, and* $\forall \mathcal{D}$ *provided by* SP*,*

$$\Pr\left[\begin{array}{l} k_{\mathsf{tag}} \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{cfg}); \\ B \leftarrow \mathsf{Beacon}(\mathsf{cfg}, k_{\mathsf{tag}}, i_{\mathsf{epoch}}, \mathsf{aux}); \\ R \leftarrow \mathsf{GenReport}(B, \mathsf{loc}); \\ \mathcal{D}' := \mathcal{D} \cup \{R\}; \\ \mathsf{out} \leftarrow \mathsf{RetrieveReports}(\mathsf{Owner}(k_{\mathsf{tag}}), \mathsf{SP}(\mathcal{D}')) : \\ \exists m \in \mathsf{out}, m = (\mathsf{loc}, \mathsf{aux}) \end{array}\right] = 1$$

Definition 2.3 (Detectability)**.** *A privacy preserving tracking protocol is detectable if* $\forall$ *valid* cfg *values,* $\forall n.u.p.p.t$ *algorithms* $\mathcal{A}$*,* $\exists$ *a negligible function* $negl(\lambda)$ *so that*

$$\Pr[\mathsf{Exp}^{\mathsf{Det},P_s}_{\mathcal{A}}(\lambda, \mathsf{cfg}) = 0] \leq negl(\lambda),$$

where $\mathsf{Exp}^{\mathsf{Det},P_s}_{\mathcal{A}}(\lambda, \mathsf{cfg})$ *appears in Figure 6.*

Definition 2.4 (Tag Indistinguishability)**.** *A privacy preserving offline finding protocol is tag indistinguishable if* $\forall$ *valid* cfg *values,* $\forall n.u.p.p.t$ *adversaries* $\mathcal{A}$*,* $\exists$ *a negligible function* $negl(\lambda)$ *so that*

$$|\Pr[\mathsf{Exp}^{\mathsf{Tag},P_h,0}_{\mathcal{A}}(\lambda, \mathsf{cfg}) = 1] - \Pr[\mathsf{Exp}^{\mathsf{Tag},P_h,1}_{\mathcal{A}}(\lambda, \mathsf{cfg}) = 1]| \leq negl(\lambda),$$

where $\mathsf{Exp}^{\mathsf{Tag},P_h,b}_{\mathcal{A}}(\lambda, \mathsf{cfg})$ *is given in Figure 7.*

Other Properties. Mayberry *et al.* [36] identify some additional properties for an offline finding system. For example, the service provider should not be able to determine the location of LTAs or volunteer devices from the encrypted location reports it receives. Our schemes in §4 achieve this property in much the same way as the constructions of [36]. Since the focus of our work is on the interaction between stalking tags and victims we omit this analysis here.

3 Multi-Dealer Secret Sharing

In a secret sharing scheme [51], a dealer divides a secret into shares such that the secret remains hidden from anyone who holds an "unauthorized" set of shares, but can be recovered from any "authorized" set of shares. These properties are referred to as privacy and correctness, respectively.

In this work, we consider a setting where *multiple* dealers may simultaneously distribute shares of their respective secrets. We put forth a notion of *multi-dealer secret sharing* (MDSS) that achieves *stronger* security and correctness properties in this setting.

3.1 Definition

We study multi-dealer secret sharing for threshold access structures. Such a scheme is parameterized by four variables:

- t_{rec}, the number of shares required to recover a secret.
- t_{priv}, the number of shares one can emit before privacy and unlinkability are broken.
- d, the number of sufficient dealers the scheme is able to tolerate.
- max, the maximum number of shares that can be input to the reconstruction algorithm.

We consider the *ramp* setting of Blakley and Meadows [11], where we allow for a gap between t_{rec} and t_{priv} larger than 1. We now formally define multi-dealer secret sharing.

Definition 3.1 (Multi-dealer secret sharing scheme)**.** A $(t_{rec}, t_{priv}, d, \mathsf{max})$-multi-dealer secret sharing scheme (MDSS) is defined over a secret space $\mathcal{S}$ of values that can be shared and consists of the following algorithms, which are expected to run in polynomial time in a security parameter λ:

- $\mathsf{Share}(s, n; r) \to \{sh_i\}_{i\in[n]}$, takes as input a secret $s \in \mathcal{S}$, an integer n, and some randomness r, and outputs a set of n shares $\mathbf{sh} = \{sh_i\}_{i\in[n]}$.
- $\mathsf{Reconstruct}(\{sh_1, \ldots, sh_w\}) \to \{s_1, \ldots, s_m\}$, takes as input a set of shares $\{sh_1, \ldots, sh_w\}$ where $w \leq \mathsf{max}$, and outputs a (potentially empty) set of secrets $\{s_1, \ldots, s_m\}$.

In the above, the tuple $(t_{rec}, t_{priv}, d, \mathsf{max})$ is an implicit input to both algorithms.

An MDSS scheme must satisfy two properties: *unlinkability*, which strengthens the standard notion of privacy (§3.1.1), and *MD-Correctness*, which strengthens the standard notion of correctness (§3.1.2).

In both definitions we will use some notational shorthand for computing a set of shares and then projecting them onto a set of indices. For a secret s, an integer n, and a set of indices $\mathcal{I} \subseteq [n]$, define $\mathsf{Proj}(s, n, \mathcal{I})$ as follows:

1. Compute $\{sh_1, \ldots, sh_n\} \leftarrow \mathsf{Share}(s, n)$
2. Return $\{sh_i\}_{i\in\mathcal{I}}$

When $\mathcal{I}$ is a singleton, we will denote it as $\hat{i}$.

3.1.1 Unlinkability

We propose a new security property for secret sharing that is stronger than the standard notion of privacy. Intuitively, this property requires that given a set of shares from multiple dealers, an adversary cannot associate (i.e., link) any unauthorized subset of shares to a common dealer.

We present two definitions: unlinkability and one-more unlinkability. The first intuitively matches the guarantee we would like to achieve, while the second is simpler and easier to use. We will show that the second definition implies the first (the first will obviously imply the second); therefore one can use the second definition without loss of generality.

Unlinkability. Our first definition is characterized by a security game $\mathsf{ULink}^b_{\mathcal{A}}(\lambda)$ where an adversary $\mathcal{A}$ can receive shares from an arbitrary polynomial number of dealers, each sharing a potentially different secret

Experiment $\mathsf{ULink}^b_{\mathcal{A}}(\lambda)$

$\{(s_i, n_{i,0}, n_{i,1}, \mathcal{I}_{i,0}, \mathcal{I}_{i,1})\}_{i\in[L]} \leftarrow \mathcal{A}(1^\lambda)$
$\forall i \in [L] : \mathbf{sh}_i \leftarrow \mathsf{Proj}(s_i, n_{i,b}, \mathcal{I}_{i,b})$
$\hat{b} \leftarrow \mathcal{A}(\mathbf{sh}_1, \ldots, \mathbf{sh}_L)$
Output $\hat{b}$

Figure 8: Unlinkability

Experiment
$\mathsf{OM\text{-}ULink}^b_{\mathcal{A}}(\lambda, t)$

$(s_0, s_1, n_0, n_1, \mathcal{I}, \hat{i}) \leftarrow \mathcal{A}(1^\lambda, t)$
$\{sh_1, \ldots, sh_t\} \leftarrow \mathsf{Proj}(s_0, n_0, \mathcal{I})$
$\{sh^*\} \leftarrow \mathsf{Proj}(s_1, n_1, \hat{i})$
If $b = 1$, $sh_t \leftarrow sh^*$
$\hat{b} \leftarrow \mathcal{A}(sh_1, \ldots, sh_t)$
Output $\hat{b}$

Figure 9: One-More Unlinkability

chosen by the adversary. For every dealer i, the adversary receives $|\mathcal{I}_{i,b}|$ number of shares of the corresponding secret for adversarially chosen index sets $\mathcal{I}_{i,0}$ and $\mathcal{I}_{i,1}$. The adversary wins the game if it correctly predicts the bit b. To disallow trivial attacks, we model security against *admissible* adversaries defined as follows:

Definition 3.2 (Admissible Adversary)**.** *An adversary $\mathcal{A}$ in experiment $\mathsf{ULink}^b_{\mathcal{A}}(\lambda)$ is said to be admissible iff:*

1. $L \in \mathsf{poly}(\lambda)$
2. $\forall i \in [L]$, $|\mathcal{I}_{i,0}| \leq t_{priv}, |\mathcal{I}_{i,1}| \leq t_{priv}$
3. $\sum_i |\mathcal{I}_{i,0}| = \sum_i |\mathcal{I}_{i,1}|$

The second requirement ensures that all share subsets are unauthorized, while the third requirement ensures that the total number of shares are equal in each experiment. The experiment $\mathsf{ULink}^b_{\mathcal{A}}(\lambda)$ is described in Figure 8.

Definition 3.3 (Unlinkability)**.** *A $(t_{rec}, t_{priv}, d, \mathsf{max})$-MDSS with secret space $\mathcal{S}$ is* ϵ-unlinkable *if there exists a function $\epsilon(\cdot)$ such that for all* admissible *adversaries $\mathcal{A}$,*

$$|\Pr[\mathsf{ULink}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{ULink}^1_{\mathcal{A}}(\lambda) = 1]| \leq \epsilon(\lambda)$$

When $\epsilon(\lambda)$ is negligibly small (resp., zero), we say that the scheme is statistically (resp., perfectly) unlinkable. Computational security can be described as well in the standard way.

One-More Unlinkability. Our second definition considers only *two* dealers. The adversary sees up to t_{priv} shares – either all from the first dealer *or* all but one from the first dealer and one share from the second dealer. We formally model this in the security game $\mathsf{OM\text{-}ULink}^b_{\mathcal{A}}(\lambda, t)$ described in Figure 9.

Definition 3.4 (Admissible Adversary)**.** *An adversary $\mathcal{A}$ in experiment $\mathsf{OM\text{-}ULink}^b_{\mathcal{A}}(\lambda, t)$ is said to be admissible iff $|\mathcal{I}| \leq t$.*

Definition 3.5 (One-More Unlinkability)**.** *We say that a $(t_{rec}, t_{priv}, d, \mathsf{max})$-MDSS with secret space $\mathcal{S}$ is* ϵ-one-more unlinkable *if for every $t \leq t_{priv}$, there exists a function $\epsilon(\cdot)$ such that for all* admissible *adversaries $\mathcal{A}$,*

$$|\Pr[\mathsf{OM\text{-}ULink}^0_{\mathcal{A}}(\lambda, t) = 1] - \Pr[\mathsf{OM\text{-}ULink}^1_{\mathcal{A}}(\lambda, t) = 1]| \leq \epsilon(\lambda)$$

We can define statistical, perfect and computational one-more unlinkability in a similar manner as above.

Relationships between definitions. Our proof that the two definitions of unlinkability are equivalent, along with a proof that unlinkability is stronger than the standard secret-sharing notion privacy, can be found in Appendix B.1.

3.1.2 MD-Correctness

Our correctness definition differs from standard secret sharing in that we must reconstruct secrets given a set of shares from *multiple* dealers. This set may include shares from both sufficient dealers (i.e. those outputting at least t_{rec} shares) and *insufficient* dealers (i.e. those outputting fewer than t_{rec} shares). We require that $\mathsf{Reconstruct}$ outputs exactly the set of secrets shared by the sufficient dealers, except with some failure probability ϵ. Similar to standard secret sharing, we only consider correctness for honestly generated (i.e. via the Share algorithm) sets of shares.

Definition 3.6 (MD-Correctness)**.** [4] *We say that a* $(t_{rec}, t_{priv}, d, \mathsf{max})$*-MDSS is* ϵ*-correct if for all sets* $\{(s_i, n_i, \mathcal{I}_i)\}_{i\in[h]}$ *where* $\forall i$*,* $\mathcal{I}_i \subseteq [n_i]$*, and* $\sum_{i\in h} |\mathcal{I}_i| \leq \mathsf{max}$*, and* $\left|\{\mathcal{I}_i \text{ s.t. } |\mathcal{I}_i| \geq t_{rec}\}\right| \leq d$*:*

$$\Pr\left[\mathsf{Reconstruct}(S) \neq \{s_i \text{ s.t. } |\mathcal{I}_i| \geq t_{rec}\} \;\middle|\; S := \bigcup_{i=1}^{h} \mathsf{Proj}(s_i, n_i, \mathcal{I}_i)\right] \leq \epsilon$$

3.2 Constructing MDSS

We present two constructions of MDSS based on variants of Shamir's secret sharing. Our first scheme (§3.2.1) achieves provable MD-correctness for $d < \frac{t_{rec}}{t_{priv}}$ number of dealers and is primarily of theoretical interest. Our second scheme (§3.2.2) achieves heuristic correctness but is able to support both more dealers and a concretely efficient reconstruction algorithm that can be executed on devices with limited capabilities. This forms the basis of our AROF construction presented in the next section.

3.2.1 Construction I

Our first scheme is based on a variant of Shamir's secret sharing scheme [51]. Recall that in Shamir's secret sharing, a secret s is an element of a finite field $\mathbb{F}$: to share s, one samples a random polynomial $p \in \mathbb{F}[z]$ with degree t_{priv} with the constraint that $p(0) = s$. The i-th share is simply the pair $(i, p(i))$ where $i \in \mathbb{F} \setminus 0$ is a fixed evaluation point.

This scheme is *not* unlinkable: consider a set of shares where two of the shares (i, y) and (i', y') are such that $i = i'$ (but $y \neq y'$). Given such a set, an adversary can determine with probability 1 that these shares correspond to *different* secrets. To achieve unlinkability, we therefore we allow the share algorithm to sample each evaluation point uniformly at random from $\mathbb{F} \setminus 0$. If $\mathbb{F}$ has size super-polynomial in the security parameter λ, any two evaluation points collide with probability negligible in λ. Crucially, this probability is the same for any two shares, *irrespective of whether they correspond to the same or different secrets.*

To achieve MD-correctness, we start with the observation that a collection of Shamir secret shares can be viewed as a Reed-Solomon codeword [38]. Therefore, recovering a secret from a collection of Shamir secret shares in the presence of noise is equivalent to the problem of Reed-Solomon decoding, and a natural choice for $\mathsf{Reconstruct}$ is an algorithm based on a Reed-Solomon decoder. Suppose that a single dealer supplies s shares in a set of size $w \leq \mathsf{max}$, and we would like to know if recovering the secret input of this dealer is possible. From a coding theory perspective, we may consider the set of shares to be a word that is distance $w - s$ away from an equivalent length Reed-Solomon codeword corresponding to the dealer's input. As long as an RS decoder exists which can recover from a fraction of $1 - \frac{s}{w}$ errors, we can recover the dealer's secret.

This, however, is not an easy condition to satisfy for a large numbers of dealers; as one example, for any non-trivial scheme with $d > 1$, we cannot use a unique decoding algorithm. Nevertheless, our problem setting can be mapped onto that of list-decoding, which can tolerate higher error (in the ramp setting) and support recovery of multiple polynomials (with enough evaluations). A well-known list-decoding algorithm for Reed-Solomon codes is the one devised by Guruswami and Sudan (GS) [27], and indeed we can use

[4] One could also consider a weakening of this definition where we allow the output of $\mathsf{Reconstruct}(S)$ to contain false positives, namely, some extraneous secrets along with the correct set of secrets. In applications where the extraneous secrets are easily discardable or non-impactful, such a relaxed requirement could be sufficient.

their algorithm as the principal component of Reconstruct to produce a provably correct scheme. The formal description of the scheme can be found in Appendix B.2.

Limitations. The GS-decoder comes with a significant limitation: it can only recover from a $1 - \sqrt{R}$ fraction of errors, where R is the message rate. Phrased in terms of MDSS parameters, this means that the construction only guarantees correctness for parameters satisfying $t_{rec} > \sqrt{\mathsf{max} \cdot t_{priv}}$. When we consider that $\mathsf{max} \geq d \cdot t_{rec}$, as each dealer must contribute at least t_{rec} shares, this gives us $\frac{t_{rec}}{d} > t_{priv}$. Therefore, the gap between t_{rec} and t_{priv} degrades linearly with the number of dealers we wish to tolerate. This is undesirable, and our next construction will focus on achieving a better relationship between the gap and the number of tolerable dealers.

Subsequent works have built list-decoders for *folded* Reed-Solomon codes (which contain multiple polynomial evaluation points in each symbol) [26] that can tolerate nearly $(1 - R)$ (i.e., optimal) error. However, to achieve our desirable trade-off between t_{priv} and t_{rec} using these decoders requires a share-size that is impractical for our AROF application. Furthermore, to the best of our knowledge, these decoding algorithms are not known to be concretely efficient. For further discussion of these and other codes, see Appendix D.

3.2.2 Construction II

Our second construction is inspired by interleaved Reed-Solomon (IRS) codes, which are known to achieve a good trade-off between message and error rate, and could feasibly result in a scheme with a smaller gap between t_{rec} and t_{priv} [20, 12]. For an IRS code, rather than sampling a single polynomial p, a dealer samples c polynomials $p_1, \ldots, p_c$ with the secret s subdivided across the constant term of each, and shares the values $(r, p_1(r), \ldots, p_c(r))$, where the evaluation points r are sampled uniformly at random as in the first construction.

There are numerous decoding algorithms for interleaved Reed-Solomon codes and other closely related noisy curve reconstruction problems in the literature [45, 55]. Unfortunately, all of these algorithms are only known to work in a random noise model - which would correspond to the single dealer setting for $MDSS$ - and most are not efficient enough to operate on limited devices at the parameter sets we require. To address these issues, we devise a concretely efficient, novel decoder that can heuristically decode up to a $\frac{c}{c+1}(1 - R)$ fraction of errors, where c is the number of polynomials, when the channel is "semi-honest" (see Heuristic Assumption 1). The decoder, which we call CH^*-MDSS, is based on a dual version of a lattice-based decoding algorithm proposed by Cohn and Heninger [19].

In their prior work, Cohn and Heninger give a general technique for list-decoding variants of Reed-Solomon codes and their siblings: first recover one (or more) multivariate polynomials Q_i which vanish on all polynomials of the appropriate degree that agree with a large fraction of input points, then recover the common roots of the Q_i. The first step is done by constructing a polynomial lattice and finding a reduced basis, while the second is done by computing a Groebner basis over the Q_i to find all common roots or, in the case where $c = 1$, by factoring a single Q_i.

We instead consider the *dual* of the polynomial lattice, as it directly contains our polynomials of interest and does not require additionally computing a Groebner basis which could be resource intensive. We show that our decoding technique can be used to decode from approximately a $\frac{c}{c+1}(1 - R)$ fraction of random errors, based on an assumption about the distribution of vectors in the polynomial lattice. Next, we show that this decoder can be adapted to our semi-honest channel, where *multiple* codewords must be recovered. The formal description of the algorithm can be found in Appendix C.

Putting all the pieces together brings us to our main construction, which can be seen in Figure 10. We now state the correctness and security guarantees achieved by Construction 10.

To formally state our heuristic assumption regarding the correctness of CH^*-MDSS, we begin by introducing two pieces of notational shorthand. First, we define the "honest" distribution of points that the input to the algorithm will draw from. This distribution is equivalent to sampling points along a collection of polynomials.

Definition 3.7 (Honest Distribution)**.** *For a field $\mathbb{F}$, an integer m, and a set of polynomials $P = \{p_1, \ldots, p_c\}$, define the following distribution $D^{\mathbb{F}}_{m,P}$ on $(\mathbb{F}^{c+1})^m$:*

MDSS Construction

$\underline{\mathsf{Share}(s, n) :}$

Sample c polynomials $p_1, \ldots, p_c$ each of degree t_{priv}, where $s = p_1(0)||...||p_c(0)$

Sample n field elements $x_1, \ldots, x_n \xleftarrow{\$} \mathbb{F}$

return $\{(x_i, p_1(x_i), \ldots, p_c(x_i))\}_{i \in n}$

$\underline{\mathsf{Reconstruct}(\{sh_1, \ldots, sh_w\}) :}$

return $\mathsf{CH}^*\text{-}\mathsf{MDSS}(\{sh_1, \ldots, sh_w\})$

Figure 10: MDSS Construction II.

- *For $i \in [m]$, sample $x_i \xleftarrow{\$} \mathbb{F}$ and set $sh_i := (x_i, p_1(x_i), \ldots, p_c(x_i))$*
- *Return $\{sh_1, \ldots, sh_m\}$*

Next, for a set of points S, let $\mathsf{Unique}(S)$ be true if all the points have a unique x-coordinate, and false otherwise. Finally, for a set $\mathcal{X}$, let $\leftarrow \mathcal{X}$ denote sampling uniformly from $\mathcal{X}$. We can now formally state our heuristic assumption.

Heuristic Assumption 1. *Let $\mathbb{F}$ be a field, and let $c, k, \mathsf{max}, m_1, \ldots, m_h$ be integers where $\mathsf{max} = \sum_{i=1}^{h} m_i$, and let $s_1, \ldots, s_h \in \mathbb{F}^c$ be vectors of field elements. Let $\zeta(\mathbb{F}, k, c, s)$ denote the set of all polynomial sets $P = \{p_1, \ldots, p_c\}$ over $\mathbb{F}$ where $\forall i \in [c]$, $\mathsf{deg}(p_i) \leq k$, and $s = (p_1(0), \ldots, p_c(0))$. Then:*

$$\Pr\left[\mathsf{CH}^*\text{-}\mathsf{MDSS}(S) \neq \left\{ s_i \ s.t. \ m_i \geq \frac{\mathsf{max} + ck}{c+1} \right\} \;\middle|\; \begin{array}{c} \forall i \in [h],\ P_i \leftarrow \zeta(\mathbb{F}, k, c, s_i) \\ \forall i \in [h],\ S_i \leftarrow D^{\mathbb{F}}_{m_i, P_i} \\ S := \bigcup_{i \in [h]} S_i \\ \mathsf{Unique}(S) = \mathsf{True} \end{array} \right] \leq negl(\log |\mathbb{F}|)$$

We can now state our claims regarding the correctness and security of Construction II.

Theorem 3.1. *For all $c \geq 1$ and finite fields $\mathbb{F}$ such that $|\mathbb{F}| \approx \lambda^{\omega(1)}$, construction 10 satisfies ϵ-unlinkability for a negligibly small $\epsilon(\lambda)$.*

Theorem 3.2. *Under Heuristic Assumption 1, for all sets of parameters satisfying $t_{rec} \geq \frac{1}{c+1}(\mathsf{max} + (c \cdot t_{priv}))$ and finite fields $\mathbb{F}$ such that $|\mathbb{F}| \approx \lambda^{\omega(1)}$, construction 10 satisfies ϵ-*correctness *for a negligibly small $\epsilon(\lambda)$.*

We defer the proofs of these theorems to Appendix B.3.

3.2.3 Optimizations

We now discuss some efficiency optimizations.

Small Fields. A suitable choice of field is required to achieve practical unlinkability and efficiency. If a dealer randomly samples the same evaluation point twice and produces two identical shares, then unlinkability is compromised. While this will occur with negligible probability if fields are large enough, in practice small fields are often best to minimize bandwidth and decoding complexity. We therefore propose a stateful variant of our construction in which dealers keep track of previously sampled evaluation points and emit *noise shares* (comprising random elements in place of the polynomial evaluation) whenever an x-coordinate is repeated. This preserves unlinkability at the cost of decreasing the number of "useful" shares passed to the reconstruction algorithm, and thereby increasing the probability of reconstruction failure.

Collision-Aware PRF. In practice, MDSS users may want to output shares in a *streaming* fashion.This would require keeping state (all previously sampled x-coordinates), which may be undesirable. As an optimization for this setting, we define a new primitive we call a *Collision-Aware PRF* (CA-PRF). Using a

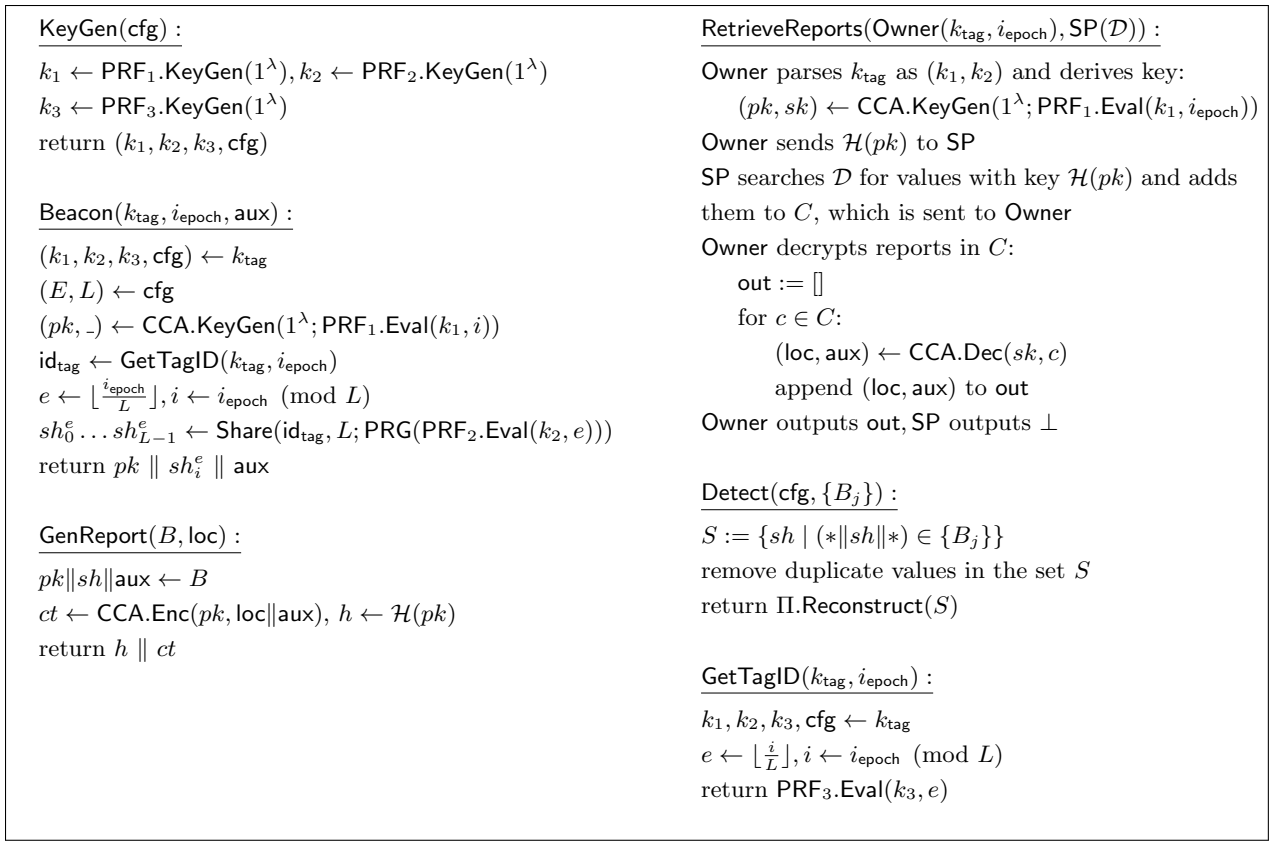

Figure 11: Main construction for abuse-resistant offline finding. This protocol assumes the existence of three pseudorandom functions PRF_i for $i \in [3]$ where the co-domain of $\mathsf{PRF}_1.\mathsf{Eval}$ and $\mathsf{PRF}_2.\mathsf{Eval}$ is $\{0,1\}^\lambda$ and the codomain of $\mathsf{PRF}_3.\mathsf{Eval}$ is $\mathbb{F}_q$. PRG outputs a sufficient number of bits for the Share algorithm. CCA is a CCA-secure PKE scheme, $\mathcal{H}$ a collision resistant hash function and Π a $(t_{rec}, t_{priv}, d, \mathsf{max})$-MDSS sharing scheme. $\|$ denotes concatenation. cfg contains the $MDSS$ parameters $t_{priv}, t_{rec}, \mathsf{max}, d$ and a separate parameter L.

CA-PRF allows an MDSS dealer to sample x-coordinates pseudorandomly while being alerted to potential collisions.

For a formal treatment of both of the above, see Appendices B.4 and B.5.

4 Construction of AROF Protocols

We now describe our main contribution: an offline finding protocol that achieves strong privacy while also admitting stalker detection.

Protocol intuition. The core algorithms of our protocol are presented in Figure 11. At a basic level, this protocol can be viewed as an extension of the offline finding construction used by Apple's FindMy [1]. Like the Apple construction, our protocol employs a pseudorandom function to derive a fresh "pseudonym" for the LTA at the start of each epoch. Just as in the Apple system, each pseudonym also doubles as a public key for a secure encryption scheme: upon receiving a pseudonym from any nearby LTA, volunteer devices in the tracking network can encrypt their current GPS coordinates and transmit the resulting ciphertext to the service provider. Owner devices can later re-derive a sequence of missed pseudonyms and the corresponding decryption keys, and use the RetrieveReports protocol to query the provider for each missed time period.

Achieving stalker detection. To enable stalker detection, our protocol makes several additions to the

basic protocol described above. First, each LTA maintains a *detectability period* consisting of L consecutive time epochs. At the start of each detectability period, the LTA generates a secret tag identifier $\mathsf{id}_{\mathsf{tag}}$. It then secret-shares $\mathsf{id}_{\mathsf{tag}}$ using an MDSS scheme configured with appropriate parameters. With each call to Beacon, the LTA generates the current pseudonym and appends one secret share to be broadcast by the LTA. The MDSS parameters are informed by how long it should take before a stalker is detected, the maximum number of stalkers that should be detected, and the maximum number of beacons that will be within any victim's detection window. We discuss more in depth how these parameters are set in §5.

Critically, volunteer devices in the offline finding network *do not transmit these secret shares to the service provider*: they are kept locally and used only to enable stalker detection. Each device maintains a set of all shares received from nearby LTAs during a time window specified in the deployment parameters. Periodically, the victim device executes the Detect algorithm to perform secret sharing recovery. If this collection contains at least t_{rec} shares emitted by one LTA (or a similar set from multiple LTAs), then the MDSS recovery algorithm will recover each $\mathsf{id}_{\mathsf{tag}}$ for the devices. Each LTA can be configured to respond to interactive connections containing $\mathsf{id}_{\mathsf{tag}}$, which enables the victim to contact and physically locate any stalking LTAs.

We build security for predicates that mostly follow the intuition from the technical overview. If t_{rec} beacons from a tag are seen within a detection window, then it is considered to be a stalking tag. Intuitively, t_{rec} is close to the number of broadcasts made in the detection window, so the victim sees almost all broadcasts emitted by the LTA. Privacy is maintained as long as no party sees more than t_{priv} beacons from any detectability period.

Security. We provide formal descriptions of the predicates, as well as theorems and proofs of security for our construction in Appendix E. Tag indistinguishability and detectability both follow naturally from the unlinkability and correctness properties of the underlying MDSS.

4.1 Implementation Considerations

We now discuss particular implementation choices that a designer must grapple with if they wish to use the scheme presented in Figure 11.

De-duplication and filtering. The duration of the anonymity epoch (*i.e.,* the time between identifier changes) is an important deployment consideration in our scheme. A shorter interval is clearly desirable to improve privacy against tracking adversaries. However, the duration of the anonymity epoch will also affect the efficiency of stalker detection. To illustrate these considerations we consider two candidate configurations.

Configuration 1: anonymity epoch ≈ broadcast interval. This is our recommended configuration, which maximizes privacy against tracking adversaries by minimizing the duration of the anonymity epoch. Under a 4-second anonymity epoch, this will result in a new pseudonym (via a call to Beacon) every broadcast that the LTA emits.[5] Our implementation transmits twice as much broadcast data at each interval, and so we propose to split each broadcast B into two separate transmissions, each sent at 4-second broadcast interval (see §6), producing 900 hourly unique secret shares from each LTA device.

Configuration 2: anonymity epoch ≫ broadcast interval. Current LTA deployments do not change the pseudonym with each broadcast.[6] This decision is likely motivated by computational costs and battery limitations. In these deployments, the LTA will re-broadcast each pseudonym many times. Applying the same logic to our protocol, these LTAs would also re-broadcast the same secret share.

Since duplicate secret shares do not aid in reconstruction, they are removed (de-duplicated) as part of the Detect algorithm. We note that this de-duplication procedure can produce some counter-intuitive effects on the noise rate. For example, LTAs that remain within range of a victim device for long periods of time (*e.g.,* stalking tags) will see proportionally more duplicate broadcasts removed, as compared to ephemeral LTAs that only briefly enter receive range of a victim device (see Figure 12.) On the positive side, these repeated broadcasts dramatically reduce the impact of erasures caused by RF-layer issues.

[5] This is based on analysis of Apple's FindMy, where LTAs broadcast every 2 seconds [30, 29].

[6] For example, Apple's Find My rotates the pseudonym every 450 broadcasts (15 minutes) in near-owner mode.

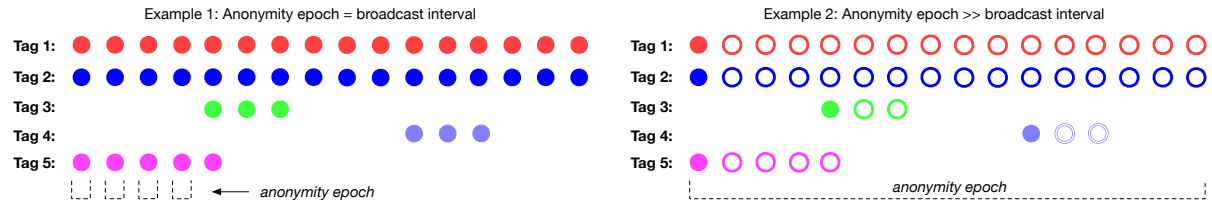

Figure 12: Illustration of the effect of de-duplicating repeated LTA broadcasts: both examples contain the same pattern of transmissions, but use different anonymity epoch durations. Filled circles represent fresh shares, hollow circles represent duplicate shares. **Left:** The anonymity epoch is short: persistent tags (Tag 1, Tag 2) produce the majority (32/43) of the unique broadcasts processed by the Detect algorithm. **Right:** The anonymity epoch is long: once duplicates are removed, the persistent tags (Tag 1, Tag 2) represent a minority (2/5) of the unique broadcasts processed by the decoding algorithm.

Pre-filtering. To compensate for the over-representation of non-stalking LTAs, systems with long-duration anonymity epochs can apply a *pre-filtering* heuristic prior to executing Detect. This heuristic takes advantage of the fact that ephemeral (*non-stalking devices*) are likely to exhibit broadcast behavior that can be recognized and filtered out prior to running Detect. For example, LTAs that remain in close proximity to the victim device for a period of time will transmit many duplicate broadcasts. Focusing detection exclusively on these broadcasts will reduce the number of non-stalker shares processed by the MDSS recovery algorithm.

Secret re-generation and detection window. In our construction, each LTA re-generates its tag identifier and secret-sharing polynomials every L epochs. This mechanism is intended to prevent tracking adversaries from correlating shares sent by the same LTA over longer periods. However, this periodic change of secrets poses a challenge for stalker detection: at boundary epochs where an LTA changes its secret, the victim may not receive a sufficient number of shares to detect a stalker. To address this, we propose two possible deployment options. A first solution is for a device to re-generate secrets relatively rarely. For example, performing re-generation every 24 hours ensures that such issues will only occur during a small fraction of each day. Second, a device could maintain two identifiers. The LTA would emit shares on both, and stagger their rotation to ensure that detection is always possible.

MAC Address Rotation To deploy with our suggested anonymity epochs would require rotating the LTA's MAC address either every 4 or 60 seconds, which is faster than the minimum rotation period of 15 minutes recommended by the BLE specification[13]. Care should therefore be taken during implementation to ensure that core BLE functionality is preserved. We note that cautious implementers could follow the recommendation and use an anonymity epoch of 15 minutes, which still represents a substantial privacy improvement over the state-of-the-art.

5 Selecting MDSS Parameters

Deploying our main construction of §4 requires us to identify reasonable system parameters for both the AROF and MDSS schemes. The challenge is to optimize privacy while also enabling *efficient* recovery in the face of reasonable noise rates. Our recommended parameters can be seen in Table 1. We now briefly describe the process we used to derive them.

High-level Considerations. We identify five high-level configuration parameters, and for the purposes of our analysis made exemplary choices for each. We note that these can be adjusted for specific deployments. Our identified parameters are:

1. The (minimum) number of minutes of continuous broadcasting after which a victim device must be able to detect a stalker LTA, which we set to 1 hour.
2. The duration of the anonymity epoch (*i.e.,* the time between pseudonym rotation), for which we consider 4 seconds and 1 minute.

3. The frequency of LTA secret/polynomial updates, which we set to 24 hours.
4. The maximum number of LTAs that will be in proximity to a victim device. We assume there will be at most three persistent ("stalker") LTAs at any given time, and at most a number of ephemeral LTAs equal to half a stalking LTA in the aggregate.
5. The bandwidth available for transmitting each secret share, which we derive to be 248 bits, based on available payload space in BLE, and 400 bits for BLEv5.

From these choices, we then optimize our MDSS scheme parameters to provide privacy for the longest period possible, while ensuring $> 99\%$ decoding success in the presence of missed broadcasts. We present two sets of parameter choices in Table 1: a recommended set given current deployment bandwidth limitations (based on widely-deployed versions of BLE), as well as a future recommendation for LTAs that support versions of BLEv5 with higher bandwidth limits. For further information on how we derived these parameters, see Appendix F.

Epoch duration	Detect time	Max stalkers	L (= 24 hrs)	$\mathbb{F}_p$ (bits)	c	Share (bits)	Privacy time	t_{rec}	t_{priv}	max
Recommended parameters (compatible with current LTA bandwidth limits):										
4 sec	60 min	3	21,600	22	10	242	39 min	825	591	3150
1 min	60 min	3	1,440	24	9	240	41 min	59	41	210
Future parameters (compatible with BLE v5 bandwidth limits):										
4 sec	60 min	3	21,600	22	17	396	46 min	825	687	3150
1 min	60 min	3	1,440	26	14	390	47 min	59	47	210

Table 1: Recommended and possible future parameters for the MDSS construction of §4, instantiated with CH*-MDSS. Privacy times are rounded to the nearest minute.

6 Implementation

We now describe our implementation of our construction in §4, including optimizations and deployment choices.[7]

Implementing the Detect algorithm. We implemented the Detect algorithm using SageMath [53], a Python-based computer algebra system that includes fast C implementations for many algorithms. Our decoder for CH*-MDSS is around 400 lines of Python and is the bulk of our Detect algorithm implementation. For each parameter set in Table 1, *e.g.,* $c = 10$ with a field size of 22 bits, the decoding matrix requires less than 10 MB, even for the largest values of max. This is compatible with available application RAM on current smartphones.

Implementing the Beacon algorithm. For ease of comparison, we adopt the algorithmic choices of Apple's FindMy scheme: our implementation uses ECIES public keys [5] over the NIST P-224 curve. During each call to Beacon we define a subroutine GeneratePublicKey to pseudorandomly generate a new pseudonym pk from the master key. We implement this portion of the algorithm in SageMath.

To implement the secret sharing scheme we use the recommended parameters from Table 1, and split the Beacon algorithm into two stateful subroutines. At the beginning of each detection period (every L epochs), we execute GeneratePolynomials to sample polynomials $p_1, \ldots, p_c$, and cache these polynomials during the entire period. The tag password ($\mathsf{id}_{\mathsf{tag}}$) is the concatenation of the constant terms of each polynomial. During every call to Beacon we execute the subroutine GenerateSecretShare to pseudorandomly generate $r \in \mathbb{F}$ and compute a secret share. We implemented these functions in SageMath, as well as in a combination of JavaScript and C in order to run experiments on a low-powered hardware platform. For more information on the hardware used, see §7.

[7] Our code can be found at `https://github.com/becgabri/abuse-resistant-private-lt`

Epoch duration	Avg. nearby LTAs	# Unique bxs received	Hardware	Detection runtime (no stalkers)	Detection runtime (stalkers present)	Polynomial recovery (all stalkers)
4 sec	1	1350	MacBook	0.60 sec	1.29 sec	1.37 sec
			RPi 3	8.07 sec	20.42 sec	20.99 sec
	2	2250	MacBook	1.88 sec	5.57 sec	8.74 sec
			RPi 3	25.05 sec	83.91 sec	126.18 sec
	3	3150	MacBook	3.86 sec	24.59 sec	32.96 sec
			RPi 3	51.83 sec	360.95 sec	484.21 sec
60 sec	1	90	MacBook	0.01 sec	0.01 sec	0.01 sec
			RPi 3	0.15 sec	0.27 sec	0.27 sec
	2	150	MacBook	0.02 sec	0.03 sec	0.04 sec
			RPi 3	0.31 sec	0.57 sec	0.73 sec
	3	210	MacBook	0.03 sec	0.08 sec	0.12 sec
			RPi 3	0.48 sec	1.27 sec	2.05 sec

Table 2: Benchmarks for our Detect algorithm (Figure 11) using the CH*-MDSS decoding algorithm with the recommended parameters of Table 1.

Detection in practice. Since we will be executing detection on resource-constrained devices, we propose that devices should continuously collect shares from the environment into a ring buffer containing the most recent N minutes of shares. The Detect algorithm can then be run every few minutes once the buffer is full. This strategy provides a relatively timely warning of stalking behavior and can be adjusted to account for computational resources. Additionally, once a stalker has been detected, their polynomial can be cached and the shares they emit quickly filtered out. This means that the expensive cases with many persistent LTAs should only occur rarely.

7 Experiments

In this section we evaluate the empirical runtime of our stalker-detection and encoding algorithms. We used three different pieces of hardware to conduct our experiments: a single core of a 2020 MacBook computer with an M1 chip and 16 GB of RAM running MacOS Ventura 13.3.1 (a), a Raspberry Pi 3 Model B running Ubuntu Server 23.04, and a Puck.js, a hardware platform based on the same SoC as the AirTag, running firmware version 2.15.

Stalker detection. To conduct this analysis we ran our CH*-MDSS-based Detect algorithm on simulated broadcasts. We split the experiments into two cases:

1. Decoding when *stalkers are present*, i.e. at least t_{rec} shares originate from the same LTA.
2. Decoding when *no stalkers are present*, the common case for most users, when no single (unknown) LTA contributes t_{rec} shares to the input.

We split these cases further to test two different anonymity epochs: 4 seconds and 60 seconds. More concretely, we performed the following experiment:

1. For each anonymity epoch (4 sec, 60 sec), and for each number of stalkers (1, 2, 3), we determine the maximum number of shares our decoder could receive (e.g. for a 4 second epoch and 2 stalkers, the decoder could receive a maximum of $(3600/4) * (2 + 0.5) = 2250$ shares).
2. We then run two experiments: first where that number of shares is generated by a combination of stalkers and ephemeral LTAs, and second where the shares are entirely ephemeral. For the first case we measure both the time to *detection* (i.e. determining the existence of a stalker), and the time to *recovery* (learning the id for all stalkers). We average the runtimes of each experiment over 500 iterations.

For each experiment, we generated the data so that all broadcasts have a unique x-coordinate in order to evaluate a fixed number of points passed to the decoding algorithm and produce an upper bound on the

Algorithm	t_{priv} (epoch)	Runtime MacBook	Runtime Puck.js
GeneratePublicKey		484 μs	
GeneratePolynomials	591 (4 sec)	6233 μs	11.88 sec
GenerateSecretShare	591 (4 sec)	87 μs	0.26 sec
GeneratePolynomials	41 (60 sec)	451 μs	0.72 sec
GenerateSecretShare	41 (60 sec)	11 μs	0.04 sec

Table 3: Runtime average over 1,000 iterations for the GeneratePublicKey, GenerateSecretShare and GeneratePolynomials subroutines of the Beacon algorithm. Secret sharing uses the 22-bit field with $c = 10$ for the 4 second epoch and the 24-bit field with $c = 9$ for the 60 second epoch.

projected runtime. Our decoder was implemented as described in §6 and configured according to Table 1. Table 2 presents our running times, which for common cases take seconds.

Encoding. We benchmarked our encoding algorithms on both the laptop and Puck.js. Table 3 shows the running times of both.

8 Limitations

The AROF scheme proposed in this work comes with some limitations. First, the number of stalking LTAs that the scheme can tolerate must be fixed at the time of deployment. Should a user be stalked by more than the number of LTAs chosen at the outset, they will be unable to detect any of the malicious LTAs. We chose a maximum of three stalking LTAs based on a privacy/cost tradeoff. At the time of writing a four-pack of AirTags costs $99, which is comparable in price to a self-contained GPS tracker. Larger numbers of tolerable stalkers can be chosen at the outset, but would result in lower privacy times than those shown in Table 1. For example, for the 4 second anonymity epoch, tolerating up to 4 stalkers would result in 33 minutes of privacy, 5 stalkers 28 minutes, and 6 stalkers 22 minutes. We note that these numbers were computed assuming the BLE broadcast size is fixed, and as discussed in §5 and Appendix F, privacy time can be increased by assuming a larger broadcast.

Additionally, our scheme is weak to large levels of collusion among trackers. We assume that the privacy adversary is not a global adversary. If they are, then the stalking victim no longer has an asymmetry, and passive stalker detection protocols are unlikely to be compatible with privacy.

9 Related Work

Offline finding. Several works have considered the privacy and integrity of the offline finding (tracker) ecosystem. Heinrich, Bittner and Hollick [29] evaluated the anti-stalking mechanism use in FindMy. Heinrich, Stute, Kornhuber and Hollick [30] also considered the privacy of Apple's FindMy protocol. Mayberry *et al.* [37] considered ways to bypass tracking alerts in AirTag devices, and in a separate work Mayberry, Blass and Fenske [36] devised protocols to protect against counterfeit tags. Depres *et al.* [23] develop a method to detect stalking devices with frequently rotating MAC addresses using physical-layer techniques such as RSSI.

Most of the previous works do not consider the attack we attempt to mitigate in this paper: tracking a user via the unchanging identifier of their own LTAs. [30] mentions such a concern, but dismisses it as the authors study only devices which rotate their identifier every 15 minutes. Similar to our work, [23] proposes a stalker-detection mechanism for LTAs with frequently changing identifiers. However, their work focuses on linking broadcasts via physical-layer attributes, whereas our work is entirely on the protocol level. Additionally, our work is the only proposal that prevents the tracking of an LTA until a signficant time in contact with it has passed.

Secret sharing. Many schemes use secret sharing for privacy applications: here we focus on some recent works closely related to ours. The Apple PSI system [9] defines the notion of a *detectable hash function* based on interleaved Reed-Solomon codes [12]. Similarly, the STAR protocol of Davidson *et al.* [22] emits shares of (multiple) secrets for private telemetry reporting: unlike our work, their protocol assumes that the decoder can recognize all shares from a dealer and so noise and unlinkability concerns are not considered. Finally, the literature on robust secret sharing (RSS) and cheater detection is itself robust, and considers many different models (see [46, 14, 15, 39, 54, 42, 17]) . A key difference between the classical RSS setting and our setting is that in RSS there is exactly one dealer; crucially, RSS does not incorporate the notion of unlinkability.

Finally, we note that our notion of unlinkability of secret sharing differs from other notions of *unlinkability* in the cryptography literature in the context of primitives such as ring signatures and anonymous credentials [21, 3, 7]. These works consider *unbounded* unlinkability, where the property holds regardless of the sample size given to the adversary. In contrast, our notion of unlinkability is necessarily in the *bounded* setting, where the adversary is permitted to see any unauthorized set of shares of a secret.

10 Conclusion and Future Work

In this work we considered the problem of constructing privacy-preserving tracking protocols that enable efficient abuse detection. We demonstrate that the use of secret sharing enables privacy-preserving offline finding while also admitting efficient algorithms for detecting stalkers. This work leaves several open questions for future work, including the development of efficient protocols secure against malicious/counterfeit tags, as well as the development of improved multi-dealer secret sharing schemes.

References

[1] FindMy: One App to Find it All. `https://www.apple.com/icloud/find-my/`.

[2] Apple Platform Security: Find My. Available at `https://support.apple.com/guide/security/find-my-security-sec6cbc80fd0/1/web/1`, 2023.

[3] Masayuki Abe, Miyako Ohkubo, and Koutarou Suzuki. 1-out-of-n signatures from a variety of keys. In Yuliang Zheng, editor, *Advances in Cryptology — ASIACRYPT 2002*, pages 415–432, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[4] Apple. An update on AirTag and unwanted tracking. Available at `https://support.apple.com/guide/security/find-my-security-sec6cbc80fd0/1/web/1`, February 2022.

[5] Richard Barnes, Karthikeyan Bhargavan, Benjamin Lipp, and Christopher A. Wood. Hybrid Public Key Encryption. RFC 9180, February 2022.

[6] Mihir Bellare, Thomas Ristenpart, Phillip Rogaway, and Till Stegers. Format-preserving encryption. In Michael J. Jacobson, Vincent Rijmen, and Reihaneh Safavi-Naini, editors, *Selected Areas in Cryptography*, pages 295–312. Springer Berlin Heidelberg, 2009.

[7] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. *J. Cryptology*, 2009.

[8] Lindsey Bever. She tracked her boyfriend using an AirTag — then killed him, police say. *Washington Post*, June 2022.

[9] Abhishek Bhowmick, Dan Boneh, Steve Myers, Kunal Talwar, and Karl Tarbe. The Apple PSI system. Available at `https://www.apple.com/child-safety/pdf/Apple_PSI_System_Security_Protocol_and_Analysis.pdf`, August 2021.

[10] John Black and Phillip Rogaway. Ciphers with arbitrary finite domains. In Bart Preneel, editor, *CT-RSA 2002*, pages 114–130, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[11] G. R. Blakley and Catherine Meadows. Security of ramp schemes. pages 242–268, 1984.

[12] Daniel Bleichenbacher, Aggelos Kiayias, and Moti Yung. Decoding of Interleaved Reed Solomon Codes over noisy data. In *ICALP '03*, 2003.

[13] Bluetooth SIG. *Core Specification*, 12 2014. Version 4.2.

[14] E. F. Brickell and D. R. Stinson. The detection of cheaters in threshold schemes. In Shafi Goldwasser, editor, *Advances in Cryptology — CRYPTO' 88*, pages 564–577, New York, NY, 1990. Springer New York.

[15] Marco Carpentieri, Alfredo De Santis, and Ugo Vaccaro. Size of shares and probability of cheating in threshold schemes. In Tor Helleseth, editor, *Advances in Cryptology — EUROCRYPT '93*, pages 118–125, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[16] CBS Chicago Team. Man killed girlfriend after she removed AirTag he'd secretly placed in her car, prosecutors say. *Channel 2 CBS News Chicago*, July 2023.

[17] Alfonso Cevallos, Serge Fehr, Rafail Ostrovsky, and Yuval Rabani. Unconditionally-secure robust secret sharing with compact shares. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, pages 195–208, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[18] Mitchell Clark. Samsung wants to make sure nobody's tracking you with its SmartTags. *The Verge*, April 2021.

[19] Henry Cohn and Nadia Heninger. Approximate common divisors via lattices. *The Open Book Series*, 1(1):271–293, 2013.

[20] Don Coppersmith and Madhu Sudan. Reconstructing curves in three (and higher) dimensional space from noisy data. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '03, page 136–142, New York, NY, USA, 2003. Association for Computing Machinery.

[21] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo G. Desmedt, editor, *Advances in Cryptology — CRYPTO '94*, pages 174–187, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[22] Alex Davidson, Peter Snyder, E. B. Quirk, Joseph Genereux, Benjamin Livshits, and Hamed Haddadi. Star: Secret sharing for private threshold aggregation reporting. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 697–710, New York, NY, USA, 2022. Association for Computing Machinery.

[23] Tess Despres, Noelle Davis, Prabal Dutta, and David Wagner. Detagtive: Linking macs to protect against malicious ble trackers. In *Proceedings of the Second Workshop on Situating Network Infrastructure with People, Practices, and Beyond*, pages 1–7, 2023.

[24] Casey Devet, Ian Goldberg, and Nadia Heninger. Optimally robust private information retrieval. In *USENIX Security Symposium*, pages 269–283, 2012.

[25] FOX 7 News. Texas man uses Apple Airtag to track down stolen truck, then shoots, kills suspect: police. Available at `https://www.fox7austin.com/news/san-antonio-texas-apple-airtag-stolen-truck-suspect-killed-police`, April 2023.

[26] V. Guruswami and A. Rudra. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. *IEEE Trans. Inf. Theor.*, 54(1):135–150, jan 2008.

[27] V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. In *Proceedings 39th Annual Symposium on Foundations of Computer Science*, pages 28–37, 1998.

[28] Bernhard Haeupler and Amirbehshad Shahrasbi. Synchronization strings and codes for insertions and deletions—a survey. *IEEE Transactions on Information Theory*, 67(6):3190–3206, 2021.

[29] Alexander Heinrich, Niklas Bittner, and Matthias Hollick. Airguard - protecting android users from stalking attacks by apple find my devices. In *WiSec '22*, 2022.

[30] Alexander Heinrich, Milan Stute, Tim Kornhuber, and Matthias Hollick. Who can find my devices? security and privacy of apple's crowd-sourced bluetooth location tracking system. *Proc. Priv. Enhancing Technol.*, 2021(3):227–245, 2021.

[31] Johana Bhuiyan and agencies. Apple and Google submit plan to fight AirTag stalking. *The Guardian*, May 2023.

[32] Thomas Kailath. *Linear systems*, volume 156. Prentice-Hall Englewood Cliffs, NJ, 1980.

[33] Erik Kay. 3 ways unknown tracker alerts on Android help keep you safe. Available at `https://blog.google/products/android/unknown-tracker-alert-google-android/`, July 2023.

[34] Brent Ledvina, Zachary Eddinger, Ben Detwiler, and Siddika Parlak Polatkan. Detecting Unwanted Location Trackers. Available at `https://datatracker.ietf.org/doc/draft-detecting-unwanted-location-trackers/00/`, May 2023. Work in Progress.

[35] Adrienne Matei. 'I was just really scared': Apple AirTags lead to stalking complaints. *The Guardian*, January 2022.

[36] Travis Mayberry, Erik-Oliver Blass, and Ellis Fenske. Blind My: An improved cryptographic protocol to prevent stalking in Apple's Find My network. In *PoPETS '23*, volume 2023.

[37] Travis Mayberry, Ellis Fenske, Dane Brown, Jeremy Martin, Christine Fossaceca, Erik C. Rye, Sam Teplov, and Lucas Foppe. Who tracks the trackers? circumventing apple's anti-tracking alerts in the find my network. In *WPES '21*, page 181–186, 2021.

[38] R. J. McEliece and D. V. Sarwate. On sharing secrets and reed-solomon codes. *Commun. ACM*, 24(9):583–584, sep 1981.

[39] R. J. McEliece and D. V. Sarwate. On sharing secrets and Reed-Solomon codes. *Commun. ACM*, 24(9):583–584, sep 1981.

[40] T. Mulders and A. Storjohann. On lattice reduction for polynomial matrices. *Journal of Symbolic Computation*, 35(4):377–401, 2003.

[41] Caleb Naysmith. Apple AirTags and Bluetooth Trackers Are Officially a Billion-Dollar Industry. *Yahoo News*, December 2022.

[42] Satoshi Obana. Almost optimum t-cheater identifiable secret sharing schemes. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, pages 284–302, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[43] F. Parvaresh and A. Vardy. Correcting errors beyond the Guruswami-Sudan radius in polynomial time. In *FOCS '05*, pages 285–294, 2005.

[44] Vasile Mihai Popov. Invariant description of linear, time-invariant controllable systems. *SIAM Journal on Control*, 10(2):252–264, 1972.

[45] Sven Puchinger and Johan Rosenkilde né Nielsen. Decoding of interleaved reed-solomon codes using improved power decoding. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 356–360. IEEE, 2017.

[46] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, STOC '89, page 73–85, New York, NY, USA, 1989. Association for Computing Machinery.

[47] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.

[48] RetailNext. RetailNext Announces General Availability of Aurora Sensor. Available at `https://retailnext.net/press-release/retailnext-announces-general-availability-of-aurora-sensor`, 2023.

[49] Nat Rubio-Licht. Tile adds anti-stalking feature after AirTag backlash. *Protocol*, March 2022.

[50] SensorMatic. ShopperTrak Traffic Insights. Available at `https://www.sensormatic.com/shoppertrak-retail-traffic-insights`, 2023.

[51] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, November 1979.

[52] Jonathan Stempel. Apple is sued by women who say AirTag lets stalkers track victims. *Reuters*, December 2022.

[53] The SageMath Developers. SageMath, June 2023.

[54] Martin Tompa and Heather Woll. How to share a secret with cheaters. In *Proceedings on Advances in Cryptology—CRYPTO '86*, page 261–265, Berlin, Heidelberg, 1987. Springer-Verlag.

[55] Jiun-Hung Yu and Hans-Andrea Loeliger. Simultaneous partial inverses and decoding interleaved reed–solomon codes. *IEEE Transactions on Information Theory*, 64(12):7511–7528, 2018.

A Details of Apple's FindMy Protocol

The Apple FindMy protocol has been extensively reverse-engineered by Heinrich *et al.* [30]. Here we briefly summarize some relevant findings.

Privacy, device types, separated mode. As discussed in §2, the FindMy protocol operates differently for LTA and non-LTA devices such as phones and tablets. Non-LTA devices maintain a single identifier that rotates every 15 minutes. AirTags maintain two separate identifiers: the *near-owner identifier* and the *separated mode identifier*. When the LTA is in range of a paired owner device, it emits the near-owner identifier which changes every 15 minutes to enable privacy. When the LTA is not in range of the owner device, it continues to evolve the near-owner identifier every 15 minutes but it only broadcasts a single byte of that identifier. For the remaining bytes of output it transmits the separated-mode identifier, which is updated only once per 24-hour period.

While Apple did not initially document the reasoning behind its AirTag design, a recent IETF draft by Apple and Google [34] does explicitly state a motivation for this decision (emphasis added):

> when in a separated state, the accessory SHALL rotate its resolvable and private address every 24 hours. *This duration allows a platform's unwanted tracking algorithms to detect that the same accessory is in proximity for some period of time, when the owner is not in physical proximity.*

Even without this explicit motivation, Apple's decision to continuously update the AirTag near-owner identifier at all times strongly indicates that power consumption is not the motivation for this decision, since total computational effort remains the same in both modes.

$\underline{\mathsf{KeyGen}(\mathsf{cfg}):}$

$k_1 \leftarrow \mathsf{PRF}_1.\mathsf{KeyGen}(1^\lambda)$

$(\_, sk) \leftarrow \mathsf{CCA.KeyGen}(1^\lambda)$

return (sk, k_1, cfg)

$\underline{\mathsf{Beacon}(k_{\mathsf{tag}}, i_{\mathsf{epoch}}, \mathsf{aux}):}$

$(sk, k_1, \mathsf{cfg}) \leftarrow k_{\mathsf{tag}}$

$k \leftarrow k_1$

for $j \in [i_{\mathsf{epoch}}]$:

  $k \leftarrow \mathsf{PRF.Eval}(k, \text{“}\texttt{update}\text{”})$

$(pk, \_) \leftarrow \mathsf{CCA.ReRandomize}(sk; \mathsf{PRF.Eval}(k, \text{“}\texttt{bx}\text{”}))$

return $pk \parallel \mathsf{aux}$

$\underline{\mathsf{GenReport}(B, \mathsf{loc}):}$

$pk \| \mathsf{aux} \leftarrow B$

$ct \leftarrow \mathsf{CCA.Enc}(pk, \mathsf{loc} \| \mathsf{aux})$, $h \leftarrow \mathcal{H}(pk)$

return $h \parallel ct$

$\underline{\mathsf{RetrieveReports}(\mathsf{Owner}(k_{\mathsf{tag}}, i_{\mathsf{epoch}}), \mathsf{SP}(\mathcal{D})):}$

Owner parses k_{tag} as (sk, k_1) and derives key:

  $k \leftarrow k_1$

  for $j \in [i_{\mathsf{epoch}}]$:

    $k \leftarrow \mathsf{PRF.Eval}(k, \text{“}\texttt{update}\text{”})$

  $(pk^*, sk^*) \leftarrow \mathsf{CCA.ReRandomize}(sk; \mathsf{PRF}_1.\mathsf{Eval}(k, \text{“}\texttt{bx}\text{”}))$

Owner sends $\mathcal{H}(pk^*)$ to SP

SP searches $\mathcal{D}$ for values with key $\mathcal{H}(pk^*)$ and adds them to C, which is sent to Owner

Owner decrypts reports in C:

  $\mathsf{out} := []$

  for $c \in C$:

    $(\mathsf{loc}, \mathsf{aux}) \leftarrow \mathsf{CCA.Dec}(sk^*, c)$

    append $(\mathsf{loc}, \mathsf{aux})$ to out

Owner outputs out, SP outputs $\perp$

Figure 13: Apple's FindMy Scheme.

A.1 Cryptography of the FindMy protocol

The Apple FindMy protocol implements an offline finding protocol that achieves perfect unlinkability at the cost of no ability to detect stalkers[8]. We provide the full details here for completeness, based on the reverse-engineering work of Heinrich *et al.* [29, 30]. At a high level, Apple's protocol has the LTA pseudorandomly generate public keys for a CCA secure encryption scheme. These public keys are used by volunteer devices to encrypt location reports. A hash of the public key along with the reports are given to a database controlled by the service provider. The owner - because it shares state with the LTA - can re-derive expected public keys and corresponding secret keys, query on the hash of the public keys, and decrypt location reports. Rather than generate *fresh* public/private key pairs, they choose to use an encryption scheme that admits *key re-randomization.* This is implemented by an extra algorithm CCA.ReRandomize that takes as input a secret key and randomness, and derives a new public, private key pair. The Beacon algorithm uses the output of the pseudorandom function evaluation to produce randomness for calling CCA.ReRandomize on input sk_0. The resulting public key is used to construct the broadcast and the PRF key is updated to a new PRF key through evaluation of the old key by evaluating a KDF on a constant value: this provides forward secrecy in the event that a device is stolen and its keys extracted (and mirrors our discussion in §G.). A non-forward secure variant of the scheme is shown in Figure 13. To achieve forward secrecy, the Beacon algorithm should not use the variable k and instead update k_1 with a single PRF evaluation every time i_{epoch} increments before using it in ReRandomize.

B MDSS

B.1 Proofs of Security Definition Relationships

In this section, we prove the following relationships between our definitions of unlinkability and privacy:

[8]This property can be weakened, as Apple suggests, by just emitting the same identifier for longer. In practice, this has meant allowing parties to be trackable for a time window of 24 hours

- We prove that one-more unlinkability implies unlinkability (Appendix B.1.1).
- We prove that unlinkability is strictly stronger than privacy (Appendix B.1.2).

B.1.1 Unlinkability v.s. One-More Unlinkability

Theorem B.1. *If a $(t_{rec}, t_{priv}, d, \mathsf{max})$-MDSS scheme is statistically (resp., perfectly or computationally) one-more unlinkable then it is also statistically (resp., perfectly or computationally) unlinkable.*

The proof below is for statistical security. The proofs for computational unlinkability and perfect unlinkability are identical, except that $\mathcal{A}$ becomes probabilistic polynomial time in the former and $negl(\lambda) = 0$ in the latter.

Proof. Let $q(\lambda) = \sum_i |I_{i,0}| = \sum_i |I_{i,1}|$. Recall that if the adversary $\mathcal{A}$ queries on $(s_1, n_{1,0}, n_{1,1}, I_{1,0}, I_{1,1}), \ldots (s_z, n_{z,0}, n_{z,1}, I_{z,0}, I_{z,1})$ then the returned shares will be $(\mathsf{Proj}(s_1, n_{1,b}, I_{1,b}), \ldots, \mathsf{Proj}(s_z, n_{z,b}, I_{z,b}))$ where b is the bit chosen by the challenger. From the adversary's perspective the list is simply $sh_1 \ldots sh_{q(\lambda)}$. When it is convenient, we will use this notation in the following proof. We start with the hybrid $\mathcal{H}_0$, which is equivalent to $\mathsf{ULink}^0_{\mathcal{A}}(\lambda)$ (note the interval $[q(\lambda)+1, q(\lambda)]$ is empty). Now consider the following sequence of hybrids $\forall i \in \{1, \ldots, q(\lambda)\}$:

$\mathcal{H}_i$**:** Replace shares at indices $q(\lambda) - i + 1$ through $q(\lambda)$, i.e. $sh_{q(\lambda)-i_1}, \ldots, sh_{q(\lambda)}$, with $\mathsf{Share}(s, 1)$ where s is a random value from $\mathcal{S}$.

Let $\mathsf{ULink}^{\mathcal{H}_i}_{\mathcal{A}}(\lambda)$ be the output of ULink with changes from $\mathcal{H}_i$ against $\mathcal{A}$.

Claim B.1.1. *$\forall i \in \{0, 1, \ldots, q(\lambda) - 1\}$, for all admissable adversaries $\mathcal{A}$, $\exists$ a negligible function $negl$, such that $|\Pr[\mathsf{ULink}^{\mathcal{H}_i}_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{ULink}^{\mathcal{H}_{i+1}}_{\mathcal{A}}(\lambda) = 1]| \leq negl(\lambda)$*

Consider the following $\mathcal{B}$ adversary against $\mathsf{OM\text{-}ULink}^b_{\mathcal{A}}(\lambda)$.
Description of $\mathcal{B}$

For $\ell \in \{q(\lambda) - i + 1, \ldots, q(\lambda)\}$, produce sh_ℓ by running $\mathsf{Share}(s, 1)$ where s is chosen uniformly from $\mathcal{S}$.

Suppose when $b = 0$ $sh_{q(\lambda)-i}$ is associated with the $i_j \in I$ index of share s_j with n_j total shares, produced as $\mathsf{Proj}(s_j, n_j, I)$. Send to the OMUL challenger $s_j, s, n_j, 1, I', 1$ where $s \leftarrow \mathcal{S}$, and $I' \subseteq I$ contains as its last element i_j (i.e. the end of I is dropped). Let $|I'| = t$. Receive $sh'_1, \ldots sh'_{t-1}, sh^*$ and let $sh_{q(\lambda)-i} = sh^*$.

Suppose the k^{th} share of s_j was replaced. For shares $k-1$ through 1 of s_j use the correct share from the set of $sh'_1, \ldots, sh'_{t-1}$

Simulate the rest of the shares $sh_1, \ldots sh_{q(\lambda)-i-t}$ correctly according to $b = 0$. Send to $\mathcal{A}$, $sh_1, \ldots sh_{q(\lambda)}$

Let $\hat{b}$ be the output of $\mathcal{A}$. Send $\hat{b}$ to the challenger

Analysis of $\mathcal{B}$
Since our scheme is one more unlinkable secure, $|\Pr[\mathsf{OM\text{-}ULink}^0_{\mathcal{B}}(\lambda) = 1] - \Pr[\mathsf{OM\text{-}ULink}^1_{\mathcal{B}}(\lambda) = 1]| \leq negl(\lambda)$ for some negligible function $negl$. So we have directly

$$\begin{aligned}
|\Pr[\mathsf{OM\text{-}ULink}^0_{\mathcal{B}}(\lambda) = 1] - \Pr[\mathsf{OM\text{-}ULink}^1_{\mathcal{B}}(\lambda) = 1]| &= |\Pr[\mathcal{A} \text{ outputs 1 in } \mathcal{H}_i] - \Pr[\mathcal{A} \text{ outputs 1 in } \mathcal{H}_{i+1}]| \\
&= |\Pr[\mathsf{ULink}^{\mathcal{H}_i}_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{ULink}^{\mathcal{H}_{i+1}}_{\mathcal{A}}(\lambda) = 1]| \\
&\leq negl(\lambda)
\end{aligned}$$

Now consider another sequence of hybrids $\mathcal{H}^*_1 \ldots \mathcal{H}^*_{q(\lambda)}$.

$\mathcal{H}_j^*$**:** Replace shares $sh_1, \ldots, sh_j$ with the shares associated with $b = 1$. For shares $sh_{j+1} \ldots, sh_{q(\lambda)}$, send a share of a random secret, i.e $\mathsf{Share}(s, 1)$ for $s \leftarrow \mathcal{S}$.

Note that $\mathcal{H}_0^* = \mathcal{H}_{q(\lambda)}$. As before, let $\mathsf{ULink}_{\mathcal{A}}^{\mathcal{H}_j^*}(\lambda)$ be the output of ULink with modifications from $\mathcal{H}_j^*$ against $\mathcal{A}$.

Claim B.1.2. *$\forall j \in \{0, \ldots q(\lambda) - 1\}$, for all adversaries $\mathcal{A}$, $\exists$ a negligible function $negl(\lambda)$ s.t.* $|\Pr[\mathsf{ULink}_{\mathcal{A}}^{\mathcal{H}_j^*}(\lambda) = 1] - \Pr[\mathsf{ULink}_{\mathcal{A}}^{\mathcal{H}_{j+1}^*}(\lambda) = 1]| \leq negl(\lambda)$

Consider a new adversary $\mathcal{B}^*$ against $\mathsf{OM\text{-}ULink}$.

Description of $\mathcal{B}^*$

For shares $sh_{j+1}, \ldots, sh_{q(\lambda)}$, use $\mathsf{Share}(s, 1)$ where s is chosen uniformly from $\mathcal{S}$.

Suppose when $b = 1$, sh_j is associated with the $i_j \in I$ index of share s_ℓ with n_ℓ total shares. Send to the OMUL challenger s_ℓ, s, n_ℓ, I' where $I' \subseteq I$ with I' containing as its last element i_j and $s \leftarrow \mathcal{S}$. Let $|I'| = t$. Receive $sh'_1 \ldots sh'_{t-1}, sh^*$. Set $sh_j = sh^*$

For all the remaining shares that should come from s_ℓ use the correct shares from $sh'_1 \ldots sh'_{t-1}$. Simulate the rest of the shares.

Send to $\mathcal{A}$, $sh_1, \ldots, sh_{q(\lambda)}$

If $\mathcal{A}$ outputs $\hat{b}$ output $\hat{b}$

Analysis of $\mathcal{B}^*$

Again, since our scheme is one-more unlinkable, we know that $|\Pr[\mathsf{OM\text{-}ULink}_{\mathcal{B}^*}^0(\lambda) = 1] - \Pr[\mathsf{OM\text{-}ULink}_{\mathcal{B}^*}^1(\lambda) = 1]| \leq negl(\lambda)$ for some negligible function $negl(\lambda)$.

$$\begin{aligned}
|\Pr[\mathsf{OM\text{-}ULink}_{\mathcal{B}}^0(\lambda) = 1] - \Pr[\mathsf{OM\text{-}ULink}_{\mathcal{B}}^1(\lambda) = 1]| &= |\Pr[\mathcal{A} \text{ outputs 1 in } \mathcal{H}_{i+1}^*] - \Pr[\mathcal{A} \text{ outputs 1 in } \mathcal{H}_i^*]| \\
&= |\Pr[\mathsf{ULink}_{\mathcal{A}}^{\mathcal{H}_{i+1}^*}(\lambda) = 1] - \Pr[\mathsf{ULink}_{\mathcal{A}}^{\mathcal{H}_i^*}(\lambda) = 1]| \\
&= |\Pr[\mathsf{ULink}_{\mathcal{A}}^{\mathcal{H}_i^*}(\lambda) = 1] - \Pr[\mathsf{ULink}_{\mathcal{A}}^{\mathcal{H}_{i+1}^*}(\lambda) = 1]| \\
&\leq negl(\lambda)
\end{aligned}$$

Finally, note that $\mathcal{H}_{q(\lambda)}^* = \mathsf{ULink}_{\mathcal{A}}^1(\lambda)$. This completes our proof, since $q(\lambda) \in \mathsf{poly}(\lambda)$ by assumption, and thus the sequence of hybrids is polynomial in length. □

B.1.2 Unlinkability v.s. Privacy

We start by recalling the standard notion of privacy for secret sharing.

Privacy. A threshold secret sharing scheme is said to be private if any set of at most t_{priv} shares of a secret s, hides s. The threshold t_{priv} is the privacy parameter of the scheme.

Definition B.1 (Privacy)**.** *We say that a $(t_{rec}, t_{priv}, d, \mathsf{max})$-MDSS with secret space $\mathcal{S}$ is ϵ-*private *if for any $s_0, s_1 \in \mathcal{S}$ and any $n \in \mathbb{Z}$ such that $n \leq t_{priv}$, for all distinguishers D:*

$$|\Pr[S \leftarrow \mathsf{Share}(s_0, n) : D(S) = 1] - \Pr[S \leftarrow \mathsf{Share}(s_1, n) : D(S) = 1]| \leq \epsilon$$

If $\epsilon = 0$, we say the scheme is perfectly private. If ϵ is a negligible function in λ, we say that the scheme is statistically private. If the definition only holds against distinguisher D running in time polynomial in λ, and for each D there exists a negligible function $\epsilon(\lambda)$ such that the bound holds, then the scheme is said to be computationally private.

Construction I

$\underline{\mathsf{Share}(s, n)}$:
Sample a polynomial p of degree t_{priv} where $s = p(0)$
Sample n field elements $x_1, \ldots, x_n \xleftarrow{\$} \mathbb{F}$
return $\{(x_i, p(x_i))\}_{i \in [n]}$

$\underline{\mathsf{Reconstruct}(\{sh_1, \ldots, sh_w\})}$:
$p_1, \ldots, p_m \leftarrow \mathsf{GSDecode}(t_{priv}, t_{rec}, \{sh_1, \ldots, sh_w\})$
return $\{p_1(0), \ldots, p_m(0)\}$

Figure 14: MDSS Construction I

Unlinkability v.s. Privacy. Unlinkability is a strictly stronger notion than privacy. We establish this in two steps: we first demonstrate that unlinkability implies privacy.

Theorem B.2. *If a $(t_{rec}, t_{priv}, d, \mathsf{max})$-MDSS scheme is statistically (resp., perfectly or computationally) unlinkable, then it is also statistically (resp., perfectly or computationally) private.*

Proof. Consider the unlinkability game $\mathsf{ULink}^b_{\mathcal{A}}(\lambda)$ where the adversary $\mathcal{A}$ chooses two secrets s_0, s_1, some integer n, and subset $I \in [n], |I| = t_{priv}$ with the following tuples $(s_0, n_{0,0} = n, n_{0,1} = 0, I_{0,0} = I, I_{0,1} = \emptyset)$, $(s_1, n_{1,0} = 0, n_{1,1} = n, I_{1,0} = \emptyset, I_{1,1} = I)$. When $b = 0$, $\mathcal{A}$ receives t_{priv} shares of s_0; otherwise, it receives t_{priv} shares of s_1. This corresponds exactly to the privacy game. □

Next, we show that privacy does *not* imply unlinkability. More specifically, there exist threshold secret sharing schemes that are private but not unlinkable.

Theorem B.3. *There exists a secret sharing scheme that is perfectly private but not computationally unlinkable.*

Proof. Consider the following modification to a perfectly private secret sharing scheme: to share a secret s, first sample a random value r and then append r to each share of s. Clearly, this modified scheme is still perfectly private. However, a polynomial-time adversary can trivially link the shares of s by comparing their suffixes. □

B.2 Construction I

In this section we give a formal description of the MDSS construction described in 3.2.1, and state Theorems regarding its correctness and security. The full description of the construction can be seen in Figure 14.

Theorem B.4 (Construction I Unlinkability)**.** *For all finite fields $\mathbb{F}$ such that $|\mathbb{F}| \approx \lambda^{\omega(1)}$, Construction 14 satisfies ϵ-unlinkability for a negligibly small $\epsilon(\lambda)$.*

Theorem B.5 (Construction I Correctness)**.** *For all sets of parameters satisfying $t_{rec} > \sqrt{\mathsf{max} \cdot t_{priv}}$ and finite fields $\mathbb{F}$ such that $|\mathbb{F}| \approx \lambda^{\omega(1)}$, Construction 14 satisfies ϵ-correctness for a negligibly small $\epsilon(\lambda)$.*

The proofs for these Theorems can be found in Appendix B.3.

B.3 Proofs for MDSS Constructions

In this section we give proofs of unlinkability and MD-correctness for Constructions 14 and 10.

B.3.1 Proof of Theorem 3.1

For an unlinkability adversary $\mathcal{A}$, let U be the event that the share sh_t the adversary playing the $\mathsf{OM\text{-}ULink}^b_{\mathcal{A}}(\lambda, t)$ game receives has a unique x-coordinate (i.e. distinct from the x-coordinate of every other received share). We now prove Theorem 3.1.

Claim B.5.1. *For all $t \leq t_{priv}$,*

$$\Pr[\mathsf{OM\text{-}ULink}^0_{\mathcal{A}}(\lambda, t) = 1 | \mathsf{U}] = \Pr[\mathsf{OM\text{-}ULink}^1_{\mathcal{A}}(\lambda, t) = 1 | \mathsf{U}]$$

Proof. As the distribution of $\{sh_0, \ldots, sh_{t-1}\}$ is identical in both games, we focus our analysis only on sh_t. When sh_t has a unique x-coordinate and $b = 0$, $\mathcal{A}$ sees the t'th point from c degree-t_{priv} polynomials, which is equivalent to a uniformly random value of $\mathbb{F}^{c+1}$. When $b = 1$, $\mathcal{A}$ always sees a uniform random value of $\mathbb{F}^{c+1}$. Therefore, since the distribution passed to $\mathcal{A}$ is identical in both cases, its behavior in the games must be identical. □

Claim B.5.2. $\Pr[!\mathsf{U}] \leq \frac{t_{priv}}{|\mathbb{F}|}$

Proof. $\Pr[!\mathsf{U}]$ is equivalent to the probability that the x-coordinate of sh_t collides with the x-coordinate of a previously received share. As there are at most t_{priv} previously received shares, and each x-coordinate is sampled uniformly, we have $\Pr[!\mathsf{U}] \leq \frac{t_{priv}}{|\mathbb{F}|}$. □

Claim B.5.3. *For all $t \leq t_{priv}$,*

$$|\Pr[\mathsf{OM\text{-}ULink}^0_{\mathcal{A}}(\lambda, t) = 1] - \Pr[\mathsf{OM\text{-}ULink}^1_{\mathcal{A}}(\lambda, t) = 1]| \leq negl(\lambda)$$

Proof. Let A be the event that $\mathsf{OM\text{-}ULink}^0_{\mathcal{A}}(\lambda, t) = 1$, and B be the event that $\mathsf{OM\text{-}ULink}^1_{\mathcal{A}}(\lambda, t) = 1$. Then,

$$\begin{aligned}
|\Pr[A] - \Pr[B]| =& \big| \Pr[A|\mathsf{U}]\Pr[\mathsf{U}] + \Pr[A|!\mathsf{U}]\Pr[!\mathsf{U}] - \Pr[B|\mathsf{U}]\Pr[\mathsf{U}] \\
&\quad - \Pr[B|!\mathsf{U}]\Pr[!\mathsf{U}] \big| \\
=& |\Pr[A|!\mathsf{U}]\Pr[!\mathsf{U}] - \Pr[B|!\mathsf{U}]\Pr[!\mathsf{U}]| && \text{(Claim B.5.1)} \\
\leq& \Pr[A|!\mathsf{U}]\Pr[!\mathsf{U}] + \Pr[B|!\mathsf{U}]\Pr[!\mathsf{U}] \\
\leq& 2\Pr[!\mathsf{U}] \\
=& 2\frac{t_{priv}}{|\mathbb{F}|} && \text{(Claim B.3.1)} \\
=& negl(\lambda) && (|\mathbb{F}| \approx \lambda^{\omega(1)})
\end{aligned}$$

The Theorem then follows from Claim B.5.3 and Theorem B.1. □

B.3.2 Proof of Theorem 3.2

Proof. Consider a field $\mathbb{F}$ and a set $\{(s_i, n_i, \mathcal{I}_i)\}_{i \in h}$ as defined in Definition 3.6. For Construction 10 and a set S, let $A(S)$ be the event that $\mathsf{Reconstruct}(S) \neq \left\{ s_i \text{ s.t. } |\mathcal{I}_i| \geq \frac{\mathsf{max} + c \cdot t_{priv}}{c+1} \right\}$. Then, we have:

$$\Pr\left[\mathsf{Reconstruct}(S) \neq \{s_i \text{ s.t. } |\mathcal{I}_i| \geq t_{rec}\} \;\middle|\; S := \bigcup_{i=1}^{h} \mathsf{Proj}(s_i, n_i, \mathcal{I}_i) \right] \tag{1}$$

$$= \Pr\left[A(S) \;\middle|\; S := \bigcup_{i=1}^{h} \mathsf{Proj}(s_i, n_i, \mathcal{I}_i) \right] \tag{2}$$

$$= \Pr\left[A(S) \;\middle|\; \begin{smallmatrix} \forall i \in [h],\ P_i \leftarrow \zeta(\mathbb{F}, t_{priv}, c, s_i) \\ \forall i \in [h],\ S_i \leftarrow D^{\mathbb{F}}_{|\mathcal{I}_i|, P_i} \\ S := \bigcup_{i \in [h]} S_i \end{smallmatrix} \right] \tag{3}$$

Where (2) follows from the conditions of Theorem 3.2, and (3) by construction.

For notational convenience, let $S \leftarrow B$ denote sampling S from the distribution described in the conditions above. Then we have:

$$
\begin{aligned}
&= \Pr[A(S) \mid S \leftarrow B] && (4)\\
&= \Pr[A(S) \mid S \leftarrow B \wedge \mathsf{Unique}(S)] \Pr[\mathsf{Unique}(S)] \\
&\quad + \Pr[A(S) \mid S \leftarrow B \wedge !\mathsf{Unique}(S)] \Pr[!\mathsf{Unique}(S)] && (5)\\
&= negl(\log |\mathbb{F}|) \cdot \Pr[\mathsf{Unique}(S)] \\
&\quad + \Pr[A(S) \mid S \leftarrow B \wedge !\mathsf{Unique}(S)] \Pr[!\mathsf{Unique}(S)] && (6)\\
&= negl(\lambda) \cdot \Pr[\mathsf{Unique}(S)] \\
&\quad + \Pr[A(S) \mid S \leftarrow B \wedge !\mathsf{Unique}(S)] \Pr[!\mathsf{Unique}(S)] && (7)\\
&= negl(\lambda) + \Pr[A(S) \mid S \leftarrow B \wedge !\mathsf{Unique}(S)] \Pr[!\mathsf{Unique}(S)] && (8)\\
&\leq negl(\lambda) + \Pr[A(S) \mid S \leftarrow B \wedge !\mathsf{Unique}(S)] \cdot \frac{\mathsf{max}(\mathsf{max}-1)}{2|\mathbb{F}|} && (9)\\
&= negl(\lambda) + negl(\lambda) && (10)\\
&= negl(\lambda) && (11)
\end{aligned}
$$

Where (6) follows from Heuristic Assumption 1, (7) from the conditions of Theorem 3.2, (9) from the birthday bound, and (10) from the fact that max is constant.

□

B.3.3 Proof of Theorem B.4

Proof. As the Share algorithm of Construction 14 is identical to that of Construction 10 for $c = 1$, this follows as a direct corollary of Theorem 3.1. □

B.3.4 Proof of Theorem B.5

To prove the Theorem B.5, we first need the formal guarantee of $\mathsf{GSDecode}$. The following theorem is due to Guruswami and Sudan [27]:

Theorem B.6 (Guruswami-Sudan Decoder)**.** *Let $\mathbb{F}$ be a field. Let k, t, n be positive integers such that $t > \sqrt{kn}$. Then there exists an algorithm $\mathsf{GSDecode}(k, t, \{(x_i, y_i)\}_{i\in[n]})$ where for all i: $x_i, y_i \in \mathbb{F}$, that outputs all univariate polynomials p of degree at most k such that $y_i = p(x_i)$ for at least t values of $i \in [n]$.*

While $\mathsf{GSDecode}$ will output all polynomials that agree on at least t_{rec} points, that is not actually what we want. To achieve correctness, we need $\mathsf{Reconstruct}$ to output only those secrets *shared by the sufficient dealers*. In other words, it should not be the case that there are t_{rec} points that lie on some new, separate polynomial. Therefore, to begin our proof, we must bound away this possibility.

We first define a predicate for this "bad" case over a specific set of t points:

Definition B.2. *For a set of points $\{(x_1, y_1), \ldots, (x_t, y_t)\}$ and a set of degree k polynomials $\{p_1, \ldots, p_d\}$, define $\mathsf{Bad}(k, \{(x_i, y_i)\}_{i\in[t]}, \{p_j\}_{j\in[h]})$ as follows:*

- *If there exists a polynomial p' such that $\forall i \in [t]$, $p'(x_i) = y_i$ and $p' \notin \{p_1, \ldots, p_h\}$, output 1.*
- *Else, output 0.*

Next, we give a lemma that says that the probability of Bad occurring is maximized when there is no set of "allowable" polynomials.

Lemma B.6.1. *For any set of polynomials $P = \{p_1, \ldots, p_h\}$:*

$$
\begin{aligned}
&\Pr[\mathsf{Bad}(k, \{(x_i, y_i)\}_{i\in[t]}, P) \mid \forall i \in [t],\ (x_i, y_i) \xleftarrow{\$} \mathbb{F}^2] \\
&\leq \Pr[\mathsf{Bad}(k, \{(x_i, y_i)\}_{i\in[t]}, \emptyset) \mid \forall i \in [t],\ (x_i, y_i) \xleftarrow{\$} \mathbb{F}^2]
\end{aligned}
$$

Proof. For any set of points $\{(x_i, y_i)\}_{i\in[t]}$, if $\mathsf{Bad}(k, \{(x_i, y_i)\}_{i\in[t]}, P)$ is satisfied, then $\mathsf{Bad}(k, \{(x_i, y_i)\}_{i\in[t]}, \emptyset)$ must clearly also be satisfied. □

We can now begin bounding the probability that this bad case occurs.

Claim B.6.1.

$$\Pr[\mathsf{Bad}(k, \{(x_i, y_i)\}_{i\in[t]}, \emptyset) \mid \forall i \in [t],\ (x_i, y_i) \xleftarrow{\$} \mathbb{F}^2] = \left(\frac{1}{|\mathbb{F}|}\right)^{(t-(k+1))}$$

Proof. For a set of t points, $(k+1)$ will uniquely determine a degree-k polynomial. Then, the remaining $(t-(k+1))$ points must all agree with that fixed polynomial, and each agrees with probability $\frac{1}{|\mathbb{F}|}$. □

Claim B.6.2. *Let $\mathbb{F}$ be a field and let $t, k, \mathsf{max}, m_1, \ldots, m_h$ be integers where $\mathsf{max} = \sum_{i=1}^{h}$, and let $s_1, \ldots, s_h \in \mathbb{F}$ be field elements. Let $\zeta(\mathbb{F}, k, c, s)$ be defined as in Heuristic Assumption 1. Then, for all $\mathcal{I} \subseteq \mathsf{max}$ where $|\mathcal{I}| = t$:*

$$\Pr\left[\mathsf{Bad}(k, S_{\mathcal{I}}, \{p_i\}_{i\in[h]}) \;\middle|\; \begin{smallmatrix}\forall i\in[h],\ p_i \leftarrow \zeta(\mathbb{F},k,1,s_i)\\ \forall i\in[h],\ S_i \leftarrow D^{\mathbb{F}}_{m_i,p_i}\\ S := \bigcup_{i\in[h]} S_i\end{smallmatrix}\right] \leq \left(\frac{1}{|\mathbb{F}|}\right)^{(t-(k+1))}$$

Where $S_{\mathcal{I}}$ indicates the set of elements of the ordered set S at the positions contained in $\mathcal{I}$.

Proof. For a set of t points $S_{\mathcal{I}}$ there are two cases: either $S_{\mathcal{I}}$ contains $k+1$ points on one of the initial polynomials p_i, or it does not. In the first case, clearly $\Pr[\mathsf{Bad}\ldots] = 0 \leq \left(\frac{1}{|\mathbb{F}|}\right)^{(t-(k+1))}$. In the second case, all the points in $S_{\mathcal{I}}$ are uniformly distributed over $\mathbb{F}$, and therefore $\Pr[\mathsf{Bad}\ldots] \leq \left(\frac{1}{|\mathbb{F}|}\right)^{(t-(k+1))}$ from Claim B.6.1 and Lemma B.6.1. □

We now define a couple pieces of notational shorthand. For a set S, let $\mathsf{Subsets}(S, t)$ denote the set of all subsets of S of size t. For integers k, t, a set of points S, and a set of polynomials $\{p_1, \ldots, p_h\}$, let $\mathsf{EB}(k, t, S, \{p_i\}_{i\in[h]})$ denote the event that $\exists S' \in \mathsf{Subsets}(S, t)$ *s.t.* $\mathsf{Bad}(k, S', \{p_i\}_{i\in[h]}) = \mathsf{True}$.

Claim B.6.3. *Let $\mathbb{F}$ be a field and let $t, k, \mathsf{max}, m_1, \ldots, m_h$ be integers where $\mathsf{max} = \sum_{i=1}^{h}$, and let $s_1, \ldots, s_h \in \mathbb{F}$ be field elements. Let $\zeta(\mathbb{F}, k, c, s)$ be defined as in Heuristic Assumption 1. Then:*

$$\Pr\left[\mathsf{EB}(k, t, S, \{p_i\}_{i\in h}) \;\middle|\; \begin{smallmatrix}\forall i\in[h],\ p_i \leftarrow \zeta(\mathbb{F},k,1,s_i)\\ \forall i\in[h],\ S_i \leftarrow D^{\mathbb{F}}_{m_i,p_i}\\ S := \bigcup_{i\in[h]} S_i\end{smallmatrix}\right] \leq \binom{\mathsf{max}}{t} \cdot \left(\frac{1}{|\mathbb{F}|}\right)^{(t-(k+1))}$$

Proof. By Claim B.6.2 we have that the probability that Bad is satisfied for any size t subset of S is at most $\left(\frac{1}{|\mathbb{F}|}\right)^{(t-(k+1))}$. As there are $\binom{\mathsf{max}}{t}$ such subsets, the Claim follows from the union bound. □

We can now prove Theorem B.5.

Proof. For all sets $\{(s_i, n_i, \mathcal{I}_i)\}_{i\in[h]}$ as described in Definition 3.6, for a set S let $A(S)$ be the event that $\mathsf{GSDecode}(S) \neq \{s_i \text{ s.t. } |\mathcal{I}_i| \geq t_{rec}\}$, and let $m_i = |\mathcal{I}_i|$ for all i. Then, we have:

$$\Pr\left[\mathsf{Reconstruct}(S) \neq \{s_i \text{ s.t. } |\mathcal{I}_i| \geq t_{rec}\} \;\middle|\; S := \bigcup_{i=1}^{h} \mathsf{Proj}(s_i, n_i, \mathcal{I}_i)\right] \tag{12}$$

$$= \Pr\left[A \;\middle|\; S := \bigcup_{i=1}^{h} \mathsf{Proj}(s_i, n_i, \mathcal{I}_i)\right] \tag{13}$$

$$= \Pr\left[A \;\middle|\; \begin{smallmatrix}\forall i\in[h],\ p_i \leftarrow \zeta(\mathbb{F},t_{priv},1,s_i)\\ \forall i\in[h],\ S_i \leftarrow D^{\mathbb{F}}_{m_i,p_i}\\ S := \bigcup_{i\in[h]} S_i\end{smallmatrix}\right] \tag{14}$$

MDSS Small Field Construction

$\underline{\mathsf{Share}(s, n)}$:

Sample c polynomials $p_1, \ldots, p_c$ each of degree t_{priv}, where $s = p_1(0)||...||p_c(0)$

Sample n field elements $x_1, \ldots, x_n \xleftarrow{\$} \mathbb{F}$

For $i \in [n]$, if $x_i \notin \{x_1, \ldots, x_{i-1}\}$, set $sh_i := (x_i, p_1(x_i), \ldots, p_c(x_i))$, else, sample $r_1, \ldots, r_c \xleftarrow{\$} \mathbb{F}$ and set $sh_i := (x_i, r_1, \ldots, r_c)$

return $\{sh_1, \ldots, sh_t\}$

$\underline{\mathsf{Reconstruct}(\{sh_1, \ldots, sh_w\})}$:

return $\mathsf{CH}^*\text{-}\mathsf{MDSS}(\{sh_1, \ldots, sh_w\})$

Figure 15: Our MDSS Construction for small fields

$$= \Pr\left[A \left| \begin{array}{c} \forall i \in [h],\, p_i \leftarrow \zeta(\mathbb{F}, t_{priv}, 1, s_i) \\ \forall i \in [h],\, S_i \leftarrow D^{\mathbb{F}}_{m_i, p_i} \\ S := \bigcup_{i \in [h]} S_i \\ \mathsf{EB}(t_{priv}, t_{rec}, S, \{p_i\}_{i \in [h]}) \end{array}\right.\right] \Pr[\mathsf{EB}(t_{priv}, t_{rec}, S, \{p_i\}_{i \in [h]})]$$

$$+ \Pr\left[A \left| \begin{array}{c} \forall i \in [h],\, p_i \leftarrow \zeta(\mathbb{F}, t_{priv}, 1, s_i) \\ \forall i \in [h],\, S_i \leftarrow D^{\mathbb{F}}_{m_i, p_i} \\ S := \bigcup_{i \in [h]} S_i \\ !\mathsf{EB}(t_{priv}, t_{rec}, S, \{p_i\}_{i \in [h]}) \end{array}\right.\right] \Pr[!\mathsf{EB}(t_{priv}, t_{rec}, S, \{p_i\}_{i \in [h]})] \tag{15}$$

$$= \Pr\left[A \left| \begin{array}{c} \forall i \in [h],\, p_i \leftarrow \zeta(\mathbb{F}, t_{priv}, 1, s_i) \\ \forall i \in [h],\, S_i \leftarrow D^{\mathbb{F}}_{m_i, p_i} \\ S := \bigcup_{i \in [h]} S_i \\ \mathsf{EB}(t_{priv}, t_{rec}, S, \{p_i\}_{i \in [h]}) \end{array}\right.\right] \Pr[\mathsf{EB}(t_{priv}, t_{rec}, S, \{p_i\}_{i \in [h]})] \tag{16}$$

$$\leq \Pr\left[A \left| \begin{array}{c} \forall i \in [h],\, p_i \leftarrow \zeta(\mathbb{F}, t_{priv}, 1, s_i) \\ \forall i \in [h],\, S_i \leftarrow D^{\mathbb{F}}_{m_i, p_i} \\ S := \bigcup_{i \in [h]} S_i \\ \mathsf{EB}(t_{priv}, t_{rec}, S, \{p_i\}_{i \in [h]}) \end{array}\right.\right] \cdot \binom{\mathsf{max}}{t_{rec}} \cdot \left(\frac{1}{|\mathbb{F}|}\right)^{(t_{rec} - (t_{priv}+1))} \tag{17}$$

$$= negl(\lambda) \tag{18}$$

Where (13) and (14) both follow from construction, (16) follows from Theorem B.6, 17 follows from Claim (B.6.3), and (18) follows from the fact that $|\mathbb{F}| \approx \lambda^{\omega(1)}$ and that max and t_{rec} are constants. □

B.4 MDSS For Small Fields

In this Section we formalize the small field construction discussed in Section 3.2.3. The full construction can be seen in Figure 15.

Theorem B.7. *For all $c \geq 1$ and all finite fields $\mathbb{F}$, Construction 15 satisfies ϵ-unlinkability for $\epsilon = 0$.*

Proof. As the distribution of $\{sh_0, \ldots, sh_{t-1}\}$ is identical in both games, we focus our analysis only on sh_t. In the $\mathsf{OM\text{-}ULink}^0_{\mathcal{A}}(\lambda, t)$ game, if the x-coordinate of sh_t collides with one of the x-coordinates of $\{sh_0, \ldots, sh_{t-1}\}$, then it is a uniformly random element of $\mathbb{F}^{c+1}$. If the x-coordinate does *not* collide, then sh_t is the t_{priv}th point on a t_{priv}-degree polynomial, and is therefore identically distributed to a uniformly random element of $\mathbb{F}^{c+1}$. In the $\mathsf{OM\text{-}ULink}^1_{\mathcal{A}}(\lambda, t)$ game, sh_t is guaranteed to be a uniformly random element of $\mathbb{F}^{c+1}$. Therefore, the distribution in both games is identical, and the advantage of all adversaries must be 0. The Theorem then follows from $B.1$. □

Theorem B.8. *Under Heuristic Assumption 1, for all sets of parameters satisfying $t_{rec} \geq \frac{1}{c+1}(\mathsf{max} + (c \cdot t_{priv}))$ and finite fields $\mathbb{F}$ such that $|\mathbb{F}| \approx \lambda^{\Omega(1)}$, construction 10 satisfies* ϵ-correctness *for $\epsilon \approx 1/\lambda^{\Omega(1)}$.*

Proof. The proof proceeds identically to the proof of Theorem 3.2 up to (5). We proceed from there:

$$\begin{aligned}
&\Pr[A(S) \mid S \leftarrow B \wedge \mathsf{Unique}(S)] \Pr[\mathsf{Unique}(S)] \\
&+ \Pr[A(S) \mid S \leftarrow B \wedge !\mathsf{Unique}(S)] \Pr[!\mathsf{Unique}(S)] && (19)\\
=&negl(\log|\mathbb{F}|) \cdot \Pr[\mathsf{Unique}(S)] \\
&+ \Pr[A(S) \mid S \leftarrow B \wedge !\mathsf{Unique}(S)] \Pr[!\mathsf{Unique}(S)] && (20)\\
=&negl(\log \lambda^{\Omega(1)}) \cdot \Pr[\mathsf{Unique}(S)] \\
&+ \Pr[A(S) \mid S \leftarrow B \wedge !\mathsf{Unique}(S)] \Pr[!\mathsf{Unique}(S)] && (21)\\
=&\frac{1}{\lambda^{\Omega(1)}} + \Pr[A(S) \mid S \leftarrow B \wedge !\mathsf{Unique}(S)] \Pr[!\mathsf{Unique}(S)] && (22)\\
\leq&\frac{1}{\lambda^{\Omega(1)}} + \Pr[A(S) \mid S \leftarrow B \wedge !\mathsf{Unique}(S)] \cdot \frac{\mathsf{max}(\mathsf{max}-1)}{2|\mathbb{F}|} && (23)\\
=&\frac{1}{\lambda^{\Omega(1)}} + \frac{1}{\lambda^{\Omega(1)}} && (24)\\
=&\frac{1}{\lambda^{\Omega(1)}} && (25)
\end{aligned}$$

Where (20) follows from Heuristic Assumption 1, and the observation that if $\mathsf{Unique}(S)$ is satisfied, the Share algorithm behaves identically to the Share algorithm on Construction 10, i.e. does not insert any "noise" shares. Next, (21) follows from the conditions of Theorem B.8, (23) from the birthday bound, and (24) as max is constant. □

B.5 Collision-Aware PRF

Realizing our constructions over small fields poses an implementation challenge: to protect the unlinkability of a given dealer, the evaluation point x for each secret share must be sampled uniformly. At the same time, our secret sharing protocols require that a dealer must emit noise shares when evaluating a secret share on duplicate coordinates. The naive approach to solving this problem requires each dealer to keep a list of past evaluation points; unfortunately this may force the dealer to retain a substantial amount of state.

Here we propose an alternative approach: rather than recording past evaluation points, each dealer will sample the evaluation point *pseudorandomly* using a specialized PRF construction that we refer to as a *collision-aware PRF*. The novel feature of this PRF construction is that it signals the presence of a collision. More concretely, when presented with an input i and some compact state computed from past queries, this function produces both a pseudorandom element in $\mathbb{F}$ as well as a separate *collision-detection signal*: this signal will equal 1 iff there exists some input $j < i$ such that $F(i) = F(j)$. In practice, a caller can evaluate this function sequentially on inputs $\{0, 1, 2, \dots\}$ to produce a sequence of evaluation points, and can respond to collision signals as required by the secret sharing protocol.

Formalizing CA-PRFs. We now provide definitions of the CA-PRF notion.

Definition B.3 (CA-PRF)**.** *Let $\mathcal{D}, \mathcal{R}$ be two sets, where $\mathcal{D}$ is an ordered set of size polynomial in the key length. A collision-aware PRF (CA-PRF) family is a function $F : \mathcal{K} \times \mathcal{D} \times \mathbb{Z}^+ \to \mathcal{R} \times \{0, 1\}$*

We will consider the case where $\mathcal{D} = \mathcal{R}$, and notationally we will assume that elements in $\mathcal{D}$ can be represented by an integer position in the set. We will use the notation $F_K(i, M_i) \to (x, b)$ to represent a query specifying the i^{th} element of $\mathcal{D}$ and a non-negative integer *count* M_i. This query produces an element $x \in \mathcal{D}$ as well as a bit $b \in \{0, 1\}$ that indicates whether there is a collision with some "earlier" input to the function.

Properties. We define a *well-formed* query as one that takes as input a pair (i, M_i) where M_i represents the number of *unique* x values produced by queries on earlier elements of the domain. More precisely, we

define $M_0 = 0$, and for $0 < i < |\mathcal{D}|$, we inductively define M_i to be the total number of (unique) elements in the set $\{x_j\}_{j \in [0,i-1]}$ where $(x_j, b_j) \leftarrow F_K(j, M_j)$.

A CA-PRF must possess two main properties: *collision-correctness* and *pseudorandomness*. We describe these below:

Collision-correctness. This property holds that for all keys K and input tuples where (i, M_i) are well-formed, the query $(x_i, b_i) \leftarrow F_K(i, M_i)$ will output $b_i = 1$ iff there exist an integer $0 \leq j < i$ such that well-formed $(x_j, b_j) \leftarrow F_K(j, M_j)$ has $x_i = x_j$.

(Non-adaptive) pseudorandomness. This is identical to the standard pseudorandomness notion for PRFs, with two caveats: (1) we consider only adversaries who evaluate the oracle non-adaptively on a fixed polynomial set of *well-formed* queries $0, \ldots, q-1$, and (2) pseudorandomness applies only to the output string x, and not to the collision bit b.

B.5.1 Construction

We now propose a concrete CA-PRF with domain and range $\{0, \ldots, p-1\}$ where p is prime, and p is polynomial in the key length. Our construction is built from a standard PRF $\bar{F}$ and some small domain PRP P. We note that such permutations can be constructed from a standard PRF via the techniques of [10, 6].

Let $\bar{F}$ be a PRF family with domain and range $\{0,1\}^\lambda$, and let P be a small-domain PRP over the domain $\{0, \ldots, p-1\}$. Our construction is described as follows:

1. **Key generation.** To generate a key, sample key k_P for the PRP P and key $k_{\bar{F}}$ for the PRF $\bar{F}$. Output $K = (k_P, k_{\bar{F}})$.
2. **Evaluation.** On input a key K, an element $i \in [0, p-1]$ and an integer count $M_i \geq 0$:
   (a) Parse K as $(k_P, k_{\bar{F}})$.
   (b) Compute $(j', b) \leftarrow \mathsf{SampleProb}(k_{\bar{F}}, i, M_i, p)$.
   (c) If $b = 0$, output $(P_{k_P}(M_i), 0)$.
   (d) Otherwise if $b = 1$, output $(P_{k_P}(j'), 1)$.
3. $\mathsf{SampleProb}(k, i, M_i, p) \rightarrow (j', b)$. If $i = 0$ this subroutine returns $(0, 0)$. Otherwise it performs the following steps:
   (a) Compute a sequence of pseudorandom coins $r \leftarrow \left(F'_{k_{F'}}(i \| M_i \| p \| 0) \| \ldots \| F'_{k_{F'}}(i \| M_i \| p \| \ell)\right)$, for some ℓ large enough that $|r|$ is sufficient to perform the remaining steps.
   (b) Use the coins r to sample a bit b such that $b = 1$ with probability $\frac{M_i}{p}$ (over the given coins.)
   (c) If $b = 0$ set $j' \leftarrow 0$, and if $b = 1$ use the remaining coins from r to uniformly sample an integer j' in the range $[0, M_i - 1]$.
   (d) Return (j', b).

Correctness. We observe that collision-correctness holds trivially following query $i = 0$. Let $i, 0 < i < p$ be the smallest query where the conditions of collision-correctness *do not* hold. Written explicitly, this implies one of two possibilities: either (1) $b_i = 0$ and yet $x_i = x_j$ where x_j is the output of some previous query $0 \leq j < i$, or (2) $b_i = 1$ and yet $x_i \neq x_j$ for all $0 \leq j < i$. Since i is the smallest value to violate collision-correctness, we can assume that collision-correctness holds for every query $0 \leq i' < i$. We consider these two cases separately and show that either case implies a contradiction.

In case (1) then there must exist some query $j < i$ such that $x_i = x_j$ and $b_j = 0$. (If there exists such a j where $b_j = 1$ then, under our stipulation that all queries $j < i$ satisfy the conditions of collision-correctness, there must be an even earlier query j satisfying $x_i = x_j$ where $b = 0$ and hence we need only consider the earlier query.) And yet recall that if $b_j = 0$ and query j satisfies the conditions of collision-correctness, then $x_j = P_{k_P}(M_j)$ and the count $M_i > M_j$ since the response x_j will necessarily have increased the number of unique values by at least one. Given that P_{k_P} is a permutation and $M_i \neq M_j$ it cannot be the case that $x_i = x_j$ because that would imply $P_{k_P}(M_i) = P_{k_P}(M_j)$. This contradicts the assumption.

In case (2) we have that $b_i = 1$ and yet $x_i \neq x_j$ for any $0 \leq j < i$. Here the construction returns $x_i = P_{k_P}(j')$ where $j' \in [0, M_i - 1]$. For the condition $x_i \neq x_j$ to be true, it would have to be the case that some value in the range $[0, M_i - 1]$ has not been queried to the permutation and returned by some previous query. And yet recall that all previous queries $0 \leq i' < i$ satisfy the conditions of collision-correctness: this means that the count must have increased by at most one following any query where $b_{i'} = 0$, and did not increase at all for queries where $b_{i'} = 1$. Since for each query where $b_{i'} = 0$ the response $x_{i'} = P_{k_P}(M_{i'})$ and such queries will encompass each $M_{i'} \in [0, M_i - 1]$ then there must exist at least one past query $j < i$ such that $x_j = P_{k_P}(j')$ for $j' \in [0, M_i - 1]$, contradicting the assumption.

Security. We now sketch a brief analysis to consider the pseudorandomness of the above construction. Recall that we consider only an adversary who issues well-formed queries. Moreover, p is polynomial in $|K|$ and hence *w.l.o.g.* we can restrict our consideration to adversaries that query F_K on all input values $\{0, \ldots, p-1\}$ in sequence. Our analysis begins with the real protocol described above, and then proceeds via a series of hybrids.

Hybrid 1. In a first hybrid, we re-implement the construction above while replacing the functions $\bar{F}_{k_{\bar{F}}}$ and P_{k_P} with a random function (RF) and a random permutation (RP) respectively, each with the appropriate input/output behavior. Clearly an adversary who distinguishes this hybrid from the real protocol with non-negligible advantage implies a distinguisher for one of these two pseudorandom objects.

Hybrid 2. In a second hybrid, we replace the coins r sampled within the SampleProb procedure with a sequence of uniformly-random coins of the same length. Since the input sequence $(i\|M_i\|p\|0), \ldots, (i\|M_i\|p\|\ell)$ does not repeat and RF is a random function, it is easy to see that the resulting coins are distributed identically in this hybrid, and hence the adversary's advantage is identical to the previous hybrid.

Hybrid 3. Here we change the description of the random permutation RP oracle to an equivalent one. Consider the permutation RP to be implemented by a lazy oracle that operates as follows: when queried on each input M_i (in sequence) the oracle first samples $x_i \in [0, p-1]$ uniformly and then checks a table to see if past queries (on inputs $M_{i'} < M_i$) have returned x_i. If not, the RP returns x_i. Otherwise it samples a new x_i and repeats the test above until it finds an x_i that is not in the table. This re-sampling process will occur S'_i times to find x_i. Clearly this oracle has the same distribution as a standard random permutation and so the adversary's advantage is identical to the previous hybrid.

Hybrid 4. We now combine the construction of F and RP as follows. When F is queried on input i (recalling that duplicate queries are not permitted) it samples $x_i \in [0, p-1]$ uniformly and examines the table of previous responses from RP to see if x_i represents the response to a previous query $RP(M_{i'})$ where $M_{i'} < M_i$. If so it sets $b_i = 1$ and otherwise sets $b_i = 0$. If $b_i = 0$ then it now simulates RP as follows: it writes x_i into the table for the RP at position M_i. If $b_i = 1$ it does not modify the table for RP. In all cases it returns (x_i, b_i) as the response from F.

This hybrid requires more analysis. Let $(x_i, b_i) = F_K(i)$ be the i^{th} query response. Observe that on query $i = 0$ this hybrid samples a uniform $x_i \in [0, p-1]$ and sets $b = 0$, which is identical behavior to the previous hybrid. As i increases monotonically we must consider two cases.

1. When $b_i = 1$: For $i > 0$ any query $F(i)$ will return $b = 1$ with probability M_i/p, exactly as in the previous hybrid. This occurs because M_i is the number of unique elements in the table RP (equivalent

to the number of previous unique responses from F) and a uniform $x_i \in [0, p-1]$ will collide with at least one element with exactly that probability. Moreover, let us consider j' to be the index of that collision (if there are multiple j', we can select one at random from the options): then j' in this hybrid will be uniform in $[0, M_i - 1]$ exactly as in the previous hybrid, and the identity $x_i = RP(j')$ will be preserved.

2. When $b_i = 0$: For $i > 0$ any query $F(i)$ will return $b = 0$ with probability $1 - (M_i/p)$ as in the previous hybrid. Let $i' < i$ be the most recent query in which $b_{i'} = 0$, and define $S_i = i - i'$ as the number of times a value $x \in [0, p-1]$ has been sampled since the i'^{th} query, including sampling of the current x_i. Observe that S_i is determined by a process of sampling values in $[0, p-1]$ and comparing them to the existing table RP which has size M_i, which is precisely the process that determines S'_i in the previous hybrid. Hence statistically these terms will be identical, and thus the process of sampling x_i will be as well.

Hence it holds that the statistical distribution of this hybrid is identical to that of the previous hybrid. Critically, this final hybrid returns uniformly random values $x_i \in [0, p-1]$ from F for each query. It is easy to see that this function has behavior identical to a random function. Hence by summation across all hybrids we show that all adversaries must distinguish the real protocol from a random function with probability at most negligible. This ends the sketch.

C Full Description of CH*-MDSS

This section is structured as follows. We begin by giving lattice preliminaries and necessary terminology. Then we show that our decoding technique can be used to decode from approximately a $\frac{c}{c+1}(1-R)$ fraction of random errors, based on an assumption about the distribution of vectors in the polynomial lattice. Finally, we adapt this decoder to our semi-honest channel, where multiple codewords must be recovered.

Polynomial lattice preliminaries. See [19] for a background on polynomial lattices and lattice reduction. Let $\mathbb{F}[z]$ be the ring of polynomials over variable z with coefficients in $\mathbb{F}$. Let $\mathbb{F}(z)$ be the field of rational functions $u(z)/d(z)$, $u(z), d(z) \in \mathbb{F}[z]$. For a polynomial $f \in \mathbb{F}[z]$, its degree is $\deg(f)$; for a rational function $f(z) = u(z)/d(z)$ in lowest terms, its degree is $\deg(u) - \deg(d)$. Consider a matrix $B \in \mathbb{F}(z)^{n \times n}$ with entries that are rational functions; let the rows of B be n-dimensional vectors $b_i \in \mathbb{F}(z)^n$. The *polynomial lattice* generated by basis B is the set of vectors $L(B) = \{v \mid v = \sum_{i=1}^{n} a_i b_i, a_i \in \mathbb{F}[z]\}$. That is, the lattice consists of vectors over $\mathbb{F}(z)$ that are $\mathbb{F}[z]$ (polynomial) linear combinations of basis vectors. We define the *length* of a vector $v = (v_1, v_2, \ldots, v_n)$, $v_i \in \mathbb{F}(z)$ to be $|v| = \max_i \deg(v_i)$. Define the determinant $\det L(B) = \det B$. For a full-rank n-dimensional polynomial lattice $L(B)$, there is a polynomial-time algorithm to compute a so-called *reduced* basis B' for $L(B)$ given a basis B [40]. Such a reduced basis is guaranteed to contain a vector of length $|v| \leq (\deg \det L(B))/n$, and (unlike in the case of integer lattices) is guaranteed to contain a shortest vector of the lattice. We will also say that $\lambda_1(B)$ is the length of the shortest vector in $L(B)$. For two vectors $v, w \in \mathbb{F}(z)^n$ define the inner product $\langle v, w \rangle = \sum_i v_i \cdot w_i$. Finally, for any lattice $L(B)$ one can define the *dual* lattice $L^*(B) = \{w \in \mathbb{F}(z)^n \mid \langle w, v \rangle \in \mathbb{F}[z] \, \forall v \in L(B)\}$. Given a basis B for a full-rank lattice $L(B)$, $(B^{-1})^T$ is an explicit basis for the dual $L^*(B)$.

A note on reduced bases of lattices. While for many applications any reduced basis suffices, we are solely interested in those that are also in *popov form* [32, 44]. It is a direction of future work to determine why such bases are needed for the correctness of our algorithms. For the rest of this paper, we assume $\mathsf{LatticeReduce}(B)$ returns the popov form of B.

Below, we consider the following problem from coding theory which is related to the heuristic assumption that is needed to prove the correctness of our algorithm.

Definition C.1 (Simultaneous Curve Reconstruction Problem [20, 12])**.** *Given n, t, k, c, q, input points $\{(\alpha_i, (\beta_{i,1}, \ldots \beta_{i,c}))\}_{i \in [n]}$, where $\forall i \in [n], j \in [c]$, $\alpha_i, \beta_{i,j} \in \mathbb{F}_q$, each α_i distinct, recover all polynomial sets $P = (p_1, \ldots p_c) \in \mathbb{F}_q[x]^c$ s.t. $\forall i, \mathsf{deg}(p_i) \leq k$ and P agrees with $T \subset [n], |T| \geq t$ points. A polynomial set $P = (p_1 \ldots p_c)$* agrees *with a point $(\alpha, (\beta_1, \ldots \beta_c))$ if $\forall j \in [c], p_j(\alpha) = \beta_j$*

Algorithm 1: CH^*

Input : $\lambda, k, n, \{(\alpha_i, \beta_{i,1}, \ldots, \beta_{i,c})\}_{i=1}^{n}$
Output: None, Multiple, or a single solution $(p_1 \ldots p_c)$

forall $j \in [c]$ **do**
  $f_j(z) = \mathsf{LagrInterpol}(\{(\alpha_1, \beta_{1,j}), \ldots, (\alpha_n, \beta_{n,j})\})$
end
$N(z) = \prod_{i=1}^{n}(z - \alpha_i)$
construct the matrix $M \in \mathbb{F}[z]^{(c+1)\times(c+1)}$:

$$M = \begin{bmatrix} z^k & f_1(z) & f_2(z) & \ldots & f_c(z) \\ & N(z) & & & \\ & & N(z) & & \\ & & & \ddots & \\ & & & & N(z) \end{bmatrix}$$

$M_{\mathsf{red}} \leftarrow \mathsf{LatticeReduce}(M)$
if *there is one shortest vector* $\vec{v} = (v_0 \ldots v_c)$ *that has a length* $\leq \lambda$ **then**
  return $(v_1 \cdot z^k/v_0, \ldots, v_c \cdot z^k/v_0)$
end
else
  return Fail
end

Figure 16: An algorithm CH^* for the simultaneous curve reconstruction problem that recovers from some number of random errors based on λ. λ is an upper bound on the length of a target vector which corresponds to a solution $P = (p_1 \ldots p_c)$ of the simultaneous curve reconstruction problem.

Given a set of input points $\{(\alpha_i, \beta_{i,1}, \ldots, \beta_{i,c})\}_{i=1}^{n}$, $(p_1 \ldots p_c)$ may be called a *solution* or a *solution set* if it agrees with at least t input points.

We briefly make some remarks about this problem from what is known in the literature. First, the difficulty of this problem depends heavily on the distribution of the points $\{(\alpha_i, (\beta_{i,1}, \ldots \beta_{i,c}))\}_{i\in[n]}$. When there is one solution, and all other points are randomly sampled (i.e. $\alpha_i, \beta_{i,j} \xleftarrow{\$} \mathbb{F}_q$) we can *theoretically* recover the solution P with high probability, provided that $t > (nk^c)^{\frac{1}{c+1}} + k + 1$ [20].We know of additional algorithms that are conjectured to achieve the more optimum bound of $t > (nk^c)^{\frac{1}{c+1}}$ [45].

To the best of our knowledge, there are no algorithms in the coding theory literature which solve this problem in a different setting from the random error model.

C.1 A warm-up for random noise channels

We begin by discussing an algorithm that solves the simultaneous curve reconstruction problem in the case of random errors and then extend our results to the channel specified in Definition 3.7. Because we want our algorithms to run on devices with limited computation power and storage, we take as our starting point a very efficient, lattice-based decoder by Cohn and Heninger [19].

As previously discussed, the decoding algorithm of Cohn and Heninger constructs a polynomial lattice whose basis vectors are weighted polynomial coefficient vectors in $\mathbb{F}[z]$ of multivariate polynomials $Q(x_1 \ldots x_c) \in \mathbb{F}[z, x_1, \ldots, x_c]$. They then compute a reduced basis of this lattice, map short vectors back to multivariate polynomials, and solve the polynomial system. The solution $v_p = (p_1 \ldots p_c)$ is heuristically shown to be a solution of the system by construction. The decoding algorithm of Cohn and Heninger has

associated with it two parameters `t` (degree) and `k` (multiplicity) which, if set optimally, can allow the decoder to recover from a large number of errors at the cost of a sharp increase in runtime. For efficiency reasons, we consider only the linear variant where degree and multiplicity are set to one.

While Cohn-Heninger is preferable to non-lattice based decoders, one downside of the algorithm is that many short vectors must be extracted from the reduced basis before solving a system of polynomials.

We observe that a target vector corresponding to any solution $p_1 \dots p_c$ is present in the dual of the lattice constructed by Cohn and Heninger, because its dot product with any vector in the primal is a multiple of the syndrome polynomial by construction. We also know that this target vector is *short* because we have degree bounds on the p_i. If the target vector is the shortest vector in this dual lattice, then it will appear in a reduced basis.

We give the explicit construction of this lattice and describe how to recover candidate solutions from the corresponding reduced basis in Figure 16; we call this algorithm CH*. The lattice basis M constructed in Figure 16 is the dual lattice of the coefficient lattice with degree and multiplicity one, re-scaled by a factor of $N(z) \cdot z^k$ where $N(z) = \prod_i (z - \alpha_i)$ and k is an upper bound on the desired degree of the solution polynomials p_i so that all entries are in $\mathbb{F}[z]$ rather than $\mathbb{F}(z)$ for ease of computation.

We now make a correctness claim regarding when the algorithm succeeds, given input corresponding to solving the simultaneous curve reconstruction problem in the presence of bounded random noise. This claim is heuristic based on the conjecture that the lattice M from Figure 16 behaves like a random lattice with a planted target vector corresponding to the solution.

Heuristic Assumption 2. *The non-target vectors in lattice M as described in Figure 16 behave like a random lattice when M is instantiated from evaluations $\{(\alpha_i, \beta_{i,1}, \dots, \beta_{i,c})\}_{i=1}^{n}$ of Reed-Solomon codes with distinct α_i and errors are chosen uniformly at random.*

Claim C.0.1. *Under Heuristic Assumption 2, instantiating Figure16 with target vector length $\lambda = \frac{1}{c+1}(k + cn) - 1$ will recover, if it exists, a solution set $(p_1 \dots p_c)$ that agrees with $t \geq \frac{1}{c+1}(ck + n) + 1$ input points.*

Proof Sketch. For a fully random lattice, we expect the length of the shortest vector v in $L(B)$ to satisfy $|v| = \lfloor (\deg \det L(B))/(\dim L(B)) \rfloor$. Thus if our target vector v_p is smaller than this bound, we expect it to be the shortest vector in the lattice and thus to be present in a reduced basis.

By construction, we have $|v_p| = k + (n - t)$[9], and

$$\deg(\vec{b_1}) \leq \frac{\deg(\det(B))}{\dim(B)} = \frac{k + c \cdot n}{c + 1} \tag{26}$$

In particular, since the degree of a polynomial must be a non-negative integer,

$$\deg(\vec{b_1}) \leq \left\lfloor \frac{k + c \cdot n}{c + 1} \right\rfloor$$

Solving for t, we expect the algorithm to succeed when $t \geq \frac{1}{c+1}(ck + n) + 1$. □

We note that the way the parameter t is set in Claim C.0.1 guarantees that the length of the target vector, λ, is always less than the shortest vector in a random lattice with determinant corresponding to our chosen parameters. Empirically, we observed that the algorithm presented in Figure 16 does succeed sometimes even when we set t so that these vectors have the same length. Let $m = k + cn \pmod{c+1}$. We observed that our target vector was the shortest in the lattice when $m = 1$. When $m \neq 1$, there were $c + 1 - m$ vectors in the reduced basis achieving the shortest vector length. While none of these short vectors equaled the expected target vector, we found empirically that scalar linear combinations of these vectors would produce the target (to be precise, given short vectors $\vec{v_1}, \dots \vec{v_\ell}$, $\vec{t} = \sum_{i=0}^{\ell} \vec{v_i}$ was the target).

Using Claim C.0.1 and the preceding empirical results, the construction in Figure 16 will succeed provided that $t \geq \frac{1}{c+1}(ck + n)$, when setting $\lambda = \frac{1}{c+1}(cn + k)$. We experimentally verified CH* on 10,000 randomly selected inputs for various configurations of n, k, t and c and observed that it succeeded in practice.

[9]The target vector corresponding to a solution $(p_1, \dots p_c)$ will have the form $(z^k \cdot E(z), p_1(z) \cdot E(z), \dots, p_c(z)) \cdot E(z))$ in the scaled dual lattice, where $E(z)$ is an error locator polynomial

C.2 Handling multiple dealers

While the previous decoder works in the case of random errors, it is clearly not useful for constructing an $MDSS$ scheme when the number of dealers is greater than one. One way to handle multiple dealers is to simply use a decoder from the coding theory literature that can handle fully malicious substitution channels, but no such decoder is known for IRS codes and currently it seems to be a fundamentally hard problem to come up with one. Fortunately, the channel we consider is *not* fully malicious but in fact semi-honest; dealers will only ever supply values introduced by running the Share algorithm. This means the input points distribution to the simultaneous curve reconstruction will have special properties; in particular, for a large enough field it is *highly unlikely* than any single input point will correspond to more than one solution. This suggests a natural iterative strategy for finding solutions; first, use an algorithm which can find a single solution; then, once a solution is found, remove all agreeing input points; go back to the first step, repeating until all existing solutions are found.

The previous strategy requires an algorithm which can find a single solution even in the presence of random noise *and* give a clear indicator when no such solution exists (since this tells us when the algorithm should terminate). Through experiments and simulations, we make the rather surprising observation that the CH^* algorithm can recover such a solution for cases where the number of agreeing points for a single solution is a little bit larger than all other existing solutions; in particular, it must have two more agreeing points than any other solution. For the latter issue, we observed that if the shortest vector in the reduced lattice of CH^* is greater than λ, or if the vector has length exactly λ and is not a solution, then no such solution exists, and the algorithm can safely terminate.

To summarize, we empirically identified a class of "bad" instances upon which our algorithm fails and a class of "good" instances upon which the algorithm succeeds. In a stroke of luck, it is even possible to *transform* bad input instances into good ones by using randomness; randomly choose a number of points to delete and run the first algorithm until a solution is found. This is already enough to construct the single solution algorithm we asked for earlier. However, it does have some undesirable properties. The runtime of the algorithm depends heavily on the expected number of lattice reductions needed to transform bad instances into good ones and - in practice - the likelihood of getting bad instances when using our decoder in an $MDSS$ scheme is relatively high since dealers will often supply about the same number of shares. This leads us to look for other algebraic solutions which have a runtime that only has a small linear dependency on the number of solutions and much smaller overall variance.

C.3 CH^*-MDSS

Through empirical analysis, we found that the lattice of Figure 16 contains a wealth of information about the structure of existing solutions that could potentially be used in a more efficient list-decoding algorithm than the one described in the last section. We first state a few key observations we made from looking at the reduced lattice basis on "hard" input instances (e.g. there are three solution sets, two agreeing with X input points, one with $X-1$ or one solution set agreeing with X, the other two $X-1$, etc) .

1. When there is one solution set that has at least one more agreeing point than any other set, scalar linear combinations of the shortest vectors in M (with each scalar equal to $1 \in \mathbb{F}$) produces the target vector associated with this solution set.

2. Otherwise, let $\vec{v} = (v_0 \dots v_{c+1})$ be a shortest vector within the reduced lattice. When multiple solutions have the maximum number of agreeing points, v_0 contains all factors $(z-\alpha_i)$, $i \in [n]$ that do *not* agree with these solution sets.

Observe that using 1 was also what allowed us to change the threshold for the *single* valid solution case: $t \geq \frac{1}{c+1}(c \cdot k + n)$ suffices instead of the bound implied in Claim C.0.1. More importantly, it reduces the class of hard instances. We now only need to deal with cases where solutions have the same number of agreeing points. With 2, we have a way to identify all points that correspond to the multiple solution sets. If we could identify all the points associated with a single solution from this set, then we are done. The main insight we

have is that we know the form of our solutions vectors: they must look like $z^k E(z), E(z)p_1(z), \ldots, E(z)p_c(z)$ for some $E, p_1 \ldots, p_c$ of appropriate degree where E is an error locator polynomial. We choose a point α that corresponds to one of the multiple solutions and construct a lattice problem to try and find all vectors in M that are multiples of $(z - \alpha)$. We know that, if there are x solutions, there are $x - 1$ such solutions that will have this form. When we add the shortest vectors in M that come from this lattice problem, we obtain a vector $(r_0 \ldots r_c)$ and an associated rational function $R = (r_1/r_0, \ldots r_c/r_0)$ which agrees with all points associated with $x - 1$ solution sets *excluding* the points from the solution set that agrees with α. Doing lagrangian interpolation on the excluded points produces the polynomial set $(p_1 \ldots p_c)$ agreeing with α. Moreover, if $x = 2$, R is directly equal to the other solution set. As an optimization, we check for this scenario every time we run the second reduction to prevent unnecessarily doing lattice reductions. Unlike the solution of the pervious section, this algorithm only needs $\approx 2s$ lattice reductions where s is the number of solutions. The whole algorithm is depicted in Figure 2.

D Alternative approaches to MDSS

We considered numerous candidate MDSS constructions, including other families of Reed-Solomon codes and insertion codes. Unfortunately, all other potential constructions had properties rendering them undesirable for our setting.

List-Decodable Reed-Solomon Codes. There is a rich existing literature on list-decoding of Reed-Solomon codes [20, 43, 26, 27]. However, each has its own weaknesses. [20] succeeds only if there is a single dealer. [43] only performs well under unlikely ambient noise regimes. This leaves [27], which we use as the basis of Construction 14 and has the weaknesses we discuss in §3.2, and [26], which we discuss below.

Folded Reed-Solomon Codes The codes presented in [26] indeed allow for parameters approaching the optimal bound of $t_{rec} > t_{priv}$. Similar to CH*-MDSS, these codes operate by including multiple polynomial evaluations in each share, where the number of evaluations per share is denoted by the "folding parameter" m. However, unlike CH*-MDSS, in Folded Reed-Solomon each evaluation is on the *same polynomial*. In our setting, this means that $t_{priv} = k/m$, where k is the polynomial's degree. To the best of our knowledge, the best known bound relating t_{rec}, t_{priv}, max, and m is Theorem 4.2 from [26], simplified below, and expressed in terms of our parameters:

$$t_{rec} \geq \sqrt[s+1]{N \left(\frac{m \cdot t_{priv}}{m - s + 1} \right)^s}$$

where $1 \leq s \leq m$ is an additional free variable.

To achieve parity with the values shown in Table 1, the above bound requires that $m \geq 40$. Given 248 bits of available BLE space (see §5), this would require a field size of $|\mathbb{F}| \leq 2^6$, a number impractical for our application.

Additionally, again to the best of our knowledge, there exist no concretely efficient decoder implementations for these codes. We leave the task of further adapting these codes to our setting to future work.

Insertion Codes Note that the channel we consider in this work is one that injects random symbols rather than replacing or erasing, as is typically the setting for Reed-Solomon codes. It is feasible that if we used codes designed for an insertion-only setting we may be able to come up with a better $MDSS$ construction. While there are codes that can tolerate any polynomially bounded number of insertion errors [28], they unfortunately rely on a labeling technique that conflicts with our desired unlinkability property. We leave the exploration of the applicability of these codes to multi-dealer secret sharing schemes for future work.

Algorithm 2: CH*-MDSS

Input : $k, t, n, \{(\alpha_i, \beta_{i,1}, \ldots, \beta_{i,c})\}_{i=1}^{n}$
Output: a list $\{(p_1^i \ldots p_c^i)\}_{i=1}^{z}$ or $\perp$

$\mathsf{solns} := []$,$\mathsf{ws} := [n]$, $\mathsf{fail} := False$
while $|\mathsf{ws}| \geq t$ *and not* fail **do**
  $\forall j \in [c], f_j(z) = \mathsf{LagrInterpol}(\{(\alpha_i, \beta_{i,j})\}_{i \in \mathsf{ws}})$, $N(z) = \prod_{i \in \mathsf{ws}}(z - \alpha_i)$

$$M = \begin{bmatrix} z^k & f_1(z) & f_2(z) & \ldots & f_c(z) \\ & N(z) & & & \\ & & N(z) & & \\ & & & \ddots & \\ & & & & N(z) \end{bmatrix}$$

  $M_{\mathsf{red}} \leftarrow \mathsf{LatticeReduce}(M)$
  if $\lambda_1(M_{\mathsf{red}}) > k + (|\mathsf{ws}| - t)$ **then**
    set fail to $True$
  else
    $\vec{v_1}, \ldots \vec{v_h} \leftarrow \mathsf{Find}(M_{\mathsf{red}}, \lambda_1(M_{\mathsf{red}}))$
    $V \leftarrow \mathsf{Translate}(\sum_{i=1}^{h} \vec{v_i}, k)$
    if $\mathsf{IsSol}(V, \mathsf{ws}, t, k)$ **then**
      $\mathsf{ProcessSol}(V, \mathsf{solns}, \mathsf{ws})$
    else
      if $\lambda_1(M_{\mathsf{red}}) == k + (|\mathsf{ws}| - t)$ **then**
        set fail to $True$
      else
        choose $i \in \mathsf{ws}$, s.t. $(z - \alpha_i) \nmid v_0$
        $\vec{b_1}, \ldots, \vec{b_m} \leftarrow \mathsf{Find}(M_{\mathsf{red}}, k + (|\mathsf{ws}| - t))$
        $(b_0^i, \ldots b_c^i) \leftarrow \vec{b_i}, \forall i \in [m]$

$$S = \begin{bmatrix} z^{|ws|} \cdot b_0^1 & b_1^1 & \ldots & b_c^1 & & & \\ \vdots & & & & & & I_m \\ z^{|ws|} \cdot b_0^m & b_1^m & \ldots & b_c^m & & & \\ z^{|ws|} \cdot (z - \alpha_i) & 0 & \ldots & \ldots & \ldots & \ldots & 0 \end{bmatrix}$$

        $S_{\mathsf{red}} \leftarrow \mathsf{LatticeReduce}(S)$
        $\vec{s_1}, \ldots \vec{s_h} \leftarrow \mathsf{Find}(S_{\mathsf{red}}, \lambda_1(S_{\mathsf{red}}))$
        $\vec{r_i} \leftarrow \sum_{j=0}^{m-1} s_{c+1+j}^i \vec{b_j}, \forall i \in [h]$
        $R \leftarrow \mathsf{Translate}(\sum_{i=1}^{h} \vec{r_i}, \mathsf{ws})$
        if $\mathsf{IsSol}(R, \mathsf{ws}, t, k)$ **then**
          $\mathsf{ProcessSol}(R, \mathsf{solns}, \mathsf{ws})$
        $(q_1, \ldots, q_c) \leftarrow R$, $M \leftarrow \{i \mid \forall j \in [c], q_j(\alpha_i) = \beta_{i,j}\}$, $I \leftarrow \{i \in \mathsf{ws} \mid (z - \alpha_i) \nmid v_0\}$, $L \leftarrow I \setminus M$
        $\forall j \in [c], p_j \leftarrow \mathsf{LagrInterpol}(\{(\alpha_i, \beta_{i,j})\}_{i \in L})$
        if $\mathsf{IsSol}((p_1 \ldots p_c), \mathsf{ws}, t, k)$ **then**
          $\mathsf{ProcessSol}((p_1, \ldots p_c), \mathsf{solns}, \mathsf{ws})$
endw
return solns

E AROF predicates and proofs (§4)

E.1 Predicates P_s and P_h

In this section we give the predicates for detectability and tag indistinguishability for our construction in Figure 11. For detectability we will need assumptions on the input to the Detect algorithm that are enforced by the MDSS parameters. In terms of the AROF, these assumptions will correspond to there being less than or equal to the maximum number of stalkers in the vicinity, with small enough noise coming from other tags so that no more than max beacons will be given to the detection algorithm. If these assumptions hold, then any tag that has supplied at least t_{rec} beacons corresponding to the same detectability period will be identified. Below is the precise statement for the predicate.

$$P_s(\mathsf{cfg}, Q, \mathsf{id}) = \text{“ } \exists\ e \text{ such that } |\{(\mathsf{id}, i) \in Q \mid e = \lfloor \tfrac{i}{L} \rfloor\}| \geq t_{rec},$$
$$|\{(\mathsf{id}', e) \text{ s.t. } |\{(\mathsf{id}', i) \in Q | e = \lfloor \tfrac{i}{L} \rfloor\}| \geq t_{rec}\}| \leq d,\ |Q| \leq \mathsf{max}\text{”}$$

For tag indistinguishability, the situation is simpler. Indistinguishability holds as long as no party ever sees more than t_{priv} beacons for either tag within any detectability period.

$$P_h(\mathsf{cfg}, Q) = \text{“ for } \mathsf{id} \in \{0,1\},\ \forall e \in \mathbb{Z}^+, |\{(\mathsf{id}, i) \in Q \mid e = \lfloor \tfrac{i}{L} \rfloor\}| \leq t_{priv} \text{ and } Q \text{ is not a multi-set”}$$

The condition that Q is not a multi-set is meant to rule out an attack where the adversary queries on the challenge indices i_0^* or i_1^* before the challenge phase: by doing so they can trivially win with an equality check on the challenge beacon.

E.2 Proof of Tag Indistinguishability

Theorem E.1. *The construction presented in 11 is tag indistinguishable.*

Proof. Throughout this section, we use two variables i_{anon} and e. Both of these are defined from the epoch $i_{\mathsf{epoch}} = i$ that is queried on and a parameter L - which is a part of cfg - as $i_{\mathsf{anon}} = i \mod L$ and $e = \lfloor \frac{i}{L} \rfloor$.

We will make a standard hybrid argument, reducing to the t_{priv}-unlinkability of the $MDSS$ scheme.

When hybrids are described, if details are omitted, it is assumed that they are the same as described in the previous hybrid. $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Tag},P,\mathcal{H}_i}$ denotes the output of modified experiment suggested by $\mathcal{H}_i$.

$\mathcal{H}_0$ = The tag indistinguishability game for predicate P, as described in figure 7, for the construction in Figure 11, with the bit $b = 0$

In our first hybrid, we try to guess the abuse epoch that $\mathcal{A}$ will query on. Because $\mathcal{A}$ is a polynomial time adversary, they can only make at most a polynomial number of queries. Meaning the advantage we lose from this step is at most $\frac{1}{\mathsf{poly}(\lambda)}$. To be more precise,

$\mathcal{H}_1$ = Let $q(\lambda)$ be an upper bound on the number of queries that $\mathcal{A}$ will make. Guess both of queries q which *first* introduce the abuse epochs that will be challenged on (note: this could be the challenge query itself and the abuse epochs that are challenged on might be the same for both tag keys). If the guess is incorrect, output 0.

Claim E.1.1. $\Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Tag},P,\mathcal{H}_1}(\lambda) = 1\right] = \frac{1}{q(\lambda)^2} \Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Tag},P,\mathcal{H}_0}(\lambda) = 1\right]$

The proof of this claim is straightforward, as the chance that the challenger samples correct guesses for both epochs - when sampling independently and uniformly at random - is equal to what is in the claim, regardless of $\mathcal{A}$'s query pattern. Let $e^{*,b}$ and $i_{\mathsf{anon}}^{*,b}$ be the respective values defined at the start of this section for $b \in \{0,1\}$ in the challenge query.

We now state a series of hybrids that all have an almost identical argument, switching out PRF evaluations with varous keys for random values. This is not complicated mainly because all PRF keys are independent of one another and no other function of the keys is given out.

$\mathcal{H}_{i,j}$ = Replace PRF evaluations using k_i with sampling random values for k^j_{tag}

There are six such hybrids, for $i \in \{1,2,3\}$ and $j \in \{0,1\}$. Choose some ordering, and re-label these hybrids as $\mathcal{H}_2, \ldots, \mathcal{H}_6$ (no particular ordering will be necessary for the claim to follow).

Claim E.1.2. *$\forall j \in [2,6]$, if the PRF is secure, then there exists a negligible function negl, s.t.*
$|\Pr\left[\mathsf{Exp}^{\mathsf{Tag},P,\mathcal{H}_j}_{\mathcal{A}}(\lambda)=1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{Tag},P,\mathcal{H}_{j-1}}_{\mathcal{A}}(\lambda)=1\right]| \leq negl(\lambda)$

We will not provide all hybrid arguments and just give an exemplary one for $\mathcal{H}_{2,0}$. Let $j^* \in [2,6]$ be the location of $\mathcal{H}_{2,0}$ in the ordering from earlier. Recall that
$\mathcal{H}_{2,0}$ = Replace PRF evaluations using k_2 with sampling random values for k^0_{tag}

We construct an adversary $\mathcal{B}$ that succeeds with non-negligible advantage in the PRF security game if $\mathcal{A}$ performs non-negligibly better in either $\mathcal{H}_{j^*-1}$ or $\mathcal{H}_{j^*}$.
Description of $\mathcal{B}$:

- Generate k^0_{tag} and k^1_{tag} using $\mathsf{KeyGen}(\mathsf{cfg})$.
- Let i be an epoch associated with k^0_{tag} that $\mathcal{B}$ must respond on (either from a non-challenge query where $\mathsf{id}=0$ or from the challenge query from $\mathcal{A}$). Calculate $e = \lfloor \frac{i}{Z} \rfloor$ and ask the PRF challenger for evaluation at e, receiving r. Use r to construct $sh^e_{i_{\mathsf{epoch}}}$, construct pk as normal.
- Let the output of $\mathcal{A}$ be b'. Output b'.

Analysis:
Let $\mathsf{Exp}^{\mathsf{Prf},b}_{\mathcal{B}}$ be the experiment for the PRF security game.

$$\begin{aligned}|\Pr\left[\mathsf{Exp}^{\mathsf{Prf},1}_{\mathcal{B}}(\lambda)=1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{Prf},0}_{\mathcal{B}}(\lambda)=1\right]| &= |\Pr\left[\mathcal{A} \text{ outputs 1 in } \mathcal{H}_{j^*}\right] - \Pr\left[\mathcal{A} \text{ outputs 1 in } \mathcal{H}_{j^*-1}\right]| \\ &= |\Pr\left[\mathsf{Exp}^{\mathsf{Tag},P,\mathcal{H}_{j^*}}_{\mathcal{A}}(\lambda)=1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{Tag},P,\mathcal{H}_{j^*-1}}_{\mathcal{A}}(\lambda)=1\right]| \\ &\leq negl(\lambda)\end{aligned}$$

It is now possible to reduce to the unlinkability of the MDSS scheme. Consider the hybrid:
$\mathcal{H}_7$: The challenge share sh^* in the challenge query is replaced with the share associated with $e^{*,1}$ and $i^{*,1}_{\mathsf{anon}}$.

Claim E.1.3. *If the MDSS scheme is t_{priv} unlinkable then, $\forall \mathcal{A}$, $\exists$ a negligible negl such that*
$|\Pr\left[\mathsf{Exp}^{\mathsf{Tag},P,\mathcal{H}_7}_{\mathcal{A}}(\lambda)=1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{Tag},P,\mathcal{H}_6}_{\mathcal{A}}(\lambda)=1\right]| \leq negl(\lambda)$

We consider the following adversary $\mathcal{B}$ against $\mathsf{OM\text{-}ULink}$.

Description of $\mathcal{B}$:

- Generate $k^b_{\mathsf{tag}} \leftarrow \mathsf{KeyGen}(\mathsf{cfg})$, $s_b \xleftarrow{\$} \mathcal{S}$ for $e^{*,b}$, $b \in \{0,1\}$
- Let $I_0 = \{1, ..., i^{*,0}_{\mathsf{epoch}}\}, I_1 = \{1, \ldots, i^{*,1}_{\mathsf{epoch}}\}$.Send to the challenger $(s_0, i^{*,0}_{\mathsf{epoch}}, i^{*,0}_{\mathsf{epoch}}, I_0, I_0 \setminus \{i^{*,0}_{\mathsf{epoch}}\})$, $(s_1, i^{*,1}_{\mathsf{epoch}}, i^{*,1}_{\mathsf{epoch}}, I_1 \setminus \{i^{*,1}_{\mathsf{epoch}}\}, I_1)$. Let sh' be the one ambiguous share, $sh^b_1, \ldots sh^b_{i^{*,b}_{\mathsf{epoch}}-1}$ the unambiguous shares for $b \in \{0,1\}$
- When answering $\mathcal{A}$'s queries directed to $\mathsf{id}=b$, calculate $i_{\mathsf{anon}} = i \pmod L, e = \lfloor \frac{i}{L} \rfloor$. If $e \neq e^{*,b}$, proceed as normal. Otherwise, choose the correct share from $sh^b_1, \ldots sh^b_{i^{*,b}_{\mathsf{epoch}}-1}$ to send to $\mathcal{A}$.

- For the challenge query, send sh'.
- Receive $\hat{b}$ from $\mathcal{A}$ and output $\hat{b}$

Analysis:
Let $\mathsf{ExpLink}^b_{\mathcal{B}}$ be the experiment for unlinkability using the bit b.

$$\begin{aligned}|\Pr\left[\mathsf{ExpLink}^1_{\mathcal{B}}(\lambda)=1\right]-\Pr\left[\mathsf{ExpLink}^0_{\mathcal{B}}(\lambda)=1\right]| &= |\Pr\left[\mathcal{A}\text{ outputs 1 in }\mathcal{H}_7\right]-\Pr\left[\mathcal{A}\text{ outputs 1 in }\mathcal{H}_6\right]\\ &= |\Pr\left[\mathsf{Exp}^{\mathsf{Tag},P,\mathcal{H}_7}_{\mathcal{A}}(\lambda)=1\right]-\Pr\left[\mathsf{Exp}^{\mathsf{Tag},P,\mathcal{H}_6}_{\mathcal{A}}(\lambda)=1\right]|\\ &\leq negl(\lambda)\end{aligned}$$

This ends our analysis. The rest of the argument is symmetric to get back to the original tag indistinguishability game using the bit $b=1$. This completes the proof.

□

E.3 Proof of Detectability

Theorem E.2. *The construction in 11 is detectable.*

Detectability holds via the properties of the list decoding algorithms. The number of shares is bound to max, there are at least t_{rec} shares from the same polynomial and the number of "dealers" present in the output is bounded to d. Note that a single tag can be equivalent to multiple dealers or one dealer with some additional shares corresponding to an insufficient dealer and the predicate still accounts for this. Detectability trivially follows. When we use heuristic algorithms, detectability is only heuristically guaranteed.

F Tuning MDSS Parameters

We now give more details on how we derived the parameters shown in Table 1.

Bandwidth, epoch duration, number of polynomials. To determine available bandwidth and broadcast rate we examined existing LTA deployments, focusing on Apple's FindMy [29, 30]. Apple's LTAs use a broadcast interval of 2 seconds and an anonymity epoch duration (*i.e.,* rotation period for the LTA identifier) of 15 minutes in near-owner mode, or 24 hours in separated mode. The 2-second broadcast interval gives a lower bound on the anonymity epoch duration.

The legacy BLE protocol provides up to 25 bytes for payload in each advertisement, but FindMy extends this by using 46 bits of the MAC address field [30].[10] This leaves 246 available bits for the protocol to transmit data.[11] Nearly all of this bandwidth is needed to transmit a 225-bit pseudonym/public key pk used for reporting. Since our protocols must also transmit a similar pk in addition to a secret share, we propose to divide the required beacon broadcasts into two consecutive broadcasts, a first broadcast containing pk and auxiliary info and a second containing only a secret share. These broadcasts can be emitted every period, doubling the number of broadcasts but maintaining the original broadcast rate. Alternatively we can alternate broadcasts and reduce the broadcast period to 4 seconds.

As described in §3.2, our secret sharing construction includes a number of polynomial evaluations in each secret share (denoted by c). The available bandwidth and the size of the field used place an upper bound on c. Using a field $\mathbb{F}_p$ with a 22-bit representation allows us to set $c=10$. A field $\mathbb{F}_p$ with a 24-bit representation gives $c=9$.[12]

Empirically measuring LTA broadcast noise. A major consideration is the amount of noise that the Detect algorithm must be robust to. This value determines the parameters (and performance) of the underlying MDSS scheme. In practice we anticipate two sources of noise:

[10] Two bits must remain unused to conform to the BLE standard.
[11] Future LTAs may use BLEv5, which supports up to 255 bytes of advertisement payload.
[12] We select prime fields $\mathbb{F}_p$ with p chosen as the largest prime of the specified bit-length.

Location	Duration	# total broadcasts	# duplicate broadcasts	Unique LTAs	Average LTAs in range	Average LTAs in range (5 min)
NYC	1 Hour	126	74	50	0.06	0.17
Toronto	1 Hour	21	9	12	0.01	0.08
Washington, DC	2.5 Hours	322	267	49	0.07	0.35

Table 4: FindMy (AirTag) advertisement collection. This table shows the statistics gathered during three data collection experiments. Headers represent: total number of broadcasts detected, total number of duplicate broadcasts, total number of unique LTAs encountered, the average number of devices within range of the receiver at any instant over the entire collection period (1 is equivalent to a single AirTag broadcasting continuously), and the average number of devices in range during the 5-minute period with the heaviest broadcast traffic.

1. Broadcasts transmitted by ephemeral (non-stalking) LTAs. These broadcasts will likely correspond to the *insufficient* dealers in the underlying MDSS scheme, and will therefore appear as *random* points.
2. Broadcasts transmitted by additional persistent (and possible stalking) LTAs, corresponding to the *sufficent* dealers in the underlying MDSS.

In Spring 2023 we conducted some limited experiments on Apple's Bluetooth-based FindMy network to measure the background (ephemeral) broadcasts sent by deployed LTAs and determine the number of *separated-mode* AirTag broadcasts encountered in major metropolitan areas.

Experimental Setup. To collect AirTag broadcasts we configured a Raspberry Pi 4 with an external GPS receiver. We configured the Pi to record all Bluetooth packets that match the separated-mode AirTag FindMy advertisement format as described by Heinrich *et al.* [29], and recorded each received broadcast along with the current time and GPS coordinates.[13]

Experiments. Using the Pi configured as described above, we walked through densely-populated areas of three major North American cities: New York City, USA (NYC), Toronto, Canada, and Washington DC, USA, logging all FindMy traffic received by the Pi.

Table 4 presents statistics of the data collected in our experiments.

Discussion. Our experiments observed many separated-mode AirTag broadcasts from AirTags deployed outside of the range of an owner device and suggest some rough lower-bounds on the ephemeral LTA noise rate for our later experiments. We express these noise calculations in terms of the number of LTAs in range of the receiver averaged over some time period. (For example, a receiver standing next to a single LTA would be within range of 1 LTA, or 1800 broadcasts per hour.)

In the experiment with the largest number of transmissions (Washington, DC), we measured an average of 0.07 LTAs within range of our receiver over the entire experimental run. However we noted bouts of increased broadcast density over five-minute periods that peaked at the equivalent of 0.35 LTAs. We stress that the FindMy network is growing rapidly, and so these numbers likely represent a lower bound on future noise rates.

Establishing privacy bounds. A main challenge in MDSS is minimizing the gap (or "ramp") between the number of shares t_{priv}, where privacy is no longer guaranteed for the sender, and t_{rec}, the number of shares a receiver must obtain in order to recover the secret. This ramp depends on external conditions like the noise rate of the channel, and is larger when we wish to tolerate higher noise rates.

Accounting for noise and missed broadcasts. There are four parameters that determine achievable t_{priv} and t_{rec} values:

- The number of persistent LTAs (candidate "stalkers") within range who contribute a sufficient set of shares.
- The number of *non-stalker* ephemeral LTA broadcasts.

[13]Code for our scanner can be found at the anonymous repository `https://github.com/anonUSENIX24/submission`

- The expected fraction of erasures, *i.e.,* broadcasts lost due to radio interference or x-coordinate collisions.
- The total number of shares collected and passed to the Detect algorithm by victim devices.

Determining t_{rec} and t_{priv}. After setting the parameters above, we select a value for t_{rec} for which we expect $> 99\%$ decoding success for any given LTA. We give a sketch of the calculation used to do so here. Let L be the number of consecutive detection windows before the parameters of an LTA are rotated. Let n be the number of shares output by an LTA in each detection window, and let N be the number of times each share is broadcast. Then, the probability that a share is dropped due to *collision in the x-coordinate* (from either the broadcast of a separate LTA or a previous broadcast of the same LTA, requiring a noise share) is: $p \leq 1 - \left(\frac{|\mathbb{F}|-1}{|\mathbb{F}|}\right)^{(L-1+\mathsf{max})*n-1}$. Next, the probability that a share is dropped due to *deletion by the channel* is $p' = (0.05)^N$ (for a pessimistic assumption of a channel with 5% drop-rate). With p we then find the smallest z such that the probability that z or fewer shares are dropped due to collisions is > 0.995 by summing binomial probabilities. We then repeat the process for shares dropped by the channel resulting in z'. Finally, we set $t_{rec} = n - z - z'$, and given t_{rec} set t_{priv} to the maximum allowable value under the bounds of the decoding algorithm. The desired decoding success rate then follows from the union bound.

Illustrating privacy. We now attempt to give intuition for how the identified parameters affect the achievable privacy of our scheme.

Increasing c. We begin with the bound enforced by CH*-MDSS: $t_{rec} \geq \frac{1}{c+1}(c \cdot (t_{priv}) + \mathsf{max})$. As c increases, this bound approaches $t_{rec} \geq t_{priv} + 1$, although diminishing returns set in fairly early, around $c = 20$. This affect is illustrated for a concrete choice of parameters in Figure 17.

Increasing Tolerable Noise. Our scheme can be configured to tolerate larger amounts of noise (from both persistant and ephemeral LTAs) by increasing max. This increase corresponds to a decrease in t_{priv} by the CH*-MDSS bound. Figure 18 shows how the achievable privacy degrades linearly with noise, with the amount of noise represented in terms of the maximum number of concurrent stalkers tolerated by the scheme.

Field Size and Accounting For Deletions. As explained above, we account for deletions by decreasing t_{rec}, which in turn decreases t_{priv} and the maximum achievable privacy. The more deletions we need to tolerate, the lower the achievable privacy. Deletions can come from two sources: physical layer issues and x-coordinate collisions. Most impactful are the x-coordinate collisions, which can cause shares to be dropped in two possible ways.

1. As discussed in §3.2, LTAs output "noise shares" when they sample an x-coordiante that collides with one previously sampled. As noise shares do not aid in recovery, they have the same effect as a dropped share.
2. CH*-MDSS cannot tolerate inputs in which the shares have colliding x-coordinates. Therefore, shares that collide with the broadcasts of *other* LTAs must be dropped before the recovery algorithm is run.

For both cases, as the field size increases the rate of collisions goes down, leading to a drop in the number of deletions the scheme needs to tolerate, and therefore an increase in achievable privacy. This relationship is shown in Figure 19

Accounting for system constraints. While high c and $|\mathbb{F}|$ is usually preferable as shown above, because of bandwidth considerations, i.e. the extremely small size of pre-v5 BLE packets, we cannot have both be high simultaneously; for example, the more polynomials we use, the smaller the field must be, increasing the probability of collision and decreasing the amount of privacy we can achieve. On the other hand, increasing $|\mathbb{F}|$ will decrease c and thus also decrease privacy. For the current limitations on BLE packets, we choose our parameters to maximize the amount of privacy by ensuring that we are not increasing c for almost no benefit or losing too many points to collisions.

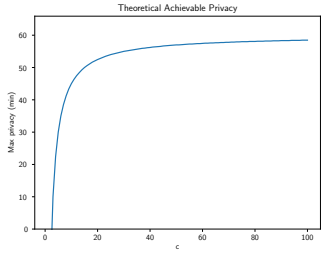

Figure 17: Shows achievable privacy, in terms of minutes, as the number of interleaved polynomials (c) increases. Here we assume an epoch duration of 4 seconds, with 3 stalkers and non-stalker broadcasts equal to half those sent by a single stalker. Note that this graph does not consider the impact of field size or collisions in the x-coordinate.

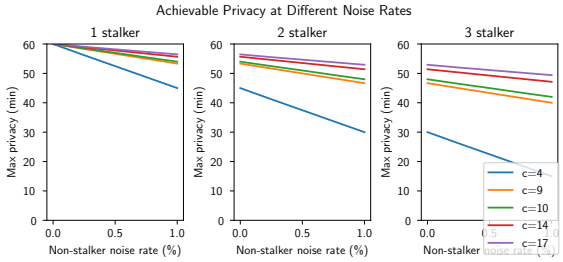

Figure 18: **Impact of interleaving RS polynomials (c).** Achievable privacy, in terms of minutes, for our construction (§4) using different values of c, assuming an epoch duration of 4 seconds. The horizontal axis shows the ephemeral noise rate measured as a percentage of a single LTA's output. Each point is computed using the CH*-MDSS bound of $t_{rec} \geq \frac{1}{c+1}(c \cdot t_{priv} + \mathsf{max})$. Note that this graph does not consider the impact of field size, or collisions in the x-coordinate.

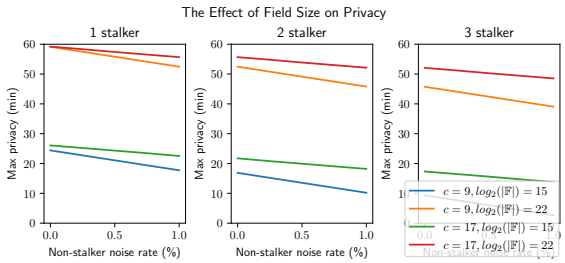

Figure 19: **Impact of collisions in the x coordinate.** Achievable privacy, in terms of minutes, for our construction (§4) for $c \in \{9, 17\}$ and field sizes $\mathbb{F}_p$ of 15 and 22 bits. This calculation assumes that polynomials are re-generated every 24 hours, and the epoch duration is 4 seconds.

G Extensions

Here we expand on the AROF extensions proposed in §2.

Adding forward secrecy. For simplicity in our proofs, the construction of §4 uses a single *fixed* key across all invocations of the Beacon algorithm. While this simplifies our presentation, in practical deployments this design may leave LTAs vulnerable to attacks in which an attacker compromises an LTA to learn past outputs. This attack can be prevented by including a mechanism to periodically update the secret key in an

irreversible manner (*e.g.,* by computing a new key using the output of a PRF), in a manner similar to the Apple FindMy protocol [29, 30].

Permissioned stalker detection. The protocols in this work assume a model of operation in which any victim device can execute stalker detection with no assistance from the service provider. A possible alternative design would involve the service provider (SP). In principle *permissioned* detection could allow service providers to authorize stalking-detection capability to legitimate users, while rejecting assistance to unauthorized tracking adversaries. Such capabilities can be achieved by encrypting each secret share under a key held by the service provider, and employing a two-party assisted decryption protocol to decrypt these values for victim devices. We leave the details to future work.

Malicious/counterfeit tags. Our constructions assume that stalker LTAs will execute the Beacon protocol honestly. This is reasonable when stalkers purchase unmodified manufacturer LTAs. However, as users become more familiar with anti-abuse countermeasures, attackers may opt to use counterfeit (or modified) tags that do not honestly execute the protocol. The primary risk in this setting is that a counterfeit LTA may improperly compute the secret share sent with each broadcast message. For example, a dishonest LTA can simply output a random value in this position, guaranteeing that a stalker tag will not be detected.

Mayberry *et al.* [36] proposed an extension to the Apple FindMy protocol that uses *blind signatures with auxiliary data* to pre-authenticate tag broadcast messages using a key held by the service provider. Briefly, this protocol requires the service provider to blindly sign each beacon message at provisioning time. LTAs then broadcast each messages and the corresponding signature: only signed data is relayed via the service provider, ensuring that counterfeit tags cannot use the offline finding network.

A similar approach can be used to ensure the validity of data generated by the Beacon algorithm in our construction of §4. This protocol would use a two-party signing procedure to (1) commit to the output of Beacon, (2) prove in zero-knowledge that the committed data (including secret shares) has been correctly formulated using the per-LTA secret key, and finally (3) obtain a blind signature from the service provider over the data itself. We defer development of the complete protocol to a future full version.

H A Window-based AROF construction

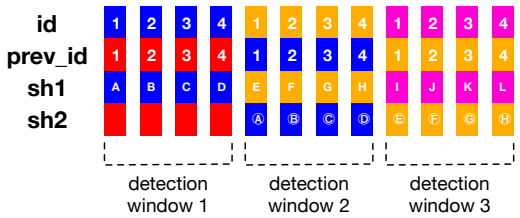

Figure 20: Illustration of broadcast identifiers in the window-based scheme for window size $\ell = 4$. id represents the identifier for the current epoch, and prev_id represents the identifier from ℓ epochs previous. For id and prev_id each unique color/number combination represent a distinct pseudonymous identifier. Circled/uncircled letters of the same color represent matching secret shares.

We describe here a simple window-based AROF construction. This construction preserves the unlinkability of LTA broadcasts against recipients who remain in the presence of an LTA for *for a well-defined and limited number of consecutive broadcasts.* Simultaneously it allows detection when a receiver is exposed to the tag for a longer period. The benefit of this scheme is its simplicity and ease of implementation. Unfortunately it offers relatively limited privacy guarantees, as we discuss further below.

Intuition. This scheme can be viewed as a simple extension of a basic FindMy-like protocol. The core idea is to augment the LTA to output an extra *auxiliary* pseudonymous identifier. Unlike the main identifier used in the FindMy scheme (which may be sent to the service provider), these auxiliary identifiers are used only by edge devices and are not relayed to the service provider. More critically, these additional identifiers are

repeated in the broadcast that occurs exactly ℓ epochs later. The value ℓ here defines a *stalker detection window*, and should be chosen carefully so that it provides privacy for a reasonable time period. The nature of these broadcasts is such that a receiver who remains within range of an LTA during any series of $\ell - 1$ epochs $(T, \ldots, T + (\ell - 1))$ should have no substantial advantage in linking the beacon advertisements sent by the LTA, as the identifiers are pseudorandom and do not repeat during this period. At the same time, a potential stalking victim who receives advertisements sent at any pair of epochs $(T, T + \ell)$ will immediately detect the repeated auxiliary identifier in the second broadcast, and can thus trace both signals to a single LTA. An illustration of the broadcast pattern is shown in Figure 20.

Of course, linking only two broadcasts is not sufficient to realize a full stalker-detection algorithm. A more complete detection procedure requires the victim to recover the tag ID, and also to consider the locations and pattern of many broadcasts sent during the detection window. To allow the victim to identify these intermediate broadcasts, we incorporate a second mechanism: each broadcast also includes one share of a key used to re-generate the full sequence of auxiliary identifiers used within the current detection window, as well as a matching share for the previous window. Given two such shares (*e.g.,* the broadcasts sent at epochs $T, T + \ell$) a victim can recover the key used to generate all of the identifiers in the window containing T. It can then use this key to re-generate and identify the intervening broadcasts. In a complete system this key can also be used to derive a tag identifier $\mathsf{id}_{\mathsf{tag}}$ for communicating with the LTA.