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

Gabrielle Beck* Harry Eldridge* 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 and devices via offline tracking networks. The tracking protocols have been designed to ensure some level of user privacy against surveillance by the vendor. Such privacy mechanisms, 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 weaken privacy guarantees in order to allow users to detect malicious stalker tags.

In this work we show how to achieve an improved trade-off between user privacy and stalker detection within the constraints of existing tracking protocols. We implement our new protocol using families of list-decodable error-correcting codes, and give efficient algorithms for stalker detection under realistic conditions.

## 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 [39].

**Privacy and stalking risks.** The widespread deployment of tracking 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 [47, 45] can easily monitor individuals' physical movements. In response to these privacy concerns, manufacturers deploy sophisticated countermeasures: for 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* [30], 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, 49, 23, 33, 15].

*Johns Hopkins University, {becgabri,hme,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 [32]:* | | | | | | |
| Near-owner mode | 15 min | 450 | ✗ | n/a | n/a | |
| Separated mode | 24 hrs | 43,200 | ● | n/a | ✗ | 15-60 min† |
| *This work (§4):* | | | | | | |
| 2-second epochs / 1-hour window | 2 sec | 1 | ● | 40 – 46 min* | ● | 60 min |
| 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.

**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, 17, 46, 32, 31]. 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. Unfortunately, the ability to detect stalking tags may conflict with anti-tracking countermeasures. To address this, manufacturers had to adopt 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 often at the cost of reducing privacy against tracking adversaries.[1] Apple and Google have jointly proposed an IETF draft [32] 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. Concretely, 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?*

In this paper 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.

**Contributions.** Specifically, our contributions include:

1. We formalize and construct *abuse-resistant private offline finding protocols* that allow both privacy against tracking adversaries and stalker detection by victim devices.

2. 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 multiple secrets

[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.

while achieving new properties of unlinkability and multi-dealer correctness. We provide efficient constructions of MDSS by combining variants of Shamir secret sharing [48] with lattice-based algorithms for list-decoding variants of Reed-Solomon codes.

3. We explore parameters for our constructions and demonstrate that existing tracking protocols can be made substantially more private using our protocols, without sacrificing stalker detection.

4. We experimentally collect data on tag broadcasts in various real-world settings to inform our parameter selection.

5. We propose several optimizations for realizing our protocols, including designing a new type of pseudorandom function called a *collision-aware PRF* that allows for the detection of PRF collisions without the need to keep large amounts of state.

6. Finally, we implement our algorithms and demonstrate that they are efficient in practice when running on modest hardware, even when configured to provide a strong degree of privacy against tracking adversaries.

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

## 1.1 Technical intuition

To prevent unauthorized tracking of LTAs, protocols such as Apple's FindMy [1] rotate the content of LTA transmissions periodically: each LTA contains a secret key that it uses to generate a sequence of pseudonymous identifiers,[2] where each identifier can be computed as a function of the key and a monotonically-increasing epoch counter [29]. An authorized owner who retains a copy of the secret key can re-derive this sequence and query the service provider on all identifiers to obtain location reports. Unauthorized users (such as tracking adversaries) should find it difficult to link the identifiers sent by an LTA across distinct epochs.[3]

While these changing identifiers provide a degree of privacy for users, they cause difficulty for stalker-detection algorithms. A potential stalking victim must determine whether an LTA has remained in close proximity for a pre-determined period of time, or over a series of distinct geographic locations. If an LTA rotates its identifier during this period, the resulting sequence of broadcasts may be difficult to link to a single emitting device. As previously noted, current anti-stalking proposals address this problem by *reducing the rate of identifier rotation* and maintaining a static identifier for many hours [32, 29].

This solution is suboptimal, even when only used in "separated mode", since users often carry LTAs outside the range of paired devices. Given the increasing sophistication of tracking solutions, LTA deployments should *increase* the rate of identifier rotation in the future to improve the privacy offered by these devices. Unfortunately, a recent attempt to standardize Apple's approach [32] may enshrine an arbitrary upper bound on the level of privacy that future tags can ever achieve, even if future hardware improvements allow for more frequent identifier rotation. More interestingly, the tradeoff Apple has adopted highlights a core challenge we address in this work: *the goal of a stalking victim is nearly identical to that of the tracking adversary.* It seems intrinsically difficult to develop systems that allow the former to succeed without providing a useful advantage to the latter.

[2] In the Find My protocol, these pseudonyms double as public encryption keys. This public key can be used by volunteer devices to encrypt their GPS location as part of the reporting process, ensuring that the network operator does not learn device locations.

[3] This approach may not eliminate every vector that could be used to track or link devices (for example, it does not address side channels such as timing, signal analysis or RF-hardware fingerprinting [24].) We consider these concerns to be out of scope for this discussion.

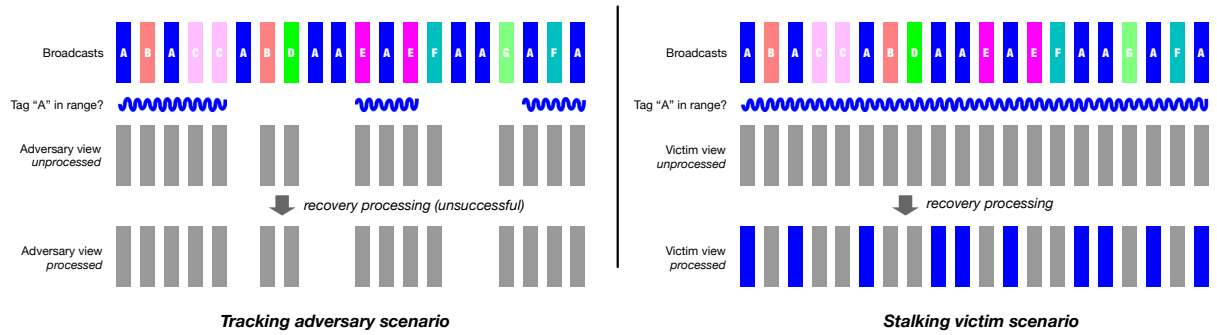


Figure 2: Illustration of two passive detection scenarios. **Left:** a static *tracking adversary* sees many broadcasts from different LTAs, including LTA "A" (blue). However, because "A" is mobile, the adversary is not in continuous range to receive its broadcasts and does not receive enough to enable 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: it can identify which broadcasts belong to LTA "A" and recover a password to communicate with the device.

**Separating stalking victims from tracking adversaries.** While tracking adversaries must solve a problem that is *similar* to that of a stalking victim, the two parties are not identical. Indeed, we can expect that in many instances tracking adversaries will be more limited in their access to broadcasts from a given LTA. For example, 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* private offline finding.

**Our approach.** Rather than broadcast a constant identifier, we propose to transmit frequently-changing and unlinkable broadcast identifiers as a means to maximize privacy. Unlike existing solutions, we structure our broadcasts so that a stalking victim can detect the stalker by *linking* the relevant series of broadcasts once the victim has observed "enough" broadcasts from the stalker, even in the presence of many non-stalker tags. However, to a tracking adversary who observes fewer broadcasts, the content of a single LTA's broadcasts will be *unlinkable*: that is, cryptographically indistinguishable from a series of broadcasts sent by many different tags.

Our goal is to construct a broadcast pattern that allows detection by a victim that has received (nearly) all broadcasts sent by an LTA during a pre-defined *detection window* time period. At the same time, we wish to ensure that tracking adversaries cannot link identifiers broadcast by an LTA unless they have also received a similarly-complete pattern of broadcasts.

**A simple solution.** As a warm-up, we first consider a construction that provides privacy against a tracking adversary who receives broadcasts within any $\ell$-sized set of *consecutive* broadcast periods, but allows identifiers to be linked by a victim who receives any transmissions immediately beyond this period: for example, any two broadcasts from periods $T, T + \ell$.

Such a solution can be easily realized in the following manner: each LTA simply constructs its identifiers sequence such that no identifier repeats during any cycle of $\ell$ consecutive broadcasts, and yet is guaranteed to contain one pair of repeated identifiers in any consecutive list of $\ell + 1$ broadcasts. We formalize such a scheme in Appendix §D. While this scheme achieves some notion of unlinkability, we note that the privacy

offered is quite fragile. A tracking adversary that encounters a given LTA at any broadcast epochs $T$ and $T+\ell$ (*i.e.,* that receives only two broadcasts from the LTA) can immediately link these broadcasts together, even if the LTA has not remained in proximity to the detector for the intervening $\ell - 1$ broadcasts.

This construction highlights subtle difficulties in even defining the property we want. We would like detection to imply that an LTA has been in *continuous proximity* to a given target, not merely that they have seen two related broadcasts in distant broadcast periods. More formally, we wish to ensure that a tracking adversary who receives an $\epsilon$-fraction of broadcasts 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.

**A tool: Multi-Dealer Secret Sharing.** To obtain a better solution, we instead use *secret sharing* [48]. In a classical secret sharing scheme, a single *dealer* shares a secret into multiple shares structured so that the original secret can be recovered from a subset of the shares. In our proposal each LTA periodically samples a secret tag identifier. Then, at each epoch, it will output a pseudonym similar to the one used in Apple's scheme, *along with a share of the secret.* By changing the pseudonym and using a different secret share at each epoch, the tag can thus preserve its privacy against tracking adversaries who receive only a fraction (or a limited absolute number) of the tag's emitted secret shares. At the same time, a stalking victim who receives a more complete set of shares will be able to recover the tag's identifier and communicate with the tag. The exact parameters to determine both privacy and the stalker-detection threshold are determined by the chosen parameters of the secret sharing scheme, as well as the broadcast traffic.

This proposal runs into several challenges. Broadcasts emitted by a stalking LTA will often be intermingled with broadcasts from other LTAs in the vicinity: this includes both ephemeral LTAs that a victim is only briefly in range of, as well as additional LTAs that remain persistently near the victim. Detecting a stalker in this setting requires robust secret recovery in the presence of *multiple dealers*, where these additional broadcasts will often comprise a majority of the broadcasts that any victim receives. In other uses of secret sharing [9, 21, 22] this problem is addressed by simply *identifying* (or labeling) all shares transmitted by a given dealer to identify which shares correspond to which set. In our work, *we cannot label these secret shares*, as shares must remain *unlinkable* until the victim receives a certain threshold.

We formally model this new setting for secret sharing as *multi-dealer secret sharing* (MDSS) and we show how to instantiate it for threshold access structures by combining variants of Shamir's secret sharing with efficient algorithms for list-decoding Reed-Solomon codes [44]. The novel aspects we address here are both theoretical and practical. From a theoretical perspective, we provide new definitions that capture *unlinkability* of shares in the setting of multiple dealers and properly define the necessary reconstruction properties. More practically, we require constructions that can efficiently reconstruct collections of *thousands of shares* from many dealers, even on memory-constrained devices such as smartphones. This requirement rules out the use of "out of the box" list decoding algorithms. Instead, we modify prior work on lattice-based decoding algorithms to enable simultaneous list decoding in seconds on hundreds or thousands of distinct shares per hour.

# 2 Preliminaries

**Notation and algorithms.** Let $\lambda$ be a security parameter. We use $\|$ to denote concatenation, and the notation $a \in_R A$ to indicate that $a$ is a uniformly sampled element from the set $A$. Our schemes will make use of pseudorandom functions $\mathsf{PRF}_i = (\mathsf{KeyGen}, \mathsf{Eval})$. We will define $\mathsf{CCA} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ to be a CCA-secure PKE scheme, and use $\mathsf{KeyGen}(1^\lambda; r \in \{0,1\}^\lambda)$ to indicate that the random coins for the key generation algorithm are provided explicitly.[4] Finally, we define $\mathcal{H}$ to be a collision-resistant hash function.

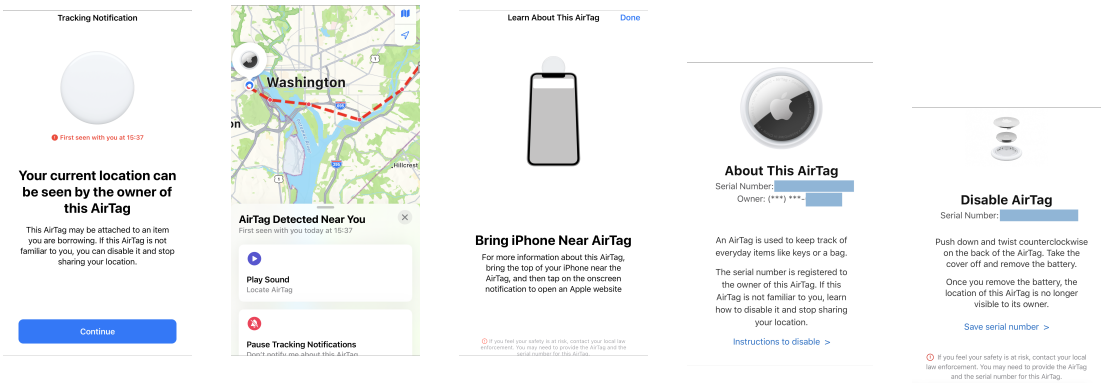


Figure 3: Apple FindMy alerts when an unrecognized AirTag is detected. The rightmost two images are presented on a website launched from the alert screen.

## 2.1 Offline finding networks

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 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, the typical setting for stalking attacks.

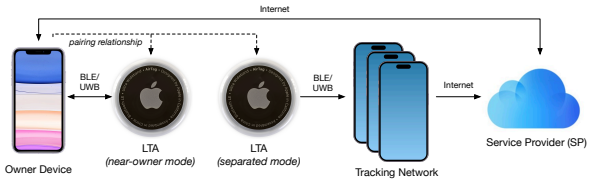


Figure 4: Components of a location-tracking network.

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 and 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.

## 2.2 Offline finding protocols

The devices in an offline finding network jointly conduct an *offline finding protocol*. Our protocols will implement both privacy (anti-tracking) and abuse-detection features. We now present high-level algorithmic definitions and security notions for a privacy-preserving abuse-resistant tracking protocol, using notation inspired by Mayberry *et al.* [34].

*A note on the security model.* 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.

[4]Note that in practice we can stretch $\{0,1\}^\lambda$ into a larger sequence using an appropriate PRG, so this requirement does not limit our choice of encryption schemes.

**Abuse-resistant Offline Finding Protocol**

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

$\mathsf{Beacon}(k_{\mathsf{tag}}, i_{\mathsf{epoch}}, \mathsf{aux}) \to B$ on input the tag key, the index of the current *anonymity epoch*, and auxiliary data $\mathsf{aux}$ (*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 $\mathsf{loc}$, it produces 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. $\mathsf{Owner}$ provides a tag key $k_{\mathsf{tag}}$ and an epoch $i_{\mathsf{epoch}}$ and $\mathsf{SP}$ uses a database $\mathcal{D}$. The output to $\mathsf{Owner}$ is potentially a list of reports and $\mathsf{SP}$'s output is $\perp$.

Figure 5: Algorithms for abuse-resistant offline finding 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 in §8.

**Definition 2.1.** An *abuse-resistant (privacy-preserving) offline-finding protocol* is a tuple of algorithms $(\mathsf{KeyGen}, \mathsf{Beacon}, \mathsf{GetTagID}, \mathsf{GenReport}, \mathsf{Detect})$ and a protocol $\mathsf{RetrieveReports}$ as specified in Figure 5.

To use an offline-finding protocol, each LTA executes the $\mathsf{KeyGen}$ algorithm with a set of deployment-specific parameters, and shares the resulting key with an owner device.[5] The LTA then initializes an *anonymity 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 $\mathsf{Beacon}$ algorithm on the current epoch counter and transmits the output one or more times.[6] Volunteer devices collect the resulting broadcasts and use the $\mathsf{GenReport}$ algorithm to generate encrypted 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 can pass all received LTA broadcasts to the $\mathsf{Detect}$ algorithm to detect the presence of nearby stalking LTAs.

**Correctness and security.** An abuse-resistant tracking scheme must satisfy several correctness and security properties as defined below.

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

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

[5]For simplicitly we have outlined a definition with a single static tag key: we note that that some constructions may also incorporate *forward-secrecy* requirements, where tag keys are "ratcheted" forward with each call to $\mathsf{Beacon}$. We discuss this extension in §8.

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

**Tag Detectability Experiment**

$Q := \emptyset; L := []; B_{\mathsf{list}} := []; \mathsf{st} = \bot$
$\forall j \in [1, o = poly(\lambda)]$ :
  $(\mathsf{id}, i, \mathsf{aux}, \mathsf{loc}, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{st}, \text{“query”})$
  If $(\mathsf{id}, *) \notin L$ :
    $k_{\mathsf{tag}} \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{cfg})$
    $L := L \cup \{(\mathsf{id}, k_{\mathsf{tag}})\}$
  find $k_{\mathsf{tag}}$ so that $(\mathsf{id}, k_{\mathsf{tag}}) \in L$
  $B \leftarrow \mathsf{Beacon}(k_{\mathsf{tag}}, i, \mathsf{aux})$
  $B_{\mathsf{list}} := B_{\mathsf{list}} \cup \{(B, \mathsf{loc})\}$
  $Q := Q \cup \{(\mathsf{id}, i)\}$
$\mathsf{out} \leftarrow \mathsf{Detect}(B_{\mathsf{list}})$
$b = 1$
$\forall (\mathsf{id}, k_{\mathsf{tag}}) \in L,$
  If $P'(\mathsf{cfg}, Q, \mathsf{id}) = 1,$
    $b' = (\exists i, (\mathsf{id}, i) \in Q \wedge$
      $\mathsf{GetTagID}(k_{\mathsf{tag}}, i) \in \mathsf{out})$
    $b = b \wedge b'$
return $b$

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

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

*Detectability.* A stalking LTA should be detectable by a victim. This detection should occur when a potential victim device sees a sufficient number (or pre-specified pattern) of beacon broadcasts from the stalking LTA. Because the pattern may differ between constructions, we capture this notion using a predicate $P'$ that is passed as a parameter to the experiment. If the received beacons implicate a particular LTA as a stalker, then an appropriate identifier from the LTA must appear in the output.

**Definition 2.3** (Detectability)**.** *A privacy preserving tracking protocol is detectable for predicate $P'$ if $\forall$ valid* $\mathsf{cfg}$ *values, $\forall n.u.p.p.t$ algorithms $\mathcal{A}$, $\exists$ a negligible function $negl(\lambda)$ so that* $\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Det},P'}(\lambda, \mathsf{cfg}) = 0] \leq negl(\lambda)$*, where* $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Det},P'}(\lambda, \mathsf{cfg})$ *appears in Figure 6.*

*Tag indistinguishability.* A tracking adversary who receives fewer broadcasts (or a different broadcast pattern) than that of a stalking victim should not be able to distinguish a given LTA's broadcasts from those of other LTAs. In order to allow maximal flexibility for different stalker-detection mechanisms, our definitions capture the latter condition via a predicate $P$ that is provided as a parameter to our experiment: this function evaluates the broadcast pattern that a receiver obtains, and outputs 1 if and only if indistinguishability should hold over the pattern. We provide the formal definition below.

**Tag Indistinguishability Experiment**

$Q := \emptyset; \mathsf{st} = \bot$

$\forall i \in \{0,1\}$ :

$\quad k^i_{\mathsf{tag}} \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{cfg})$

$\forall j \in [1, o = poly(\lambda)]$ :

$\quad (\mathsf{id}, i, \mathsf{aux}, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{st}, \text{“}\texttt{query}\text{”})$

$\quad Q := Q \cup \{(\mathsf{id}, i)\}$

$\quad B \leftarrow \mathsf{Beacon}(k^{\mathsf{id}}_{\mathsf{tag}}, i, \mathsf{aux})$

$\quad \mathsf{st} \leftarrow \mathcal{A}(\mathsf{st}, \text{“}\texttt{query response}\text{”}, B)$

$(i^*_0, i^*_1, \mathsf{aux}, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{st}, \text{“}\texttt{challenge}\text{”})$

$Q := Q \cup \{(0, i^*_0), (1, i^*_1)\}$

$B^* \leftarrow \mathsf{Beacon}(k^b_{\mathsf{tag}}, i^*_b, \mathsf{aux})$

$\hat{b} \leftarrow \mathcal{A}(\mathsf{st}, \text{“}\texttt{challenge response}\text{”}, B^*)$

If $P(\mathsf{cfg}, Q) = 1$ output $\hat{b}$, else output 0

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

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

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

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

We note that the game presented in Figure 7 only guarantees indistinguishability when aux is the *same* across both challenge broadcasts. A stronger security definition might allow the adversary to choose different aux values for each tag key, provided they have the same length. Although we do not present this solution in our protocols (and it does not appear to be implemented in existing deployments) our schemes could satisfy such a definition by changing the Beacon algorithm to encrypt aux using a symmetric-key encryption scheme that achieves key privacy.

*Location indistinguishability.* Mayberry *et al.* [34] identify several additional properties that must be present in an offline finding system. For example, it must be the case that the service provider cannot determine the location of LTAs or volunteer devices from the encrypted location reports it receives. Our schemes in §4 achieves this property in much the same way as the constructions of [34]. Since this paper is mainly concerned with the interaction between stalking tags and victims we omit this analysis here.

# 3 Multi-Dealer Secret Sharing

In a secret sharing scheme [48], a dealer divides a secret into *shares* such that a subset of the shares can be used to recover the initial secret. In this work, we consider a setting where *multiple dealers* may emit shares of different secrets, such that a receiver can recover these secrets from a set of mixed shares. We investigate new security and functionality requirements for this setting.

- *Unlinkability:* This property requires that given a set of shares, an adversary cannot determine whether two or more shares are associated with the same secret (or dealer). Since it is always possible to link shares via reconstruction, we require that the share set does not contain too many shares of the same secret. We emphasize that unlinkability is a stronger property than the standard notion of *privacy* for

secret sharing which only guarantees that an adversary cannot recover the secret without obtaining a sufficient set of shares.

- *MD-Correctness:* This property requires that, given shares for one or more secrets, a receiver can reconstruct *all* secrets for which they have received a sufficient number of shares.

In this work, we study multi-dealer secret sharing schemes 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 dealers the scheme is able to tolerate.
- $\mathsf{max}$, the maximum number of shares that can be input to the reconstruction algorithm.

We consider a setting analogous to the *ramp* setting of Blakley and Meadows [11], where we allow for a gap between $t_{rec}$ and $t_{priv}$ larger than one. Further, we note that our notion of multi-dealer secret sharing has connections with robust secret sharing [43] which we discuss in §9. 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 $\mathcal{S}$ and an index space $\mathcal{I}$, and consists of the following PPT algorithms:

- $\mathsf{Share}(s, \mathcal{I}'; r) \to \{sh_i\}_{i \in \mathcal{I}'}$, takes as input a secret $s \in \mathcal{S}$, a set of indices $\mathcal{I}' \subseteq \mathcal{I}$, and some randomness $r$, and outputs a set of shares $\{sh_i\}_{i \in \mathcal{I}'}$.
- $\mathsf{Reconstruct}(\{sh_1, \ldots, sh_w\}) \to \{s_1, \ldots, s_m\}$, 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. For convenience we will sometimes use a stateful variant of the $\mathsf{Share}$ algorithm with different syntax. For a single index $i \in \mathcal{I}$, the stateful algorithm $\mathsf{Share}(s, i; r)$ outputs only $sh_i$, the share of $s$ at index $i$.

A multi-dealer secret sharing scheme must satisfy three properties: *privacy*, *unlinkability*, and *MD-Correctness*. These properties are defined below.

**Definition 3.2** (Privacy)**.** *We say that a $(t_{rec}, t_{priv}, d, \mathsf{max})$-MDSS with secret space $\mathcal{S}$ and index space $\mathcal{I}$ is* $t_{priv}$-private *if for any $s, s' \in \mathcal{S}$ and $\mathcal{I}' \subseteq \mathcal{I}$ such that $|\mathcal{I}'| \leq t_{priv}$:*

$$\mathsf{Share}_{\mathcal{I}'}(s) \approx \mathsf{Share}_{\mathcal{I}'}(s')$$

**Unlinkability.** Notions of *unlinkability* appear in the cryptography literature in the context of primitives such as ring signatures and anonymous credentials [20, 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 an *a priori* bounded number ($t_{priv}$) of shares of a secret.

**Definition 3.3** (Unlinkability)**.** *We say that a $(t_{rec}, t_{priv}, d, \mathsf{max})$-MDSS with secret space $\mathcal{S}$ and index space $\mathcal{I}$ is $t_{priv}$-*unlinkable *if for all adversaries $\mathcal{A}$,*

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

where $\mathsf{ExpLink}^b_{\mathcal{A}}$ is described in Figure 8.

*Unlinkability v.s. Privacy.* $t_{priv}$-unlinkability is a strictly stronger notion than $t_{priv}$-privacy. Specifically, while $t_{priv}$-unlinkability can be seen to imply $t_{priv}$-privacy via a straightforward hybrid argument, the reverse is not true. Consider the following modification to a private secret sharing scheme: to share a secret, first

**MDSS Unlinkability Experiment**

$r_0, r_1 \in_R \{0,1\}^\lambda; L := \emptyset$
$(s_0, \mathsf{st}) \leftarrow \mathcal{A}(\text{“}\texttt{init}\text{”})$
$\forall j \in [t_{priv} - 1]:$
  $(i, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{st}, \text{“}\texttt{query}\text{”})$
  $L := L \cup \{i\}$
  $\mathsf{st} \leftarrow \mathcal{A}(\mathsf{st}, \text{“}\texttt{query response}\text{”}, \mathsf{Share}(s_0, i; r_0))$
$(s_1, i_0^*, i_1^*, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{st}, \text{“}\texttt{challenge}\text{”})$
If $i_0^* \in L$: end and output 0
$\hat{b} \leftarrow \mathcal{A}(\mathsf{st}, \text{“}\texttt{challenge response}\text{”}, \mathsf{Share}(s_b, i_b^*; r_b))$
Output $\hat{b}$

Figure 8: Experiment $\mathsf{ExpLink}_{\mathcal{A}}^{b}(\lambda)$

sample a random value $r$ and then append $r$ to each share computed using the private secret sharing scheme. Clearly this modified scheme is still private. However, the shares of a secret are trivially linkable by comparing their suffixes.

**MD-Correctness.** To define MD-Correctness we require that the Reconstruct algorithms outputs all the secrets shared by *sufficient* dealers, while handling "noise" shares output by *insufficient* ones. Additionally, we consider the setting where $t_{rec}$ shares of a secret may *not* be sufficient for reconstruction with some probability $\epsilon$. Our correctness requirement stipulates that Reconstruct should output all secrets $s$ for which there are at least $t_{rec}$ shares in the input, *and* for which $s$ could be reconstructed from *just* those $t_{rec}$ shares. As this is equivalent to requiring that a secret can be reconstructed under ideal conditions, we refer to this "noiseless" reconstruction algorithm as IdealReconstruct.

We use IdealReconstruct to define the idea of *consistency*. We say that a secret $s$ is *consistent* with a set of shares if at least $t_{rec}$ of the shares could have been output by a sharing of $s$, and those $t_{rec}$ shares are sufficient for reconstructing $s$. Similarly, we define the *consistent set* for a set of shares to be the set of all consistent secrets.

**Definition 3.4** (Consistent Set)**.** *For a set of shares* $\{sh_1, \ldots, sh_n\}$ *and an ideal reconstruction algorithm* IdealReconstruct*,*

$$\begin{aligned}&\mathsf{Consistent}(\{\mathsf{sh}_1, \ldots, \mathsf{sh}_\mathsf{n}\}) = \\ &\quad \{s | \exists r, \mathcal{I}' \ s.t. \ |\mathsf{Share}(s, \mathcal{I}'; r) \cap \{sh_1, \ldots, sh_n\}| \geq t_{rec} \\ &\quad \ and \ \mathsf{IdealReconstruct}(\mathsf{Share}(s, \mathcal{I}'; r)) = s\}\end{aligned}$$

We use this definition of consistency in two ways: first, for a game representing the *honest* case, in which all shares given to the Reconstruct algorithm are generated by Share, and second in the stronger *adversarial* case, where the shares are generated arbitrarily. We can now formally define our notions of correctness.

**Definition 3.5** (Weak-MD-Correctness)**.** *We say that a* $(t_{rec}, t_{priv}, d, \mathsf{max})$*-MDSS is* $\epsilon$-weakly-correct *with respect to an ideal reconstruction algorithm* IdealReconstruct *if for all sets* $\mathsf{DS} = \{(s_1, \mathcal{I}_1), \ldots, (s_d, \mathcal{I}_d)\}$ *and* $\mathsf{NS} = \{(s'_1, \mathcal{I}'_1), \ldots, (s'_m, \mathcal{I}'_m)\}$ *where* $\forall j \in [d]$*,* $|\mathcal{I}_j| \geq t_{rec}$ *and* $\forall k \in [m], |\mathcal{I}'_k| < t_{rec}$ *and* $\sum_{j \in [d]} |\mathcal{I}_j| + \sum_{k \in [m]} |\mathcal{I}_m| \leq \mathsf{max}$*:*

$$\Pr[\mathsf{ExpWCor}_{\mathsf{DS},\mathsf{NS}}(\lambda) = 1] \leq negl(\lambda)$$

*and for all* $s \in \mathcal{S}$ *and sets* $\mathcal{I}' \subset \mathcal{I}$ *where* $|\mathcal{I}'| = t_{rec}$*:*

$$\Pr[\mathsf{IdealReconstruct}(\mathsf{Share}(s, \mathcal{I}')) \neq s] \leq \epsilon$$

**Weak MD-Correctness Experiment**

$\mathsf{S} := \emptyset$
$\forall (s, \mathcal{I}') \in \mathsf{DS} \cup \mathsf{NS}$ :
$\quad \mathsf{S} = \mathsf{S} \cup \mathsf{Share}(s, \mathcal{I}')$
If $\mathsf{Reconstruct}(\mathsf{S}) \neq \mathsf{Consistent}(\mathsf{S})$,
output 1, else output 0

Figure 9: Experiment $\mathsf{ExpWCor}_{\mathsf{DS},\mathsf{NS}}(\lambda)$

**Strong MD-Correctness Experiment**

$\{sh_1, \ldots, sh_{\mathsf{max}}\} \leftarrow \mathcal{A}()$
If $\mathsf{Reconstruct}(\{sh_1, \ldots, sh_{\mathsf{max}}\}) \neq$
$\mathsf{Consistent}(\{\mathsf{sh}_1, \ldots, \mathsf{sh}_{\mathsf{max}}\})$, output 1, else output 0

Figure 10: Experiment $\mathsf{ExpSCor}_{\mathcal{A}}(\lambda)$

*Where* $\mathsf{ExpWCor}_{\mathsf{DS},\mathsf{NS}}(\lambda)$ *is defined in Figure 9.*

**Definition 3.6** (Strong-MD-Correctness)**.** *We say that a* $(t_{rec}, t_{priv}, d, \mathsf{max})$*-MDSS is* $\epsilon$-strongly-correct *with respect to an ideal reconstruction algorithm* $\mathsf{IdealReconstruct}$ *if for all adversaries* $\mathcal{A}$*:*

$$\Pr[\mathsf{ExpSCor}_{\mathcal{A}}(\lambda) = 1] = 0$$

*and for all* $s \in \mathcal{S}$ *and sets* $\mathcal{I}' \subset \mathcal{I}$ *where* $|\mathcal{I}'| = t_{rec}$*:*

$$\Pr[\mathsf{IdealReconstruct}(\mathsf{Share}(s, \mathcal{I}')) \neq s] \leq \epsilon$$

*Where* $\mathsf{ExpSCor}_{\mathcal{A}}(\lambda)$ *is defined in Figure 10.*

## 3.1 Constructing MDSS

A natural choice for constructing MDSS begins with the secret-sharing construction of Shamir [48]. In Shamir's scheme, 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 $p(0) = s$. A secret share is a pair $(r, p(r))$ where $r \in \mathbb{F} \setminus 0$ is a chosen evaluation point.

**Achieving unlinkability.** In typical uses of Shamir's secret sharing, the evaluation points are set to fixed numbers, say 1 to $n$ for generating $n$ shares. This, however, violates unlinkability: given a set of shares, any pair of shares with the same evaluation point $r$ must correspond to different secrets. To achieve unlinkability, we therefore deviate from this approach. Specifically, we sample each evaluation point uniformly at random from $\mathbb{F}$. If $\mathbb{F}$ has size super-polynomial in the security parameter $\lambda$, any two evaluation points collide with probability negligible in $\lambda$. Crucially, this probability is the same for any two shares, irrespective of whether they correspond to the same or different secrets.

**Achieving MD-correctness.** We observe that a collection of Shamir secret shares can be viewed as a Reed-Solomon codeword [36]. Recovering a secret from a collection of Shamir secret shares in the presence of noise is equivalent to the problem of Reed-Solomon decoding.

Given this, a natural choice for the reconstruction algorithm is to use a Reed-Solomon decoding algorithm. Unfortunately, the classical bounds for decoding Reed-Solomon codes leave a large gap between the number of shares required to recover a secret and the privacy threshold in the presence of realistic error rates. These

**Algorithm 1:** $\mathsf{CH}^*$

**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**
**if** *there are multiple vectors with the shortest length and the shortest length* $\leq \lambda$ **then**
  **return** Multiple
**else**
  **return** None
**end**

Figure 11: The $\mathsf{CH}^*$ sub-algorithm of our $\mathsf{CH}^*$-MDSS inputs a set of evaluation points and parameters, and either returns a set of polynomials $p_i \in \mathbb{F}[x]$ matching the evaluation points or returns a flag indicating that there are no solutions or multiple matching solutions. The matrix $M$ is a lattice basis given in row vector format. $\lambda$ is an upper bound on the length of our target vector.

bounds can be improved somewhat by using the list-decoding algorithm of Guruswami and Sudan [26]. While this construction achieves fully rigorous *strong-md-correctness*, the running time and memory requirements scale poorly, and the asymptotic bounds still leave a large gap.

**Our construction: Multiple polynomial evaluations.** In order to achieve a better tradeoff and more efficient decoding algorithms, we use a construction inspired by interleaved Reed-Solomon (IRS) codes [19, 12]. 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 $(r, p_1(r), \ldots, p_c(r))$.

There are numerous decoding algorithms for interleaved Reed-Solomon codes and other closely related noisy curve reconstruction problems in the literature [42, 52]. Unfortunately, many of these algorithms are not efficient enough (both in terms of RAM and computation) to operate on limited devices at the parameter sets we require. To address this, we adopt an IRS decoder based on a dual form of a lattice-based decoding algorithm proposed by Cohn and Heninger [18], which we call $\mathsf{CH}^*$-MDSS. We describe this decoder in detail below.

**Polynomial lattice preliminaries.** See [18] 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*

**Algorithm 2:** CH*-MDSS Construction

**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 $\perp$

```
1  solns := [], ws := [n], fail := False
2  while |ws| ≥ t and not fail do
3      res ← CH*(k + (|ws| − t), k, n, {(α_i, β_{i,1} ... β_{i,c})}_{i∈ws}).
4      if res = None then
5          return solns
6      else if res = Multiple then
7          s ← ⌊|ws| / t⌋, found ← False
8          while not found: do
               // randomly remove points until a single solution is found
9              pts ← ws.pop(2(s − 1))
10             res ← CH*(k + (|ws| − t), k, n, {(α_i, β_{i,1} ... β_{i,c})}_{i∈ws})
11             ws ← ws ∪ pts
12             if res can be parsed as (p_1 ... p_c) then
13                 found ← True
14                 add (p_1 ... p_c) to solns, remove agreeing points from ws
15             end
16         endw
17     else
18         parse res as (p_1 ... p_c), add (p_1 ... p_c) to solns, remove agreeing points from ws
19     end
20 endw
21 return solns
```

Figure 12: Our CH*-MDSS construction. We iteratively apply the CH* algorithm to recover solution vectors with the largest number of agreeing points in our sample until there are no more solutions. If the CH* algorithm indicates that there are multiple valid solutions, we randomly remove coordinates until the algorithm succeeds. pop($x$) randomly removes and returns $x$ values from a set.

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 $B$ [38]. 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. 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 description of CH*-MDSS.** We now provide a brief description of our lattice-based recovery algorithm. Throughout this section, we will say an ordered set of polynomials $(p_1 \ldots p_c) \in \mathbb{F}[z]^c$ *agrees with* a point $(\alpha, \beta_1, \ldots, \beta_c) \in \mathbb{F}^{c+1}$ if $\forall j \in [c], \beta_j = p_j(\alpha)$. 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. The main intuition behind our technique is to use the linear variant of [18] (with degree and multiplicity one) but instead of looking through the coefficient lattice, we use the dual lattice.

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.

One downside of the Cohn-Heninger algorithm is the need to extract many short vectors from the reduced basis and solve a system of polynomials. This incurs additional computational cost.

We observe that a target vector corresponding to any solution $p_1 \ldots p_c$ 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 11; we call this algorithm CH*. The lattice basis $M$ constructed in Figure 11 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.

First, we make a correctness claim regarding when the algorithm succeeds when given input corresponding to at most one solution with a bounded amount of random noise. This claim is heuristic based on the conjecture that the lattice $M$ from Figure 11 behaves like a random lattice with a planted target vector corresponding to the solution.

**Heuristic Assumption 1.** *The non-target vectors in lattice $M$ as described in Figure 11 behave like a random lattice when $M$ is instantiated from evaluations $\{(\alpha_i, \beta_{i,1}, \ldots, \beta_{i,c})\}_{i=1}^{n}$ of Reed-Solomon codes with distinct $\alpha_i$ and errors are chosen uniformly at random.*

**Claim 3.0.1.** *Under Heuristic Assumption 1, instantiating Figure11 with target vector length $\lambda = \frac{1}{c+1}(k + cn) - 1$ will recover, if it exists, a solution set $(p_1 \ldots p_c)$ 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 $|v_p| = k + (n - t)$[7], and

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

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

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 3.0.1 guarantees that the length of the target vector, $\lambda$, 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 11 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 the short vectors equaled our target vector, they did contain information that could be used to find it: the construction presented in Appendix F utilizes this information to achieve a slightly better bound[8].

Using Claim 3.0.1 and the preceding empirical results, the construction in Figure 11 can achieve $t \geq \frac{1}{c+1}(ck+n) + \frac{c}{c+1}$, when setting $\lambda = \frac{1}{c+1}(c(n-1)+k)$. We experimentally verified this version of the algorithm on 10,000 randomly selected inputs for various configurations of $n, k, t$ and $c$ and observed that it succeeds in practice for these non-adversarially constructed cases where this is at most one unique solution and bounded random noise.

**Handling additional solutions** The approach given in the algorithm we call $\mathsf{CH}^*$ works when there is exactly one solution set. We need to extend the above algorithm to cases where there are multiple valid solutions, if we want an $MDSS$ construction tolerating $d > 1$ dealers. The "natural" way to recover multiple solutions in this approach would be to examine the dual of the lattice corresponding to the higher degree and multiplicity construction given in [18]. Unfortunately, increasing these parameters results in an exponential increase in the lattice dimension, and the corresponding algorithm running times quickly grow too large for our application.

Experimentally, we observe that the $\mathsf{CH}^*$ algorithm with $\lambda$ set to $\frac{1}{c+1}(c(k+1)+n$ can recover a single solution set even when the input distribution includes multiple solution sets, as long as one set agrees with two more evaluation points than any other set. In particular, the shortest vector corresponds to the solution with the *most* agreeing points. We can then remove the points associated with this set from the original input, and run the algorithm again to find another solution, provided that it too has two more agreeing points than any solution sets left. This idea, extended to recover all solutions, is formalized below.

**Experimental Observation 3.0.1.** *Let $\{(\alpha_i, \beta_{i,1}, \ldots, \beta_{i,c})\}_{i=1}^{n}$ be a set of input points with distinct $\alpha_i$, such that there exists polynomial sets $P_1 \ldots P_h$ where $\forall \ell \in [h]$, $P_\ell = (p_{\ell,1}, \ldots p_{\ell,c}) \in \mathbb{F}[z]^c$, $\forall j \in [c], \deg(p_{\ell,j}) \leq k$, and $\exists T_\ell \subset [n], |T_\ell| \geq \frac{1}{c+1}(c(k+1)+n)$ s.t. $\forall w \in T_\ell, \forall j \in [c], p_{\ell,j}(\alpha_w) = \beta_{w,j}$. Let $S = [n] \setminus \cup_{i=1}^{h} T_i$. If it is the case that for all pairs $T_i, T_j$, the difference between $|T_i|$ and $|T_j|$ is at least 2, $S$ consists of input points that are random and those which correspond to a polynomial set $P = (p_1 \ldots p_c)$ with less than $\frac{1}{c+1}(c(k+1)+n)$ agreeing input points, and all polynomial sets $P$ are chosen independently at random, then repeated application of the algorithm in Figure 11, with removal of points corresponding to each found $P_i$, recovers $P_1, \ldots P_h$ with high probability.*

Unfortunately, when two input solution sets have numbers of agreeing points that differed by one or zero points, the reduced lattice basis did not contain any target vectors. Interestingly, the *lengths* of the vectors in the reduced basis give us a method of distinguishing from the case of no solutions: in the multiple solution case, the shortest vector in the reduced basis has a length less than our target vector length of $k + (n-t)$. Recall that when no solution exists, the shortest vector length is the floor of the bound in Equation 1.

We have two approaches to recover solutions in this aberrant case of nearly equal solution sets:

1. The simplest method is to delete points at random and re-try the decoding algorithm until it succeeds; we expect to re-try the algorithm a small number of times until one solution has two more points

[8] In particular, the algorithm achieves $t \geq \frac{1}{c+1}(ck+n)$

**MDSS Construction**

$\underline{\mathsf{Share}(s, \mathcal{I}') :}$

Sample $c$ polynomials $p_1, \ldots, p_c$, where $s = p_1(0)||...||p_c(0)$

Sample $|\mathcal{I}'|$ field elements $x_1, \ldots, x_{|\mathcal{I}'|} \xleftarrow{\$} \mathbb{F}$

return $\{(x_i, p_1(x_i), \ldots, p_c(x_i))\}_{i \in \mathcal{I}'}$

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

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

Figure 13: Our MDSS construction. A share consists of a random evaluation point $x_i$ together with evaluations of a collection of $c$ randomly generated polynomials at that point.

than any other solution set. This random deletion algorithm is probabilistic polynomial time and the running time varies depending on how many solution sets must be recovered. We have tested this algorithm on more than 20,000 instances sampled from distributions corresponding to up to three valid polynomial solution sets for a number of different $n, k, t$ and $c$ values where each input has distinct x-coordinates. Throughout these tests, we have had no failures. This is the method we use for our experimental timings.

2. A second method recovers a single solution as a short polynomial linear combination of vectors in our reduced basis or by applying a second lattice reduction; we describe this method in Appendix F. This approach is more efficient since it requires fewer lattice reductions and achieves a better bound than the randomized deletion algorithm. We tested this approach on 5,000 inputs sampled from distributions corresponding to up to five valid polynomial solution sets with no failures.

We put together the above techniques in the algorithm described in Figure 12 to solve the MDSS problem. We repeatedly apply the CH* algorithm to the collected set of points. This algorithm will either return a valid solution vector, indicate that there are multiple solutions, or indicate that there are no valid solutions. If there are multiple solutions, we use one of the above methods to recover the solutions.

The following claim summarizes the bounds for which we expect the algorithm to work for randomly constructed problem instances.

**Claim 3.0.2.** *Let $\{(\alpha_i, \beta_{i,1}, \ldots, \beta_{i,c})\}_{i=1}^n$ be a set of input points with distinct $\alpha_i$, such that there exists polynomial sets $P_1 \ldots P_h$ where $\forall \ell \in [h]$, $P_\ell = (p_{\ell,1}, \ldots p_{\ell,c}) \in \mathbb{F}[z]^c$, $\forall j \in [c], \deg(p_{\ell,j}) \leq k$, and $\exists T_\ell \subset [n], |T_\ell| \geq \frac{1}{c+1}(c(k+1)+n)$ s.t. $\forall w \in T_\ell, \forall j \in [c], p_{\ell,j}(\alpha_w) = \beta_{w,j}$. Let $S = [n] \setminus \cup_{i=1}^h T_i$. If $S$ consists of input points that are random and those which correspond to a polynomial set $P = (p_1 \ldots p_c)$ with less than $\frac{1}{c+1}(c(k+1)+n)$ agreeing input points and all polynomial sets $P$ are chosen independently at random, then* $\mathsf{CH}^*\text{-}\mathsf{MDSS}$ *recovers $P_1, \ldots P_h$ with high probability.*

**On adversarial channels**. As one final note, the channels that we consider in our setting are not fully adversarial. For one, x-coordinates must be unique as our algorithm only accomplishes the task of list-decoding and not list-recovery. But even in this more limited setting, we only consider input distributions where other polynomials are chosen uniformly at random and independent of other input. We do not, for example, handle input points that correspond to multiple solution sets. This happens exceedingly rarely when polynomial sets are chosen uniformly and independently of one another, but we have no such assurances if the channel is fully adversarial. We leave the exploration of what can be achieved under these types of channels to future work.

Putting all the previously described pieces together brings us to our main construction, which can be seen in Figure 13.

**Claim 3.0.3.** *For all $c \geq 1$ and finite fields $\mathbb{F}$ such that $|\mathbb{F}| \approx 2^\lambda$, construction 13 satisfies* unlinkability.

The proof of Claim 3.0.3 is a straightforward extension of the proof of privacy for traditional Shamir sharing, and can be found in Appendix B.

**Claim 3.0.4.** *For all sets of parameters satisfying* $t_{rec} \geq \frac{1}{c+1}(\mathsf{max} + c \cdot (t_{priv} + 1))$ *and finite fields* $\mathbb{F}$ *such that* $|\mathbb{F}| \approx 2^{\lambda}$*, construction 13 heuristically satisfies* $\epsilon$-weak-md-correctness *for a negligible* $\epsilon$*.*

**Heuristic reconstruction.** Our reconstruction algorithm $\mathsf{CH}^*\text{-}\mathsf{MDSS}$ is only heuristic. We experimentally validated its performance for random noise and develop algorithmic approaches that experimentally cope well with adversarial noise patterns.

**Using 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. Simply avoiding the repeated evaluation points does not address this problem, since *the absence* of collisions also weakens unlinkability. To address this concern, we 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.[9]

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

*List-Decodable Reed-Solomon Codes.* There is rich existing literature on list-decoding of Reed-Solomon codes [19, 41, 25, 26]. Some of these codes lead to constructions that have a non-optimal relationship between $t_{rec}$ and $t_{priv}$ [19, 26] or only perform well for unlikely ambient noise regimes [41]. In particular, for $c = 1$—which is simply Reed-Solomon encoding—we know of *no* explicit, *efficient* decoders beyond the Johnson radius.[10] Reed Solomon codes that achieve this bound [26] can be used in an MDSS construction but the Johnson bound imposes a large gap between $t_{priv}$ and $t_{rec}$ where $t_{rec} > \sqrt{t_{priv}\mathsf{max}}$. Note that our preference is to construct $MDSS$ schemes where $t_{rec} > t_{priv}$. Different variants of IRS codes have been proposed that have smaller gaps between these parameters but have their own problematic elements [41, 19, 25]. Parvaresh-Vardy codes [41]are only an improvement over [26] when at very low message rates (which translates in our setting to situations where $\mathsf{max} \gg t_{priv}$). Coppersmith-Sudan [19] is inefficient at optimum parameter sets, and even at optimum can at best achieve $t_{rec} > 2t_{priv}$. It also does not handle multiple dealers. In theory [25] could work well for our setting, but it is also not concretely efficient for optimum values of $t_{rec}$ and $t_{priv}$. There are currently no public existing implementations of their decoding algorithm either, to the best of our knowledge. The works of [18] and [22] suggest a possible efficient lattice-based decoder for general IRS codes but neither considers the equivalent of a multi-dealer setting.

*Insertion/deletion Codes.* Finally, we considered other codes that may have properties that are desirable in our setting. The channel we consider in this work is one that injects and deletes symbols at random locations rather than substituting existing symbols (which is the normal setting for Reed-Solomon codes). Codes exist in the insertion/deletion setting which can tolerate any polynomially bounded number of insertion errors [27]. Unfortunately, current constructions of these codes 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.

[9] We show unlinkability is not compromised for this change in Appendix B.2, as well has how to select parameters to achieve a target reconstruction success rate in Appendix G. If deployment parameters are chosen so that the number of such collisions is small, this will have only a modest effect on efficiency.

[10] The Johnson radius is a bound that every code can be list-decoded up to while retaining small (polynomial) list size. Some codes can be decoded efficiently beyond this bound.

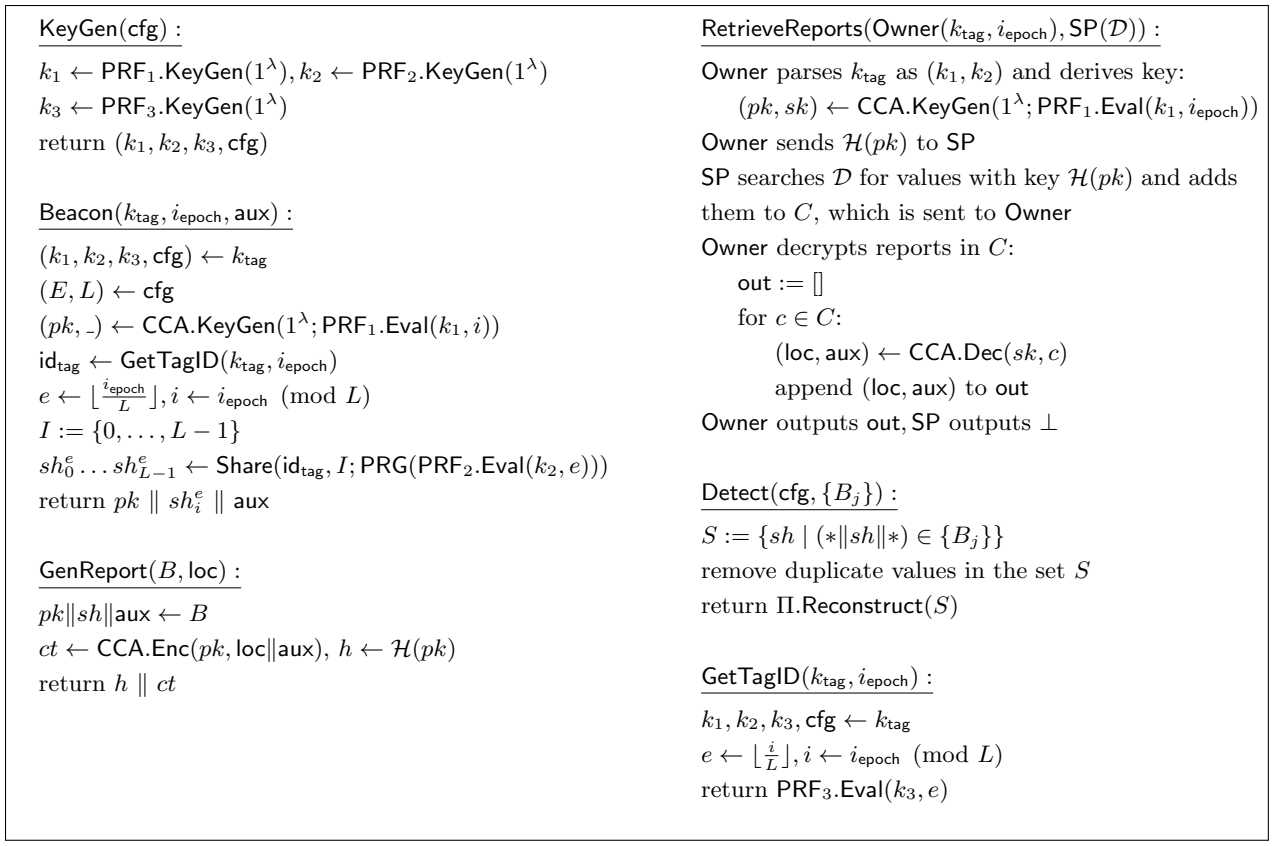


Figure 14: Main construction for abuse-resistant offline finding. This protocol assumes the existence of three pseudorandom functions $\mathsf{PRF}_i$ for $i \in [3]$ where the co-domain of $\mathsf{PRF}_1.\mathsf{Eval}$ and $\mathsf{PRF}_2.\mathsf{Eval}$ is $\{0,1\}^\lambda$ and the codomain of $\mathsf{PRF}_3.\mathsf{Eval}$ 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 $\Pi$ a $(t_{rec}, t_{priv}, d, \mathsf{max})$-MDSS sharing scheme.

# 4 Abuse-resistant Private Offline Finding

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

The core algorithms of our protocol are presented in Figure 14. A good deal of the complexity present is due to FindMy [11]: this includes any use of $k_1$ and the CCA-secure encryption scheme. The RetrieveReports algorithm is very close to the original one presented by Apple. The sole purpose of $k_2$ and $k_3$ is to prevent our construction from being stateful. In practice, one may wish to securely sample and cache the current $\mathsf{id}_{\mathsf{tag}}$ and the shares $sh_0 \ldots sh_{L-1}$.

**Achieving stalker detection.** To enable stalker detection, our protocol makes several critical additions to the basic protocol described above. First, each LTA maintains a *detection period* consisting of $L$ consecutive time epochs. At the start of each detection 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.

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. To do this, each device maintains

[11] See Appendix A for a more in-depth description of FindMy

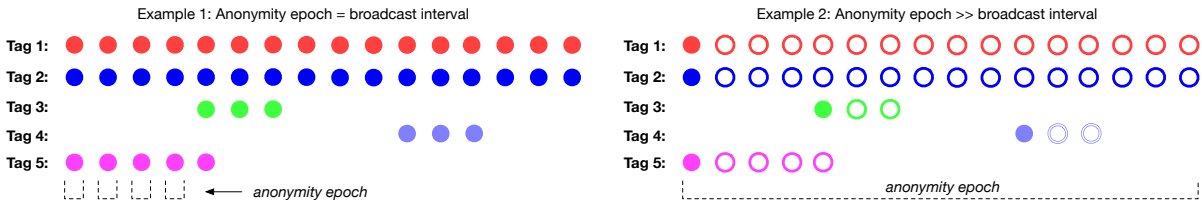


Figure 15: Illustration of the effect of de-duplicating repeated LTA broadcasts: both examples contain the same pattern of transmissions, but use different anonymity epoch durations. Filled circles represent fresh shares, hollow circles represent duplicate shares. **Left:** The anonymity epoch is short: 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.

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.

**Security.** We provide formal theorems and proofs of security for our construction in Appendix E. Briefly, the construction provides strong privacy (in the sense of Definition 2.4) under the condition that a tracking adversary receives fewer than $t_{priv}$ total broadcasts within a given $L$-epoch detection period. This guarantee follows naturally from the unlinkability property of the underlying MDSS. The conditions for detection require that at least $t_{rec}$ shares are received from a victim device, although in practice this threshold can be subject to specific conditions about the noise rate and the number of stalkers that we will discuss in the next section.

## 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 14.

**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. In the best case, this will result in a new pseudonym (via a call to Beacon) with every broadcast that the LTA emits. For systems with a similar broadcast rate to existing LTA deployments, this would imply an anonymity epoch duration of 2-4 seconds,[12] producing between 900 to 1800 hourly unique secret shares from each LTA device. Our experiments in §7 show that this traffic rate can be efficiently decoded by our algorithms.

*Configuration 2: anonymity epoch ≫ broadcast rate.* Current LTA deployments do not change the pseudonym with each broadcast.[13] 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.

[12]This is based on analysis of Apple's FindMy, where LTAs broadcast every 2 seconds [29, 28]. 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 the same 2-second broadcast interval (see §6).

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

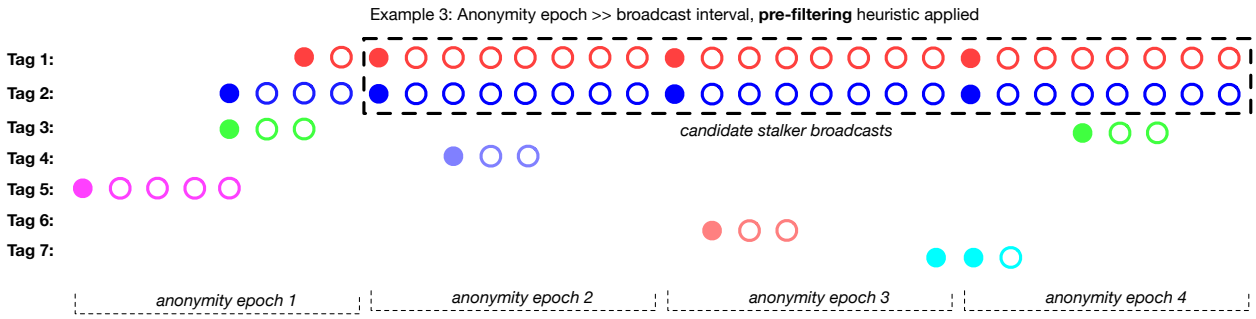


Figure 16: Illustration of a pre-filtering heuristic when the anonymity epoch is longer than the broadcast interval. Hollow circles represent duplicate broadcasts sent by each LTA. In this example, only the broadcasts identified within the dotted rectangle are candidates to be emitted by stalking tags (and hence their broadcasts passed to Detect), since only these LTAs transmit repeated identifiers for a large fraction of each epoch.

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 15.) On the positive side, these repeated broadcasts dramatically reduce the impact of erasures caused by RF-layer issues.

*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 noise 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: during periods where an LTA changes its secret, the victim may not receive a sufficient number of shares to detect a stalker. This problem is similar to the "midnight problem" that affects existing LTA deployments in which identifiers change once every day.[14] To address this, we propose two possible deployment options:

*Use long re-generation periods.* The most straightforward solution is to re-generate secrets relatively rarely. For example, $L$ can be selected so that re-generation occurs once per 24-hour period, given a deployment with a 1-hour detection window. This approach ensures that detection issues will only occur during one hour of each day.

*Use multiple secrets simultaneously.* Each LTA can maintain two tag identifiers (with corresponding secret-sharing polynomials) at all times, emitting one share of each identifier during every broadcast period. To ensure that stalker detection is never interrupted, the LTA can stagger its regeneration period to ensure that stalker detection will always operate for at least one identifier within the given detection window. For example, using a 1-hour detection window each tag could re-generate one of its two secrets every hour.

[14] For example in Apple's FindMy, the LTA pseudonym changes once every 24 hours in separated mode, which means that stalker detection may be more challenging close to this changeover time.

# 5 Selecting MDSS Parameters

Deploying our main construction of §4 requires us to identify reasonable secret sharing parameters. The challenge is to optimize privacy while also enabling *efficient* secret sharing recovery in the face of reasonable noise rates. We first present an overview of our chosen parameters, and then in the following sections we discuss the process of deriving these parameters.

**Overview and recommendations.** We identify four high-level configuration parameters: (1) the (minimum) number of minutes of continuous broadcasting after which a victim device must be able to detect a stalker LTA, (2) the duration of the anonymity epoch (*i.e.,* the time between pseudonym rotation), (3) the frequency of LTA secret/polynomial updates, and (4) the bandwidth available for transmitting each secret share. These parameters, along with assumptions about broadcast rate and worst-case noise rates, will determine the privacy guarantees of the scheme as well as the MDSS parameters.

For the purposes of this analysis we made several exemplary choices for the above parameters, and note that these can be adjusted for specific deployments. Concretely:

1. We configure stalker detection to operate after an LTA has been in proximity to a victim device for 1 hour.
2. We consider three epoch durations inspired by existing LTA networks: 2 seconds, 4 seconds and 1 minute.
3. We determine that LTAs will change their LTA-specific secret (and secret sharing polynomials) every 24-hour period, to ensure that broadcasts cannot be correlated over longer time periods.
4. We consider a worst-case *lost broadcast* rate of up to 5% for transmissions sent by stalking LTAs.[15]
5. We assume that at most three persistent ("stalker") LTAs will be in proximity to the victim at any given time.
6. We derive a bandwidth limit for current-generation LTAs of 248 bits, based on the available payload space in BLE (pre-v5 advertisement messages.) For BLEv5 deployments we assume a higher bandwidth limit of up to 400 bits.

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 (for further discussion on this see Appendix G). 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.
We now give more details on parameter derivation.

**Bandwidth, epoch duration, number of polynomials.** To determine available bandwidth and broadcast rate we examined existing LTA deployments, focusing on Apple's FindMy [28, 29]. 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 [29].[16] This leaves 246 available bits for the protocol to transmit data.[17] 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

[15] In our laboratory experiments we did not observe loss rates this extreme for nearby devices: we chose this pessimistic assumption to anticipate more challenging real-world conditions.
[16] Two bits must remain unused to conform to the BLE standard.
[17] Future LTAs may use BLEv5, which supports up to 255 bytes of advertisement payload.

| 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):* | | | | | | | | | | |
| 2 sec | 60 min | 3 | 43,200 | 22 | 10 | 242 | 40 min | 1652 | 1186 | 6300 |
| 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):* | | | | | | | | | | |
| 2 sec | 60 min | 3 | 43,200 | 22 | 17 | 396 | 46 min | 1652 | 1377 | 6300 |
| 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. We consider three different anonymity epoch durations to support LTAs with different capabilities, and assume "ambient" noise equal to half a stalker's number of broadcasts. Privacy times are rounded to the nearest minute.

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 Section 3.1, 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 $c = 9$.[18]

**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:

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.* [28], and recorded each received broadcast along with the current time and GPS coordinates.

**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 2 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

[18] 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 2: 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.

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.
- 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.

We account for erasures by lowering the number of broadcasts needed to detect an LTA. Based off the epoch duration and detection window, we can calculate the number of unique shares, $T$, which would be broadcast by a persistent LTA if they were present for the whole window. From these parameters and an upper bound on the noise that should be tolerated (from both persistent and ephemeral LTAs) it is possible to derive an upper bound on the number of broadcasts $E$ lost due to collisions and deletions. $t_{rec}$ is then set to $T - E$ to account for these losses.

*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: $t_{rec} \geq \frac{1}{c+1}(c \cdot (t_{priv} + 1) + \mathsf{max})$. 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

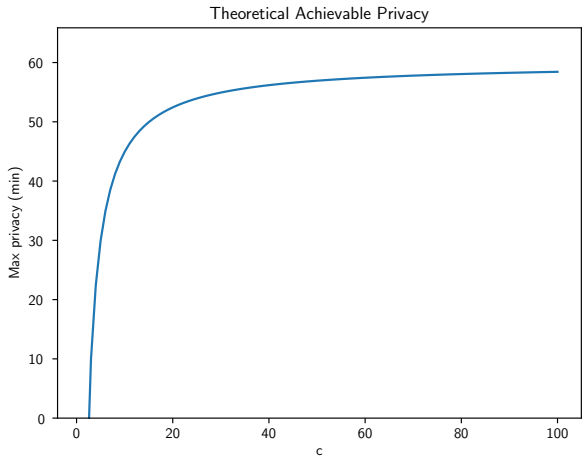


Figure 17: Shows achievable privacy, in terms of minutes, as the number of interleaved polynomials ($c$) increases. Here we assume an epoch duration of 4 seconds, with 3 stalkers and non-stalker broadcasts equal to half those sent by a single stalker. Note that this graph does not consider the impact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

tolerate, the lower the achievable privacy. Deletions can come from two sources: physical layer issues and $x$-coordinate collisions. In practice we assume that physical layer issues are minimal, configuring our schemes to tolerate up to a 5% physical drop rate. More impactful are the $x$-coordinate collisions, which can cause shares to be dropped in two possible ways.

1. As discussed in Section 3.1, 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 current 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.

**Limitations of our overall approach.** First, we note that the detection window provides a strict upper bound on the privacy we can guarantee. Second, our scheme requires us to estimate the maximum noise level that stalking victims will encounter. If actual noise exceeds this estimate, recovery may fail and a stalking tag will not be detected. Such a failure could occur from either a significantly larger than expected number of ephemeral LTAs, or a stalker deliberately using enough LTAs to overwhelm the system. We attempt to mitigate these potential failures by setting parameters so that the former is unlikely to happen in practice and the latter is *economically* unlikely: rather than purchase sufficient LTAs a potential stalker would be more likely to purchase a more sophisticated tracker.

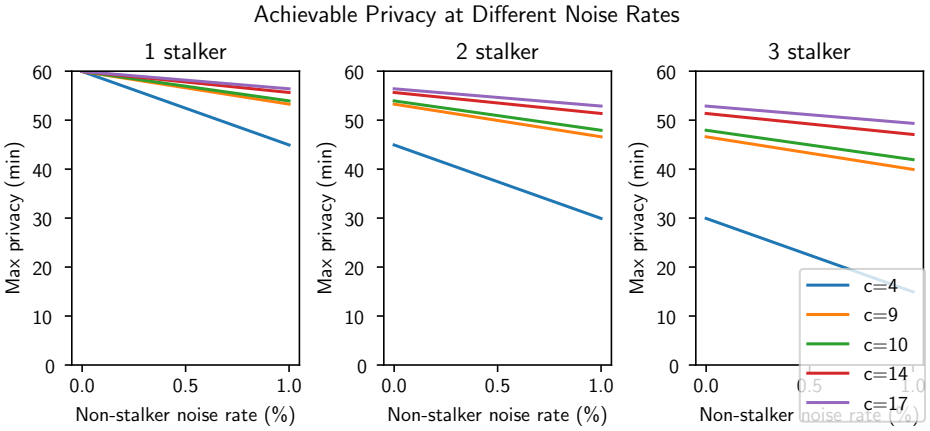


Figure 18: **Impact of interleaving RS polynomials ($c$).** Achievable privacy, in terms of minutes, for our construction (§4) using different values of $c$, assuming an epoch duration of 4 seconds. The horizontal axis shows the ephemeral noise rate measured as a percentage of a single LTA's output.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

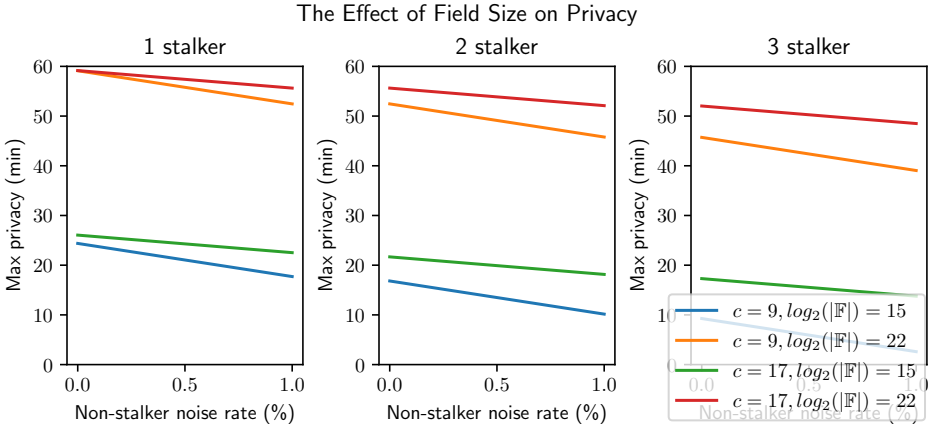


Figure 19: **Impact of collisions in the $x$ coordinate.** Achievable privacy, in terms of minutes, for our construction (§4) for $c \in \{9, 17\}$ and field sizes $\mathbb{F}_p$ of 15 and 22 bits. This calculation assumes that polynomials are re-generated every 24 hours, and the epoch duration is 4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

| Epoch duration | Avg. nearby LTAs | # Unique bxs received | Detection runtime (no stalkers) | Detection runtime (stalkers present) | Polynomial recovery (all stalkers) |
|---|---|---|---|---|---|
| 2 sec | 1 | 2700 | 2.35 | 4.42 sec | 5.82 sec |
| | 2 | 4500 | 5.29 sec | 30.67 sec | 39.45 sec |
| | 3 | 6300 | 9.43 sec | 103.25 sec | 162.62 sec |
| 4 sec | 1 | 1350 | 1.37 sec | 1.06 sec | 2.14 sec |
| | 2 | 2250 | 2.13 sec | 7.28 sec | 10.08 sec |
| | 3 | 3150 | 3.23 sec | 25.89 sec | 39.71 sec |
| 60 sec | 1 | 90 | 0.97 sec | 0.03 sec | 0.98 sec |
| | 2 | 150 | 0.97 sec | 0.08 sec | 1.04 sec |
| | 3 | 210 | 0.98 sec | 0.20 sec | 1.24 sec |

Table 3: Benchmarks for our Detect algorithm (Figure 14 using the CH*-MDSS decoding algorithm with the recommended parameters of Table 1.

# 6 Implementation

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

**Implementing the Detect algorithm.** We implemented our algorithms using SageMath [50], 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.

A production deployment will need to store received broadcasts with the corresponding time and GPS coordinates. These can be used to display candidate stalker tag locations on a map for the user, or used to implement more sophisticated heuristics.

**Implementing the Beacon algorithm.** For ease of comparison, we adopt the algorithmic choices used by 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.

To implement the secret sharing scheme we use the recommended parameters from Table 1, and we further optimize the Beacon algorithm to divide it 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. As described in Section 21 if $r$ has not been previously generated during the detection period, we emit the share $(r, p_1(r), \ldots, p_c(r))$. If $x$ has previously been used, we emit $(r, s_1, \ldots, s_c)$ where $s_1, \ldots, s_c$ are pseudorandom field elements.

*An optimization: using a collision-aware PRF.* A challenge in our implementation is the need for each dealer to detect when $x$ has been repeated. Implemented naively this can require a non-trivial amount of memory. Although we do not implement it for our current encoder, it may be desirable to sample $x$ *pseudorandomly* in a manner that does not require us to keep track of all past $x$ coordinates. To accomplish this, we propose a novel symmetric-key cryptographic primitive that we refer to as a *collision-aware* PRF (CA-PRF). We present the details in Appendix C.

**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. $N$ is chosen to correspond to a *detection window* that ensures that $t_{rec}$ shares will be received from any nearby LTAs, given a $\delta$ broadcast loss rate. The Detect algorithm can be run

every few minutes once the buffer is full. This strategy will provides a relatively timely warning of stalking behavior and can be adjusted to account for computational resources.

Note that the decoding time of CH*-MDSS is variable. Detecting whether *any* sufficient share-set is present requires exactly one lattice reduction. If $d \geq 1$ sufficient share-sets are present, then the lattice reduction will need to be repeated at least $d + 1$ times to recover each polynomial. Once a polynomial has been recovered, future shares from the same LTA can be efficiently detected (and filtered out) prior to running the decoder again. Hence, although polynomial recovery is relatively more expensive than basic detection, this optimization ensures that it will only be used rarely: *i.e.,* when *new* persistent LTAs are present.

# 7 Experiments

In this section we evaluate the empirical runtime of our stalker-detection and encoding algorithms. Our experiments were conducted on a single core of a 2020 MacBook computer with an M1 chip and 16 GB of RAM running MacOS Ventura 13.3.1 (a).

**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 three different anonymity epochs: 2 seconds, 4 seconds, and 60 seconds. More concretely, we performed the following experiment:

1. For each anonymity epoch (2 sec, 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. 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 projected runtime. Our decoder was implemented as described in §6 and configured according to Table 1. Table 3 presents our running times, which for common cases take seconds.

**Encoding.** In our implementation, the time required to generate a secret share is dominated by public-key generation, which takes well under a second. We benchmarked the relative running times of key generation, secret share generation and polynomial rotation and present the results in Table 4.

# 8 Extensions

In this section we briefly discuss some extensions to the constructions in this work.

**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 [28, 29].

| Algorithm | $t_{priv}$ (epoch dur.) | Runtime |
|---|---|---|
| GeneratePublicKey | | 480 $\mu$s |
| GeneratePolynomials | 41 (60 sec) | 350 $\mu s$ |
| GenerateSecretShare | 41 (60 sec) | 10 $\mu$s |
| GeneratePolynomials | 591 (4 sec) | 5070 $\mu$s |
| GenerateSecretShare | 591 (4 sec) | 80 $\mu$s |
| GeneratePolynomials | 1652 (2 sec) | 10,490 $\mu$s |
| GenerateSecretShare | 1652 (2 sec) | 160 $\mu$s |

Table 4: Runtime average over 10,000 iterations for the GeneratePublicKey, GenerateSecretShare and GeneratePolynomials subroutines of the Beacon algorithm (see §6). Secret sharing uses the 22-bit field with $c = 10$ for the 2 second and 4 second epochs, and the 24-bit field with $c = 9$ for the 60 second epoch. For secret sharing (only), the runtime also depends on the polynomial degree $t_{priv}$.

**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.* [34] 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.

# 9 Related Work

*Offline finding.* Several works have considered the privacy and integrity of the offline finding (tracker) ecosystem. Heinrich, Bittner and Hollick [28] evaluated the anti-stalking mechanism use in FindMy. Heinrich, Stute, Kornhuber and Hollick [29] also considered the privacy of Apple's FindMy protocol. Mayberry *et al.* [35] considered ways to bypass tracking alerts in AirTag devices, and in a separate work Mayberry, Blass and Fenske [34] devised protocols to protect against counterfeit tags.

*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 of Bhowmick *et al.* [9] defines the notion of a *detectable hash function* based on interleaved Reed-Solomon codes [12] for secret sharing. Similarly, the STAR protocol of Davidson *et al.* [21] emits shares of (multiple) secrets for private telemetry reporting: unlike our work, that 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 [43, 13, 14, 37, 51, 40, 16].

A key difference between the classical RSS setting and our setting is that in RSS there is exactly one dealer and thus RSS does not incorporate the notion of unlinkability that is required by our protocols.

# 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:

1. **Enhanced security against malicious tags.** While we present one approach to detecting/preventing malicious (counterfeit) tag broadcasts in §8, this solution provides only limited security against a sophisticated attacker. Devising more powerful detection strategies remains a problem for future work.

2. **Realizing improved MDSS schemes.** The MDSS notion we realize in this work likely has many applications in privacy-preserving protocols. Although the instantiations we decribe in this paper are efficient, developing improved constructions remains an open problem. Such schemes may employ different codes, which can benefit from more performant list decoding algorithms.

3. **General access structures.** The constructions in this work consider two different broadcast patterns for enabling stalker detection. Future works may expand this consideration to other patterns, including general access structures.

4. **Alternative applications.** The schemes described in this work may have applications to other areas where privacy-preserving protocols can benefit from selective de-anonymization. These include tools such as bounded group signatures (where overuse of a single signing key would de-anonymize the signer) as well as other applications to private transactions.

# 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] 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.

[14] 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.

[15] 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.

[16] 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.

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

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

[19] 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.

[20] 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.

[21] 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.

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

[23] 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.

[24] H. Givehchian, N. Bhaskar, E. Rodriguez Herrera, H. Lopez Soto, C. Dameff, D. Bharadia, and A. Schulman. Evaluating physical-layer BLE location tracking attacks on mobile devices. In *IEEE Symposium on Security and Privacy (S&P '22)*, pages 507–521, 2022.

[25] 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.

[26] 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.

[27] 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.

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

[29] 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.

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

[31] 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.

[32] 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.

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

[34] 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.

[35] 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.

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

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

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

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

[40] 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.

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

[42] 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.

[43] 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.

[44] 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.

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

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

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

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

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

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

[51] 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.

[52] 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.* [29]. Here we briefly summarize some relevant findings.

**Privacy, device types, separated mode.** As discussed in §2.1, 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 [32] 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 $(sk, k_1, \mathsf{cfg})$

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

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

$k \leftarrow k_1$

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

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

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

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 $k_{\mathsf{tag}}$ as $(sk, k_1)$ and derives key:

  $k \leftarrow k_1$

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

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

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

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 $C$:

  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 20: 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[19]. We provide the full details here for completeness, based on the reverse-engineering work of Heinrich *et al.* [28, 29]. 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 §8.). A non-forward secure variant of the scheme is shown in Figure 20. 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 $i_{\mathsf{epoch}}$ increments before using it in ReRandomize.

[19]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

# B Proofs for MDSS schemes

## B.1 Proof of Claim 3.0.3

For an unlinkability adversary $\mathcal{A}$, let $\mathsf{U}$ be the event that the share $\mathcal{A}$ receives as a response to its "challenge" query in $\mathsf{ExpLink}^b_{\mathcal{A}}(\lambda)$ has a unique $x$ coordinate. We now prove Claim 3.0.3.

**Claim B.0.1.**

$$\Pr[\mathsf{ExpLink}^0_{\mathcal{A}}(\lambda) = 1 | \mathsf{U}] = \Pr[\mathsf{ExpLink}^1_{\mathcal{A}}(\lambda) = 1 | \mathsf{U}]$$

*Proof.* Before the "challenge" stage of the game the distribution that $\mathcal{A}$ sees is identical. We can focus on the response $\mathcal{A}$ sees to the challenge query. When the challenge response $x$ coordinate is unique and $b = 0$, $\mathcal{A}$ sees the $t_{priv}$th point from $c$ degree $t_{priv}$ polynomials, which is equivalent to a uniformly random value of $\mathbb{F}^{c+1}$. When $b = 1$, $\mathcal{A}$ 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.0.2.** $\Pr[!\mathsf{U}] \leq \frac{t_{priv}}{|\mathbb{F}|}$

*Proof.* $\Pr[!\mathsf{U}]$ is equivalent to the probability that the $x$ coordinate $\mathcal{A}$ receives in the challenge phase 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 $\Pr[!\mathsf{U}] = \frac{t_{priv}}{|\mathbb{F}|}$. □

**Claim B.0.3.**

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

*Proof.* Let $\mathsf{ZL}$ be the event that $\mathsf{ExpLink}^0_{\mathcal{A}}(\lambda) = 1$, and $\mathsf{OW}$ be the event that $\mathsf{ExpLink}^1_{\mathcal{A}}(\lambda) = 1$. Then,

$$\begin{aligned}
&|\Pr[\mathsf{ZL}] - \Pr[\mathsf{OW}]| \\
=&|\Pr[\mathsf{ZL}|\mathsf{U}]\Pr[\mathsf{U}] + \Pr[\mathsf{ZL}|!\mathsf{U}]\Pr[!\mathsf{U}] - \Pr[\mathsf{OW}|\mathsf{U}]\Pr[\mathsf{U}] \\
&- \Pr[\mathsf{OW}|!\mathsf{U}]\Pr[!\mathsf{U}]| \\
=&|\Pr[\mathsf{ZL}|!\mathsf{U}]\Pr[!\mathsf{U}] - \Pr[\mathsf{OW}|!\mathsf{U}]\Pr[!\mathsf{U}]| && \text{(Claim B.0.1)} \\
\leq&\Pr[\mathsf{ZL}|!\mathsf{U}]\Pr[!\mathsf{U}] + \Pr[\mathsf{OW}|!\mathsf{U}]\Pr[!\mathsf{U}] \\
\leq&2\Pr[!\mathsf{U}] \\
=&2\frac{t_{priv}}{|\mathbb{F}|} && \text{(Claim B.1)} \\
=&negl(\lambda) && (|\mathbb{F}| \approx 2^\lambda)
\end{aligned}$$

□

## B.2 Proof of Unlinkability for Small Field Construction

*Proof Sketch.* This result extends the proof of unlinkability for large fields. Before the challenge phase, if the challenger responds with shares with colliding $x$-coordinates, $t_{priv}$-unlinkability implies $x$-unlinkability for all $x \leq t_{priv}$. If the x-coordinate of the challenge collides with a previous share, it is equally likely to have come from either secret, since both are sampled from the uniform distribution. The y-coordinates also have the same distribution: if $b = 0$ the value is a $c$-tuple of random field elements, and when $b = 1$ it is the evaluation of $c$ polynomials of degree $t_{priv}$ which is indistinguishable from a random $c$-tuple. The analysis for unique $x$ coordinates is the same as in B.0.1. □

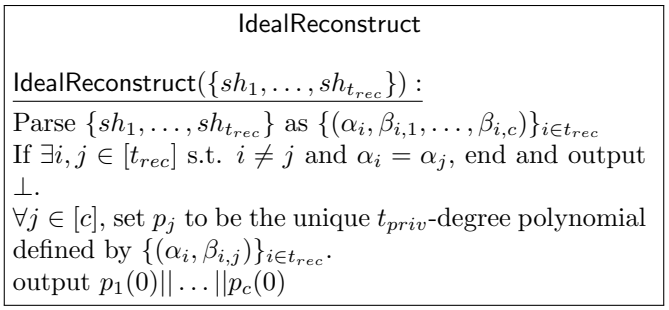


Figure 21: The IdealReconstruct algorithm

### B.3 Ideal Reconstruct

In this section we define an IdealReconstruct algorithm for Construction 13 and bound its probability of failure. Figure 21 illustrates the IdealReconstruct algorithm.

**Claim B.0.4.** *For the* Share *algorithm defined in construction 13, and for all $s \in \mathcal{S}$ and $\mathcal{I}' \subset \mathcal{I}$ where $|\mathcal{I}'| = t_{rec}$,*

$$\Pr[\mathsf{IdealReconstruct}(\mathsf{Share}(s, \mathcal{I}')) \neq s] \leq \frac{t_{rec}(t_{rec} - 1)}{2|\mathbb{F}|}$$

*Proof.* It is clear that correct reconstruction is guaranteed when $\forall i, j \in [t_{rec}]$, $\alpha_i \neq \alpha_j$.

IdealReconstruct thus fails when two shares have the same $x$-coordinate. This probability is given by the birthday bound of $(t_{rec}(t_{rec} - 1))/(2|\mathbb{F}|)$ with $t_{rec}$ balls and $|\mathbb{F}|$ bins. □

We note that for a non-heuristic construction of MDSS, for example using the Guruswami-Sudan [26] list-decoding algorithm for Reed-Solomon codes, claim B.0.4 combined with the algorithm guarantees results in an MDSS construction that can be formally proved to satisfy $\epsilon$-strong-md-correctness. We defer a formal treatment to the full version.

## C Collision-Aware PRFs

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 $\{0, 1, 2, \ldots\}$ 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 C.1** (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 $b$.

## C.1 A construction for a CA-PRF

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. $\mathsf{SampleProb}(k, i, M_i, p) \to (j', b)$. 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 $\ell$ 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

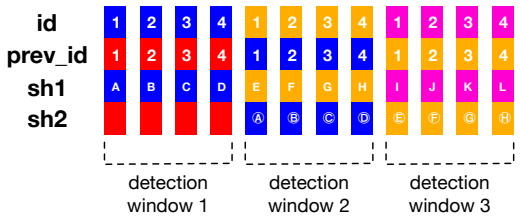


Figure 22: Illustration of broadcast identifiers in the window-based scheme (Appendix §D) for window size $\ell = 4$. id represents the identifier for the current epoch, and prev_id represents the identifier from $\ell$ epochs previous. For id and prev_id each unique color/number combination represent a distinct pseudonymous identifier. Circled/uncircled letters of the same color represent matching secret shares.

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.

# D A Window-based construction

As a separate contribution, we describe a simple window-based 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

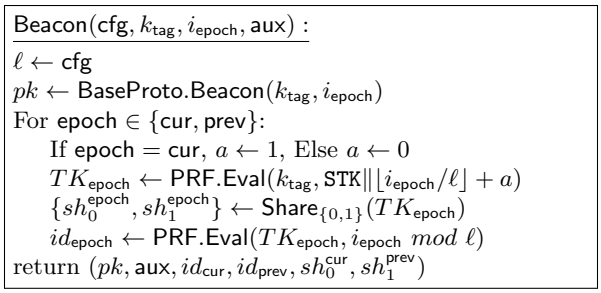


Figure 23: The Beacon and Detect algorithms for our window-based abuse-resistant tracking protocol. STK is a constant. KeyGen and RetrieveReports are identical to those in BaseProto. The GenReport algorithm simply parses $pk$ from each broadcast and otherwise operates as in BaseProto.

by edge devices and are not relayed to the service provider. More critically, these additional identifiers are repeated in the broadcast that occurs exactly $\ell$ epochs later. The value $\ell$ 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, because 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 22.

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.

For space reasons we leave a formal description of the scheme and security proofs to a future full version of this work.

**Privacy limitations of the window-based scheme.** While the scheme above meets reasonable notions of privacy and abuse-detection, it is also quite fragile. The main limitation of this approach is that a tracking adversary may be able to link (de-anonymize) a tag even in situations where the tag *has not been in proximity to the receiver for a significant fraction of the detection window.* For example, a tracking adversary may be able to de-anonymize an LTA that even briefly enters its receive range during two consecutive detection windows. This is very different from our desired functionality, which is that receivers can only de-anonymize LTAs that remain consistently within range during (a large fraction of) the detection window. In the next section we propose a different construction that achieves a more robust functionality.

## D.1 Proofs of Tag Indistinguishability and Detectability

### D.1.1 Tag Indistinguishability

In the 2-linkable scheme we have a variable $\ell$ that specifies the length of a stalker detection window. Broadcasts are structured in such a way that the if a device never sees a broadcast for a pair of epochs $(T, T+\ell)$, for any $T$, then recovery is not possible. We can describe this more formally as a predicate $P$:

$P(\mathsf{cfg}, Q) =$"For $\mathsf{id} \in \{0,1\}$, for all epochs $T$, if $(\mathsf{id}, T) \in Q$, then $(\mathsf{id}, T+\ell) \notin Q$ and $Q$ is not a multi-set"

**Theorem D.1.** *The 2-linkable construction presented in 23 satisfies tag indistinguishability for predicate $P$.*

*Proof Sketch.* For this scheme, we provide a description of the hybrids and the claims we need to make, as the overall proof idea is very straightforward. Because we never need to answer $\mathsf{Beacon}$ queries for a specific identity $\mathsf{id}$ on epoch $T$ and $T+\ell$, the $\mathsf{PRF}$ key $TK_{\mathsf{epoch}}$ is not recoverable for *any* $\mathsf{epoch}$. As $\mathcal{A}$ is a polynomial time adversary, they will make at most a polynomial number of queries $q(\lambda)$ including the challenge query. For each query, we have four hybrids: the first two change the additive shares to be shares of random values (not the $\mathsf{PRF}$ keys used to construct $\mathsf{id}_{\mathsf{epoch}}$, for $\mathsf{epoch} \in (\mathsf{cur}, \mathsf{prev})$). The latter two replace $\mathsf{PRF}$ evaluations with random values. We note that the first two are based off of *information theoretic* assumptions while the latter are computational.

After this is done, it is easy to argue indistinguishability, since all values that are uniformly random and not at all related to the bit $b$ chosen by the challenger. If details are omitted from a hybrid, it is assumed they are the same as the previous hybrid. The notation $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Tag},P,b,\mathcal{H}_i}$ denotes the tag indistinguishability experiment with predicate $P$ and using bit $b$ with the changes included from $\mathcal{H}_i$.

$\mathcal{H}_0 =$ The tag indistinguishability game, with predicate $P$ for the scheme in figure 23 and using bit $b = 0$.

A sequence of hybrids $k \in \{0, \ldots, q'(\lambda)\}$, $j \in \{0, \ldots, 3\}$, $\mathcal{H}_{k,j} =$ All of the next lines apply to query $k$. If $j = 0$, replace $sh_0^{\mathsf{cur}}$ with a random value. If $j = 1$, replace $sh_1^{\mathsf{prev}}$ with a random value. If $j = 2$, replace $\mathsf{id}_{\mathsf{cur}}$ with a random value from the $\mathsf{PRF}$'s co-domain. If $j = 3$, replace $\mathsf{id}_{\mathsf{prev}}$ with a random value from the $\mathsf{PRF}$'s co-domain.

We note that $\mathcal{H}_0 = \mathcal{H}_{0,0} = \ldots = \mathcal{H}_{0,3}$ (we consider all queries to be one indexed, not zero indexed).

Consider the chain of hybrids $\mathcal{H}_{0,0} \ldots \mathcal{H}_{0,3}, \mathcal{H}_{1,0}, \ldots \mathcal{H}_{1,3}, \ldots \mathcal{H}_{q'(\lambda),0}, \ldots \mathcal{H}_{q'(\lambda),3}$, re-indexed by $j$ from 1 to $4(q'(\lambda)+1)$.

**Claim D.1.1.** $\forall j \in \{0, \ldots 4(q'(\lambda)+1)-1\}$, *if the* $\mathsf{PRF}$ *is secure, then there exists a negligible function negl, s.t.* $\Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Tag},P,0,\mathcal{H}_j}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Tag},P,0,\mathcal{H}_{j+1}}(\lambda) = 1\right] \leq negl(\lambda)$

Now that all queries receive random values, not based on the bit $b$, it is easy to switch the bit.

**Claim D.1.2.** $\Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Tag},P,0,\mathcal{H}_{4(q'(\lambda)+1)}}(\lambda) = 1\right] = \Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Tag},P,1,\mathcal{H}_{4(q'(\lambda)+1)}}(\lambda) = 1\right]$

The rest of this argument would be symmetric and so we end the proof sketch here. □

### D.1.2 Detectability

Detectability in the case of our 2-linkable scheme is simply related to seeing at least two $\mathsf{Beacon}$ calls that are precisely $\ell$ epochs apart from the same device. To put it more precisely,

$$P'(\mathsf{cfg}, Q, \mathsf{id}) = \text{“ } \exists \mathsf{id}, T \text{ s.t. } Q \text{ contains } (\mathsf{id}, T) \in Q \text{ and } (\mathsf{id}, T+\ell) \in Q \text{ ”}$$

We note that detectability is easily satisfied for this construction: the detection algorithm simply looks for two broadcasts $B = (pk, id_{\mathsf{cur}}, id_{\mathsf{prev}}, sh_0^{\mathsf{cur}}, sh_1^{\mathsf{prev}})$, $B' = (pk', id'_{\mathsf{cur}}, id'_{\mathsf{prev}}, sh_0^{\mathsf{cur}'}, sh_1^{\mathsf{prev}'})$ where $id_{\mathsf{cur}} = id'_{\mathsf{prev}}$. We recover $TK$ as $\mathsf{Recover}_{\{0,1\}}(sh_1^{\mathsf{prev}'}, sh_0^{\mathsf{cur}})$ and identify all $\mathsf{Beacon}$ values sent by the LTA in the detection window.

# E MDSS-based construction theorems and proofs (§4)

This appendix contains security arguments for the tracking protocol of §4. Throughout this section, we use two variables $i_{\mathsf{anon}}$ 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 $\mathsf{cfg}$ - as $i_{\mathsf{anon}} = i \mod L$ and $e = \lfloor \frac{i}{L} \rfloor$.

## E.1 Tag-Indistinguishability

Recall that in the $k$-linkable scheme an LTA broadcasts shares for $L$ from the same polynomial. If an observer sees $\leq t_{priv}$ unique shares from these epochs they should not be able to identify the LTA. More formally,

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

*Proof.* 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 14, 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.0.1.** $\Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Tag},P,\mathcal{H}_1}(\lambda) = 1\right] = \frac{1}{q(\lambda)^2} \Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Tag},P,\mathcal{H}_0}(\lambda) = 1\right]$

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 $i^{*,b}_{\mathsf{anon}}$ 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 $\mathsf{PRF}$ evaluations with varous keys for random values. This is not complicated mainly because all $\mathsf{PRF}$ keys are independent of one another and no other function of the keys is given out.

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

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

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 $\mathsf{PRF}$ evaluations using $k_2$ with sampling random values for $k^0_{\mathsf{tag}}$

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

Description of $\mathcal{B}$:

- Generate $k^0_{\mathsf{tag}}$ and $k^1_{\mathsf{tag}}$ using $\mathsf{KeyGen}(\mathsf{cfg})$.
- Let $i$ be an epoch associated with $k^0_{\mathsf{tag}}$ 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 $\mathsf{PRF}$ 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 $\mathsf{Exp}_{\mathcal{B}}^{\mathsf{Prf},b}$ be the experiment for the $\mathsf{PRF}$ security game.

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

We now describe a sequence of hybrids that will allow us to use *inconsistent* randomness to share secrets. Doing this is okay, because - up to the $t_{priv}$ threshold - it is not possible to tell if shares are using the same randomness or not. Thus, for this chain of arguments, we will rely on the $t_{priv}$-unlinkability of the $MDSS$ scheme. We now describe the hybrids in general:

$\mathcal{H}_{7,j}^b$ = For query $t_{priv}-j$ to $\mathsf{id}=b$ with $\lfloor\frac{i}{L}\rfloor=e^{*,b}$ from $\mathcal{A}$, change the response to provide a share that uses *new* randomness (i.e if $s_b$ is the secret that is being used to construct $\mathsf{Beacon}$, sample $r\in_R\{0,1\}^\lambda$ , compute $sh\leftarrow\mathsf{Share}(s_b,i_{\mathsf{anon}}^{*,b};r)$ and use $sh$ for responding to $\mathcal{A}$)

$\mathcal{H}_{7,0}^b$ corresponds to changing the *challenge* query for $\mathsf{id}=b$.

**Claim E.0.3.** *If the $MDSS$ scheme is $t_{priv}$-unlinkable, then there exists a negligible function, negl s.t.* $\forall b\in\{0,1\},\forall j\in[t_{priv}-1]$, $|\Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Tag},P,\mathcal{H}_{7,j+1}^b}(\lambda)=1\right]-\Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Tag},P,\mathcal{H}_{7,j}^b}(\lambda)=1\right]|\leq negl(\lambda)$

We consider the following adversary $\mathcal{B}$ against $\mathsf{ExpLink}$.
Description of $\mathcal{B}$ :

- Generate $k_{\mathsf{tag}}^b\leftarrow\mathsf{KeyGen}(\mathsf{cfg})$, $s_b\xleftarrow{\$}\mathcal{S}$ for $e^{*,b}$, $b\in\{0,1\}$
- 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, let $m$ be the number of queries made to $e^{*,b}$ already. If $m+1=(t_{priv}-j)$ (and $j\neq 0$), query $\mathcal{B}$'s challenger $i$ times to get to the challenge query. For the challenge query send $(s_b,i_{\mathsf{anon}},i_{\mathsf{anon}})$. Receive $sh^*$ and use $sh^*$ to construct the query response to $\mathcal{A}$. If $m<(t_{priv}-j)$, then $\mathcal{B}$ queries the challenger on $i_{\mathsf{anon}}$, receives $sh$ and uses this to construct the response. If $m+1>(t_{priv}-j)$, choose $r\in_R\{0,1\}^\lambda$, calculate $sh\leftarrow\mathsf{Share}(s_b,i_{\mathsf{anon}};r)$ and use this to structure responses.
- For the challenge query, if $j=0,\mathsf{id}=b=0$, and the number of queries is less than $t_{priv}-1$, query up to $t_{priv}-1$ random values. Then query on $(s_0,i_{\mathsf{anon}},i_{\mathsf{anon}})$ where $i_{\mathsf{anon}}=i\pmod L$ where $i$ is the epoch that was received. Use this to construct a response. Otherwise, calculate a random share of $s_b$ and use this instead.
- Receive $\hat{b}$ from $\mathcal{A}$ and output $\hat{b}$

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

$$
\begin{aligned}
&|\Pr\left[\mathsf{ExpLink}_{\mathcal{B}}^1(\lambda)=1\right]-\Pr\left[\mathsf{ExpLink}_{\mathcal{B}}^0(\lambda)=1\right]|\\
&=|\Pr\left[\mathcal{A}\text{ outputs 1 in }\mathcal{H}_{7,j+1}^b\right]-\Pr\left[\mathcal{A}\text{ outputs 1 in }\mathcal{H}_{7,j}^b\right]\\
&=\Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Tag},P,\mathcal{H}_{7,j+1}^b}(\lambda)=1\right]-\Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Tag},P,\mathcal{H}_{7,j}^b}(\lambda)=1\right]\\
&\leq negl(\lambda)
\end{aligned}
$$

While these hybrid chains (for $b \in \{0,1\}$) have been described in parallel they can be chained sequentially. We also would like to comment that this same argument holds for showing that the experiments defined by $\mathcal{H}_6$ and $\mathcal{H}^0_{7,0}$ are indistinguishable. We consider the last in this sequence to be $\mathcal{H}_7$ i.e. $\mathcal{H}_7 = \mathcal{H}^1_{7,t_{priv}-1}$. We now may finally switch the challenger's bit $b$ in $\mathcal{H}_7$ by relying on the the fact that unlinkability does not allows us to distinguish shares made from different secrets.

$\mathcal{H}_8$ = Switch the challenge bit $b$ from 0 to 1.

**Claim E.0.4.** *If the MDSS scheme is $t_{priv}$-unlinkable, then there exists a negligible function, negl s.t.* $|\Pr\left[\mathsf{Exp}^{\mathsf{Tag},P,\mathcal{H}_8}_{\mathcal{A}}(\lambda)=1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{Tag},P,\mathcal{H}_7}_{\mathcal{A}}(\lambda)=1\right]| \leq negl(\lambda)$

Description of $\mathcal{B}$

- Generate $k^0_{\mathsf{tag}}$ and $k^1_{\mathsf{tag}}$ from $\mathsf{KeyGen}$. Choose $s_0, s_1 \leftarrow \mathcal{S}$ for the challenge epochs $e^{*,0}, e^{*,1}$. Initialize $\mathcal{B}$'s challenger with $s_0$.
- For each non-challenge query, $i, \mathsf{id}, \mathsf{aux}$, calculate $i_{\mathsf{anon}} = i \pmod L, e = \lfloor \frac{i}{L} \rfloor$. If $e \neq e^{*,b}$ for either $b = 0$ or $b = 1$, proceed as normal. Otherwise, calculate $r \in_R \{0,1\}^\lambda$ and compute $sh \leftarrow \mathsf{Share}(s_b, i_{\mathsf{anon}}; r)$ and use $sh$.
- For the challenge query, calculate $i^{*,0}_{\mathsf{anon}}$ and $i^{*,1}_{\mathsf{anon}}$ from the input. Send $t_{priv} - 1$ random queries to the challenger and then send the challenge query $(s_1, i^{*,0}_{\mathsf{anon}}, i^{*,1}_{\mathsf{anon}})$. Receive $sh$ and use this to structure the reply to $\mathcal{A}$.
- If $\mathcal{A}$ outputs $\hat{b}$, output $\hat{b}$

Analysis:

$$
\begin{aligned}
&|\Pr\left[\mathsf{ExpLink}^1_{\mathcal{B}}(\lambda)=1\right] - \Pr\left[\mathsf{ExpLink}^0_{\mathcal{B}}(\lambda)=1\right]| \\
&= |\Pr\left[\mathcal{A} \text{ outputs 1 in } \mathcal{H}_8\right] - \\
&\Pr\left[\mathcal{A} \text{ outputs 1 in } \mathcal{H}_7\right]| \\
&= |\Pr\left[\mathsf{Exp}^{\mathsf{Tag},P,\mathcal{H}_8}_{\mathcal{A}}(\lambda)=1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{Tag},P,\mathcal{H}_7}_{\mathcal{A}}(\lambda)=1\right]| \\
&\leq negl(\lambda)
\end{aligned}
$$

We end here with our analysis, as the rest is simply symmetric backwards steps to the original experiment with the bit $b = 1$. This concludes our proof. □

## E.2 Detectability

In our $k$-linkable solution, recovery is possible only when we see at least $t_{rec}$ beacons from a key associated with an $\mathsf{id}$ in a single "abuse epoch" $e$. We need to meet the additional restriction that we cannot receive too many values in an abuse epoch. We give a precise formulation:

$$P'(\mathsf{cfg}, Q, \mathsf{id}) = \text{“ } \exists\ e \text{ such that } |\{(\mathsf{id}, i) \in Q \mid e = \lfloor \tfrac{i}{L} \rfloor\}| \geq t_{rec} \text{ and } |\{(*, i) \in Q \mid e = \lfloor \tfrac{i}{L} \rfloor\}| \leq \mathsf{max}\text{”}$$

Detectability holds via the properties of the list decoding algorithms. When we use heuristic algorithms, detectability correctness is only heuristic.

# F An Alternative CH*-MDSS

In Section 3.1, we noted that the lattice of Figure 11 appears to contain a wealth of information about the structure of existing solutions, that we could potentially use to come up with a more efficient list-decoding algorithm. Here we explore such a possibility. 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 $X - 1$) .

1. When there is one solution set that has more agreeing points than any other set, *adding* the shortest vectors in $M$ produces the vector associated with this solution set.

2. When multiple solutions have the maximum number of agreeing points, $v_0$ contains factors $(z - \alpha_i)$ $\forall i \in [n]$ that do *not* agree with those solution sets.

We observed that using 1 (adding the shortest vectors instead of just assuming the shortest vectors will directly be solutions) allows us to even change the threshold for the *single* valid solution case: $t \geq \frac{1}{c+1}(c \cdot k + n)$ suffices. More importantly, it reduces the class of hard instances: we now only need to worry when there are inputs that have the same amount of agreeing points. With 2, we have a way to identify all points that correspond to the multiple solution sets. If we can identify the points associated with a single solution from this set, then we could possibly develop an iterative procedure which produces all solutions. The main insight we make is that we know the form of our solutions vectors: they must look like $E(z), E(z)p_1(z), \ldots, E(z)p_c(z)$ for some $E, p_1 \ldots, p_c$ of appropriate degree where $E$ is an error locator polynomial. We choose a point $\alpha$ that corresponds to one of the multiple solutions and construct a lattice problem to try and find those 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 sum the shortest vectors in $M$ that come from this lattice problem, we obtain a vector $(r_0 \ldots r_c)$ and associated rational functions $R = (r_1/r_0, \ldots r_c/r_0)$ which agree with all points associated with $x - 1$ solution sets *excluding* the points from the solution set that agrees with $\alpha$. Doing lagrangian interpolation on the excluded points produces the polynomial set $(p_1 \ldots p_c)$ agreeing with $\alpha$. 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. The whole algorithm is depicted in Figure 24.

# G Calibrating Correctness for Parameter Selection

First, we determine the probability that a share broadcast from a stalking LTA may be dropped due to a collision in the $x$-coordinate. This can happen for two reasons: either when a separate LTA broadcasts a share with the same $x$-coordinate, or when the LTA itself had previously sampled the $x$-coordinate within the detection period and therefore must output a noise share. Therefore, for a point to survive, all other shares broadcast within the detection window must have distinct $x$-coordinates. Additionally, it must be the case that at no point previously in the detection period did the LTA sample the $x$-coordinate, and therefore be forced to output a noise share. In the worst case there have been $L-1$ previous detection windows. Thus, the probability $p$ that a share is dropped is:

$$p \leq 1 - \left(\frac{|\mathbb{F}| - 1}{|\mathbb{F}|}\right)^{(L-1+\mathsf{max})*n-1}$$

Given $p$, let $P_i$ be the probability that exactly $i$ shares from a single stalking LTA are dropped due to collisions. To achieve the target success rate, we find the smallest $z$ such that:

$$\sum_{i=0}^{z} P_i > 0.995$$

We then repeat the process for channel deletions. Let $b$ be the number of times each share is broadcast. Then, assuming a channel deletion rate of 0.05, the probability $p'$ that a share is never received by a stalking victim is $p' = (0.05)^b$. Let $P'_i$ be the the probability that exactly $i$ shares from a single stalking LTA are dropped due to channel issues. We then find the smallest $z'$ such that $\sum_{i=0}^{z'} P'_i > 0.995$. 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.

For a stalking LTA let $\mathsf{chd}$ and $\mathsf{cld}$ be random variables representing the number of shares that are dropped by channel issues and collisions respectively. Following this, we have:

$$\begin{aligned}
\Pr[\text{decoding fails}] &= \Pr[n - \mathsf{chd} - \mathsf{cld} < t_{rec}] \\
&= \Pr[\mathsf{chd} + \mathsf{cld} > z' + z] \\
&\leq \Pr[\mathsf{chd} > z' \vee \mathsf{cld} > z] \\
&\leq \Pr[\mathsf{chd} > z'] + \Pr[\mathsf{cld} > z] \\
&\leq 0.01
\end{aligned}$$

as desired.

**Algorithm 3:** An Alternative 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* $\mathsf{fail}$ **do**
  $\forall j \in [c], f_j(z) = \mathsf{LagrInterpol}(\{(\alpha_i, \beta_{i,j})\}_{i \in \mathsf{ws}})$, $N(z) = \prod_{i \in \mathsf{ws}}(z - \alpha_i)$
  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** *the shortest vector in* $M_{\mathsf{red}}$ *is larger than* $k + (|\mathsf{ws}| - t)$ **then**
    set $\mathsf{fail}$ to $True$
  **else**
    find all vectors in $M_{\mathsf{red}}$ with the shortest length, denoted as $\vec{v_1}, \ldots \vec{v_h}$
    $\vec{v} \leftarrow \vec{v_1} + \ldots + \vec{v_h}$, $(v_0, \ldots, v_c) \leftarrow \vec{v}$
    **if** $\forall i \in [c]$, $v_0 | (v_i \cdot z^k)$ **then**
      add $(v_1 \cdot z^k / v_0, \ldots, v_c \cdot z^k / v_0)$ to $\mathsf{solns}$, remove agreeing points in $\mathsf{ws}$
    **else**
      **if** *the shortest vector in* $M_{\mathsf{red}}$ *has length exactly equal to* $k + (|\mathsf{ws}| - t)$ **then**
        set $\mathsf{fail}$ to $True$
      **else**
        choose $i \in \mathsf{ws}$, s.t. $(z - \alpha_i) \nmid v_0$
        let $\vec{b_1}, \ldots, \vec{b_m}$ be row vectors from $M_{\mathsf{red}}$ where $\forall i, \mathsf{Norm}(\vec{b_i}) \leq k + (|\mathsf{ws}| - t)$ and $\vec{b_i} = (b_0^i, \ldots b_c^i)$, construct the matrix $S \in \mathbb{F}_q[z]^{(m+1)\times(c+1+m)}$:

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

        $S_{\mathsf{red}} \leftarrow \mathsf{LatticeReduce}(S)$
        find all vectors in $S_{\mathsf{red}}$ with the shortest length, denoted as $\vec{s_1}, \ldots \vec{s_h}$, construct vectors $\vec{r_1}, \ldots \vec{r_h}$ as $r_i = \sum_{j=0}^{m-1} s_{c+1+j}^i \vec{b_j}$
        $\vec{r} \leftarrow \vec{r_1} + \ldots + \vec{r_h}$, $(r_0, \ldots, r_c) \leftarrow \vec{r}$, $R \leftarrow (r_1 \cdot z^{|\mathsf{ws}|}/r_0, \ldots, r_c \cdot z^{|\mathsf{ws}|}/r_0)$
        **if** *all polynomials in* $R$ *have degree* $\leq k$ *and* $\forall i \in [c]$, $r_0 \mid z^{|\mathsf{ws}|} r_i$ **then**
          add $R$ to $\mathsf{solns}$, remove agreeing points in $\mathsf{ws}$
        $(q_1, \ldots, q_c) \leftarrow R$, $M \leftarrow \{i \mid \forall j \in [c], q_j(\alpha_i) = \beta_{i,j}\}$, $I \leftarrow \{i \in \mathsf{ws} \mid (z - \alpha_i) \nmid v_0\}$, $L \leftarrow I \setminus M$
        $\forall j \in [c], p_j \leftarrow \mathsf{LagrInterpol}(\{(\alpha_i, \beta_{i,j})\}_{i \in L})$
        **if** $(p_1 \ldots p_c)$ *agree with at least* $t$ *points and each* $p_j$ *has degree* $k$ **then**
          add $(p_1 \ldots p_c)$ to $\mathsf{solns}$, remove agreeing points from $\mathsf{ws}$
**endw**
**return** $\mathsf{solns}$

Figure 24: An alternative construction of CH*-MDSS. LatticeReduce is an algorithm for lattice reductions on polynomial lattices.