Birds of a Feather Flock Together: How Set Bias Helps to Deanonymize You via Revealed Intersection Sizes

Xiaojie Guo[1], Ye Han[1], Zheli Liu[1,*], Ding Wang[1], Yan Jia[1,*], Jin Li[2]

[1]*College of Cyber Science, Nankai University,*
{xiaojie.guo, hanye}@mail.nankai.edu.cn, {liuzheli, wangding, jiay}@nankai.edu.cn
[2]*Institute of AI and Blockchain, Guangzhou University, jinli71@gmail.com*

Abstract

Secure two-party protocols that compute intersection-related statistics have attracted much attention from the industry. These protocols enable two organizations to jointly compute a function (e.g., count and sum) over the intersection of their sets without explicitly revealing this intersection. However, most of such protocols will reveal the intersection size of the two sets in the end. In this work, we are interested in how well an attacker can leverage the revealed intersection sizes to infer some elements' membership of one organization's set. Even disclosing an element's membership of one organization's set to the other organization may violate privacy regulations (e.g., GDPR) since such an element is usually used to identify a person between two organizations. We are the first to study this *set membership leakage* in intersection-size-revealing protocols. We propose two attacks, namely, baseline attack and feature-aware attack, to evaluate this leakage in realistic scenarios. In particular, our feature-aware attack exploits the realistic set bias that elements with specific features are more likely to be the members of one organization's set. The results show that our two attacks can infer $2.0 \sim 72.7$ set members on average in three realistic scenarios. If the set bias is not weak, the feature-aware attack will outperform the baseline one. For example, in COVID-19 contact tracing, the feature-aware attack can find 25.9 tokens of infected patients in 135 protocol invocations, $1.5\times$ more than the baseline attack. We discuss how such results may cause negative real-world impacts and propose possible defenses against our attacks.

1 Introduction

Secure two-party computation related to the intersection of two organizations' sets has attracted much attention from the industry. A well-known protocol is Private Set Intersection Cardinality (PSI-CA) [25, 31, 43, 44, 45, 53], which returns the intersection size to one organization without disclosing other information to either organization. This protocol is an essential building block of many applications, such as contact tracing [36, 66, 70], social network [56, 63, 65], and association rule mining [72]. Another prevalent protocol is Private Intersection-Sum with Cardinality (PSI-SUM) [52, 61], which aggregates the values held by one organization and associated with the indices in the intersection. PSI-SUM was first proposed in Google's work [52] to measure the revenue owing to ad conversion. Recently, Facebook proposed Private-ID [28], a protocol for its privacy-preserving measurement of conversion lift [13] in online advertising.

Although these secure computation protocols will never reveal the intersection of two organizations' sets to either organization, they reveal the intersection size of the two sets. As long as such an *intersection-size-revealing protocol* is repeatedly invoked between the two organizations, a **toy attack** can use the revealed intersection sizes to test whether some elements are in the intersection. In this attack, one organization (acting as the *attacker*) can input a singleton set in each protocol invocation with the other organization (acting as the *victim*) and observe the resulting intersection size. If this size equals one, then the involved element is in the intersection; otherwise, it is not. We stress that the attacker can use its target elements in protocol invocations, and *these elements' membership of the intersection essentially reveals their membership of the victim's set*. In other words, the toy attack allows the attacker to infer the target elements' membership of the victim's set. If the victim's set is static, N protocol invocations can leak N elements' set membership.

Our work. We move beyond this toy attack and initiate the first study on the **set membership leakage** resulting from revealed intersection sizes. This leakage is a general issue for all protocols that compute intersection-related statistics *without disclosing any information about the intersection except its size*. Many real-world systems use such protocols to hide as much information about the victim's set as possible from the attacker. In these systems, disclosing any element in the victim's set is not allowed. However, this work shows that it can be risky for the systems with such a security requirement to use intersection-size-revealing protocols.

This work begins with designing two set membership inference attacks to test whether target elements are in the victim's set. First, we show that, even if the attacker learns nothing but intersection sizes, there is a **baseline attack** that leads to a more severe set membership leakage than the toy attack. Second, we consider a realistic situation where the victim's

*Zheli Liu and Yan Jia are corresponding authors.

set is biased concerning specific features. Roughly speaking, this **set bias** means that *the elements with the features are more likely to be included in the victim's set*. An example set bias appears in the health authority's set of COVID-19 patients. This set is supposed to be biased concerning the `fever` feature of persons since fever is a common symptom of COVID-19. We propose a **feature-aware attack** against such a biased set. The intuition is that if the attacker knows the bias-correlated features of target elements, it can use them to improve its inference efficiency.

This work subsequently evaluates the set membership leakage caused by our two attacks and the toy attack in three scenarios: (i) COVID-19 contact tracing from PSI-CA, (ii) the measurement of ad conversion revenue from PSI-SUM, and (iii) the measurement of ad conversion lift from Private-ID. Our measurement shows that the set bias in scenario (i) is non-negligible while those in the other two scenarios are weak. Each scenario allows many invocations of its intersection-size-revealing protocol, where the victim's set is *dynamic*.

In scenario (i), our baseline (resp., feature-aware) attack can identify $3.7 \sim 7.2\times$ (resp., $5.2 \sim 9.0\times$) more set members than the toy attack. In particular, our attacks require far fewer protocol invocations to find as many set members as those found in the toy attack. On average, our baseline (resp., feature-aware) attack can find $12.0/18.1/17.6$ (resp., $13.4/22.4/25.9$) set members in roughly $70/135/135$ protocol invocations when there are initially $14/28/56$ set members in $512/1024/2048$ target elements. In scenarios (ii) and (iii), the toy attack may fail to find any set member, but our attacks still work. For example, in scenario (ii), our baseline (resp., feature-aware) attack can find $2.0/4.0/7.5$ (resp., $2.0/4.2/7.5$) set members in roughly $15/33/60$ protocol invocations when there are initially $2/4/7$ set members in $512/1024/2048$ target elements. Since the victim's set changes with time, our attacks can find some set members different from the initial ones in this scenario. The above results show that our attacks can help the attacker guess the elements in the victim's set in the three scenarios.

This work finally discusses the real-world implications of our results and some possible defenses against our attacks. Notably, the attacker can link its chosen elements to real-world persons in the considered scenarios. Therefore, the set membership leakage of these elements reveals whether the linked persons have interacted with the victim, thereby their associated elements being recorded in the victim's set. Such personal information should be only known to the organization in the interaction, and it violates privacy regulations (e.g., GDPR and HIPPA) that this information is leaked to another unauthorized organization. We also discuss the negative consequences resulting from the leakage of this interaction in the three scenarios. For example, the results indicate that it is possible to combine our attacks with the linkage attack [14] to infer the COVID-19 patients recorded by the health authority. This inference may affect the daily life of these patients.

2 Background

We describe the functionalities of prevalent intersection-size-revealing protocols and the set membership inference problem. Then, we present the threat model of these protocols.

2.1 Intersection-size-revealing Protocols

There are many secure two-party protocols that compute functions of the intersection of two parties' sets without revealing any information about the intersection except its size.

PSI-CA [25, 31, 43, 44, 45, 53]**.** This protocol is a black box that receives a set X from one party and a set Y from the other party. It internally computes the intersection size $|X \cap Y|$ and sends this value to one of the two parties.

PSI-SUM [52, 61]**.** This protocol is a black box that receives a set X from one party and a table $\{(y_i, v_i)\}_{y_i \in Y}$ of index-value pairs from the other party. It internally aggregates the values associated with the indices in the intersection $X \cap Y$ and gets the sum $\sum_{y_i \in X \cap Y} v_i$. Then, the protocol sends this sum and the intersection size $|X \cap Y|$ to the party that inputs the table.

Private-ID [28]**.** This protocol is a black box that receives a set X from one party and a set Y from the other party. It internally assigns random tags to all elements in the union $X \cup Y$ so that both parties will get the same tag for each element in the intersection $X \cap Y$. Then, the protocol sends these tags and the intersection size $|X \cap Y|$ to both parties. The tags can be used in the subsequent secure computation of many statistics of the intersection $X \cap Y$.

Circuit-based PSI [55]**.** This protocol is a black box that receives a table $\{(x_i, u_i)\}_{x_i \in X}$ from one party and a table $\{(y_i, v_i)\}_{y_i \in Y}$ from the other party. Circuit-based PSI can compute *any* function of the two parties' values indexed by the elements in the intersection $X \cap Y$. One can think that circuit-based PSI is the ultimate solution for the computation of intersection-related statistics. [55] implements a protocol of circuit-based PSI and reveals the intersection size $|X \cap Y|$ to both parties to improve protocol efficiency.

Set membership inference problem. For the party that receives intersection size from intersection-size-revealing protocols, **this size measures the similarity between its and the other party's sets**. Since the party can choose its set sent to the protocols, it can measure the other party's set in the way it wants. An interesting problem is whether the party, with many target elements, can determine these elements' membership of the other party's set by leveraging this measurement ability.

2.2 Threat Model

Attacker's abilities. The attacker has the following abilities.

- The attacker is a party of intersection-size-revealing protocols. In each protocol invocation, it is allowed to choose

its input and learn the intersection size resulting from this input and the other party's set. Here, the other party is the victim, and its set can be dynamic.

- The attacker is allowed to invoke the protocols with the same victim repeatedly.

Attacker's background knowledge. We consider the following attackers with two kinds of background knowledge.

- *Baseline attacker*. This attacker has no background knowledge about the victim's set. We use the performance of this attacker to measure the *least* set membership leakage resulting from intersection-size-revealing protocols.
- *Feature-aware attacker*. This attacker knows that some features are correlated with the bias in the victim's set. In other words, the attacker knows that the victim's set tends to include elements with specific features. This is the background knowledge about the set bias in the victim's set.

Attacker's goal. The attacker has many target elements and is interested in these elements' membership of the victim's set. If this set changes with time, the attacker will consider the target elements' **historical set membership**, which indicates whether the target elements appeared in any snapshot of the victim's set from the beginning to the end of the attack.

3 Set Membership Inference Attacks

Here, we introduce two set membership inference attacks performed by our two attackers, respectively. Then, we discuss the case where the victim's set is dynamic.

3.1 Baseline Attack

The baseline attacker adopts a binary-search-like strategy. It first constructs such a binary tree that (i) each node stores a non-empty set, (ii) the root stores the set of target elements, and (iii) the set stored in a non-leaf node will be partitioned into two non-empty and disjoint subsets, which will be respectively stored in the two child nodes of the non-leaf node.

For node v of the binary tree T, let $T_v \subseteq X$ denote the set stored in node v and $\mathsf{leftChild}(v)$ (resp., $\mathsf{rightChild}(v)$) denote the left (resp., right) child node of v. Our baseline attack follows the blueprint in Algorithm 1 and uses Algorithm 2 as a subroutine to build a binary tree.

To infer the target elements' set membership, the attacker runs depth-first search (DFS) for the *queued* subtrees (the queue initially contains the constructed binary tree). In each DFS, the attacker (i) traverses a subtree popped from the queue, (ii) invokes the protocol with the victim to receive the intersection size regarding the set stored in each visited node, and (iii) terminates at a node where the size of the stored set equals the received intersection size. The unvisited subtrees during this search are pushed into the queue for future DFS

Algorithm 1: Blueprint of set membership inference

Input: A set X of target elements, background knowledge $\mathtt{BK}$, an algorithm buildTree.
Output: A set membership predicate σ such that, for each $x_i \in X$, $\sigma(x_i) = 1$ if x_i is in the victim's set; otherwise $\sigma(x_i) = 0$.

1 Initialize $Z \leftarrow \varnothing$.
2 Input X to the protocol and get an intersection size m.
3 $T \leftarrow \mathsf{buildTree}(X, \mathtt{BK})$. // Build a binary tree based on the attacker's background knowledge
4 forest $\leftarrow \{(m/|X|, (m, T.\mathsf{root}))\}$. // A priority queue of (priority, subtree-info) tuples
5 **while** forest is not empty **do**
6 $(m_{\mathsf{node}}, \mathsf{node}) \leftarrow \mathsf{forest.pop}()$.
7 **while** $0 < m_{\mathsf{node}} < |T_{\mathsf{node}}|$ **do**
8 $\mathsf{L} \leftarrow \mathsf{leftChild}(\mathsf{node})$, $\mathsf{R} \leftarrow \mathsf{rightChild}(\mathsf{node})$.
9 **if** T_{R} contains more elements than T_{L} **then**
10 Input T_{L} to the protocol and get an intersection size m_{L}.
11 $m_{\mathsf{R}} \leftarrow m_{\mathsf{node}} - m_{\mathsf{L}}$.
12 **else**
13 Input T_{R} to the protocol and get an intersection size m_{R}.
14 $m_{\mathsf{L}} \leftarrow m_{\mathsf{node}} - m_{\mathsf{R}}$.
15 **if** $m_{\mathsf{R}}/|T_{\mathsf{R}}| > m_{\mathsf{L}}/|T_{\mathsf{L}}|$ **then**
16 $\mathsf{forest.push}((m_{\mathsf{L}}/|T_{\mathsf{L}}|, (m_{\mathsf{L}}, \mathsf{L})))$.
17 $\mathsf{node} \leftarrow \mathsf{R}$, $m_{\mathsf{node}} \leftarrow m_{\mathsf{R}}$. // Goto right child
18 **else**
19 $\mathsf{forest.push}((m_{\mathsf{R}}/|T_{\mathsf{R}}|, (m_{\mathsf{R}}, \mathsf{R})))$.
20 $\mathsf{node} \leftarrow \mathsf{L}$, $m_{\mathsf{node}} \leftarrow m_{\mathsf{L}}$. // Goto left child
21 **if** $m_{\mathsf{node}} > 0$ **then** $Z \leftarrow Z \cup T_{\mathsf{node}}$.
22 Let σ be such a predicate that $\sigma(x_i) = 1$ iff $x_i \in X \cap Z$.
23 **return** σ.

Algorithm 2: buildTree for the baseline attacker

Input: A set X. Ignore the parameter $\mathtt{BK}$.
Output: A binary tree T.

1 Initialize T to contain a single node root that stores X.
2 Build a *balanced binary tree* T by recursively dividing a stored set into two non-empty and disjoint subsets.
3 **return** T.

(lines 16 and 19). In the end, the attacker empties the queue and finds all subsets containing the target elements that the victim also holds. Meanwhile, the attacker can see that the other target elements are non-members.

The remaining question is how the baseline attacker partitions the set stored in a non-leaf node. Recall that, for the baseline attacker, each target element has the same probability of being a set member. It would be better for this attacker to randomly and evenly partition the set and build a *balanced* binary tree in the end. This practice ensures the minimum expected length of a DFS path, or rather, the minimum expected

number of protocol invocations.

Efficiency analysis. Our baseline attack is more efficient than the toy attack due to the binary-search-like strategy. Suppose that the victim's set Y is static and the attacker has constructed a binary tree T. Our first observation is that the following equality holds for any non-leaf node v of the tree T:

$$|T_v \cap Y| = |T_{\mathsf{leftChild}(v)} \cap Y| + |T_{\mathsf{rightChild}(v)} \cap Y|. \quad (1)$$

With this equality, the attacker can locally determine the intersection size of one child node (e.g., $|T_{\mathsf{rightChild}(v)} \cap Y|$) if it has learned that of the parent node (e.g., $|T_v \cap Y|$) and that of the other child node (e.g., $|T_{\mathsf{leftChild}(v)} \cap Y|$). Thus, the attacker can save the vain invocations for the intersection sizes that can be determined locally. In our implementation, the attacker uses this property to maintain the queue of subtrees where, except for the initial case, the intersection size of each pushed subtree's root has been locally determined (lines 16 and 19). Thus, the attacker can save *at least* half of the protocol invocations for every non-root layer of the binary tree.

Recall that in a balanced binary tree with $N := |X|$ leaf nodes, there are $2N - 2^{\lceil \log_2 N \rceil}$ leaf nodes with depth $\lceil \log_2 N \rceil$. There is an *upper bound* of the *total* number of invocations to determine all intersection sizes on the binary tree:

$$1 + \frac{1}{2} \cdot \sum_{d=1}^{\lceil \log_2 N \rceil - 1} 2^d + \frac{1}{2} \left(2N - 2^{\lceil \log_2 N \rceil} \right) = N, \quad (2)$$

which equals the total number of invocations required by the toy attack. This bound guarantees that the efficiency of our baseline attack is no worse than that of the toy attack.

Our second observation is that this upper bound is loose, given that each DFS may early terminate at a non-leaf node with a depth less than $\lceil \log_2 N \rceil$. This early termination comes from the fact that some singleton sets each of a target element are *merged* into a larger subset of the victim's set. When a DFS encounters this subset, the attacker will learn the set membership of a batch of target elements and terminate this DFS. In this case, the number of invocations is reduced due to the shortened length of the DFS path. Moreover, the number of the remaining target elements is also reduced and fewer DFS passes are required for further inference. Thus, our baseline attack should be more efficient than the toy attack.

Efficiency optimization: greedy DFS. Since the merge of set members improves attack efficiency, the attacker should first search the subtree with the greatest merge probability. This probability is positively related to the ratio of the intersection size of the subtree's root to the number of target elements in this subtree. This ratio is known to the attacker, and the attacker can use this ratio as the *priority score* to sort the subtrees pushed into the (priority) queue. In this way, the attacker will first run DFS on the popped subtree with the greatest merge probability (line 6). In each DFS, the attacker recursively visits the child node whose subtree has the greatest priority score (line 15). An example is presented in Figure 1.

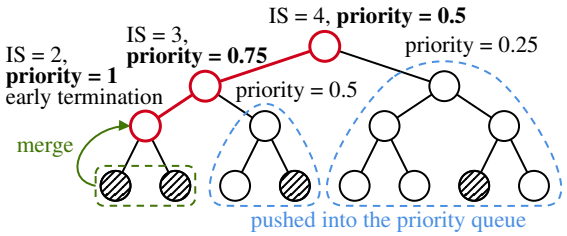


Figure 1: An example of early termination. The DFS path with intersection size (IS) and priority is highlighted in red, and each shadowed leaf node corresponds to a set member.

Algorithm 3: buildTree for the feature-aware attacker

Input: A set X, background knowledge BK.
Output: A binary tree T.

1 Parse BK as a table D_X such that $D_X[x_i]$ denotes the features associated with a target element $x_i \in X$.
2 Initialize T to contain a single node root.
3 $T_{\mathsf{root}} \leftarrow X$, queue $\leftarrow \{\mathsf{root}\}$.
4 **while** queue is not empty **do**
5 node $\leftarrow$ queue.pop().
6 Use features $\{x_i \in T_{\mathsf{node}} : D_X[x_i]\}$ to run k-means to divide T_{node} into two subsets S_{L} and S_{R}.
7 Append two child nodes L and R to node.
8 $T_{\mathsf{L}} \leftarrow S_{\mathsf{L}}$, $T_{\mathsf{R}} \leftarrow S_{\mathsf{R}}$.
9 **if** $|S_{\mathsf{L}}| > 1$ **then** queue.push(L).
10 **if** $|S_{\mathsf{R}}| > 1$ **then** queue.push(R).
11 **return** T.

3.2 Feature-aware Attack

Our feature-aware attack also follows the blueprint in Algorithm 1 but turns to Algorithm 3 to build a different binary tree. The only difference between the feature-aware attack and the baseline attack is how the subsets of target elements are distributed over the constructed binary tree.

Recall that the ideal event in a set membership inference attack is that the attacker inputs a set of target elements to an intersection-size-revealing protocol and finds that they all are members of the victim's set. This event significantly improves the attack efficiency in that (i) the length of the DFS path is reduced, and (ii) the number of the remaining target elements whose set membership is to be inferred is also reduced.

The intuition behind our feature-aware attack is that the attacker can trigger this ideal event *more often than the baseline attacker*. Recall that the set bias inclines the victim's set to include the elements associated with specific features. The feature-aware attacker can use its knowledge about this set bias and the related features to adjust the subset distribution over the constructed tree. More concretely, the attacker uses the known features to partition a set of target elements into two disjoint subsets such that (i) the elements in the same subset have similar features, and (ii) any two elements belonging to two different subsets are different in the features. Using

this partitioning strategy, the attacker is more likely to find set members in batches.

In our implementation, we simply use `k-means` [57, 58] to hierarchically partition a set stored in a parent node into two subsets respectively stored in the two child nodes. The hierarchical invocations of `k-means` naturally assign the elements with different features to the leaf nodes far away on the binary tree. Meanwhile, the target elements with similar features are clustered in the leaf nodes close to each other. The resulting binary tree is expected to enable more early terminations than its counterpart in the baseline attack.

There is, however, a limitation of our feature-aware attack: the constructed binary tree may not be balanced. This limitation is because `k-means` does not necessarily partition a set into two subsets of the same size.

3.3 Applying the Attacks to A Dynamic Set

Following the blueprint (Algorithm 1), our two attacks require several protocol invocations to infer the target elements' set membership. Due to Equation 1, the inference result must be correct if the victim's set is static in these invocations. However, this set can be dynamic in realistic scenarios. We need to consider how such a change will affect the correctness of our attacks.

An important observation is that this effect occurs only if the changed elements in the victim's set happen to be those targeted by the attacker. Otherwise, the set change does not affect Equation 1, and the two attacks' correctness still holds. Hence, we consider two primary cases regarding the *changing* intersection of the two parties' sets:

- **Increasing intersection.** The victim's set grows during the attack, *and* the added elements have been targeted.
- **Decreasing intersection.** The victim's set shrinks during the attack, *and* the removed elements have been targeted.

Let $t(v) \geq 0$ denote the number of invocations before the attacker determines the intersection size of the set stored in the node v and Y^t denote the snapshot of the victim's set Y after $t \geq 0$ invocations. For the increasing case and any non-leaf node v with two child nodes L and R, we can see that

$$\begin{aligned} &|T_{\mathsf{L}} \cap Y^{t(v)}| \leq |T_{\mathsf{L}} \cap Y^{t(\mathsf{L})}|, \quad |T_{\mathsf{R}} \cap Y^{t(v)}| \leq |T_{\mathsf{R}} \cap Y^{t(\mathsf{R})}|, \\ &|T_v \cap Y^{t(v)}| = |T_{\mathsf{L}} \cap Y^{t(v)}| + |T_{\mathsf{R}} \cap Y^{t(v)}| \\ &\qquad\qquad \leq |T_{\mathsf{L}} \cap Y^{t(\mathsf{L})}| + |T_{\mathsf{R}} \cap Y^{t(\mathsf{R})}|, \end{aligned} \tag{3}$$

where the equality holds if $Y^{t(v)} = Y^{t(\mathsf{L})} = Y^{t(\mathsf{R})}$.

Inequality 3 shows that, for an increasing intersection, Algorithm 1 may *underestimate* one child node's *current* intersection size due to the simple subtraction in lines 11 and 14. Given that the attacker's goal is historical set membership (Section 2.2), this underestimation will lead to false negatives and no false positive. False negatives come from the fact that if Algorithm 1 visits the subtree containing an element newly added to the victim's set, another set member in the other subtree will be *falsely* recognized as a non-member. However, since all intersection sizes are underestimated, there is no capacity for a node whose intersection size meets the termination condition (i.e., $m_{\mathsf{node}} = |T_{\mathsf{node}}|$) to contain a non-member element. Thus, there is no false positive in this case.

For a decreasing intersection, we can follow a similar analysis to the previous case and find that

$$|T_v \cap Y^{t(v)}| \geq |T_{\mathsf{L}} \cap Y^{t(\mathsf{L})}| + |T_{\mathsf{R}} \cap Y^{t(\mathsf{R})}|, \tag{4}$$

where the equality holds if $Y^{t(v)} = Y^{t(\mathsf{L})} = Y^{t(\mathsf{R})}$. Inequality 4 shows that the attacker may *overestimate* one child node's *current* intersection size. The overestimation here will cause false positives and no false negative. False positives are due to the *false* early termination on a node whose intersection size is overestimated and does not actually equal its capacity. The overestimation also promises that no set member will be missed, and thus there is no false negative.

A mixed case. In this case, the victim's set may change in the way that, after each invocation, some target elements are added to the set while some others already in the set are removed. This mixed case can be regarded as a composition of the two primary cases. It is expected that the attacker will obtain both false positives and false negatives in this case.

4 Evaluation Setup

Here, we aim to evaluate the realistic set membership leakage caused by our attacks. Section 4.1 describes three realistic scenarios that claim to use intersection-size-revealing protocols. Section 4.2 presents the sources of the real-world datasets related to these scenarios. Section 4.3 shows the characteristics of the victim's *dynamic* sets in these datasets. Section 4.4 describes the attacker's inputs to be used to evaluate our two attacks. Section 4.5 introduces our evaluation metrics.

4.1 Realistic Scenarios

Table 1 presents some intersection-size-revealing protocols, followed by their application scenarios and high-level systems. Due to the application of such protocols, these real systems allow the revelation of intersection size. How the protocols are used in the scenarios is recalled below.

COVID-19 contact tracing from PSI-CA. How PSI-CA is used in COVID-19 contact tracing varies with system design. *Token-based* contact tracing systems (e.g., Epione [69, 70] and Catalic [35, 36]) allow two persons to automatically exchange via Bluetooth their real-time tokens when they are in close proximity. Each person maintains a set of tokens received from others. To identify any exposure to COVID-19 but preserve the privacy of infected patients, a person can

Table 1: Intersection-size-revealing protocols and their applications to realistic scenarios.

Protocol	Scenario	High-level System	Developer
PSI-CA and its variant	COVID-19 contact tracing	Epione [69, 70]	Researchers
		Catalic [35, 36]	Researchers
		COVID-19 Alert [66, 67]	Openmined
PSI-SUM	Measurement of ad conversion revenue	Private Join and Compute [46, 47, 52]	Google
Private-ID	Measurement of ad conversion lift	Private Lift [40, 41, 42, 62]	Facebook

use the token set to invoke a PSI-CA with the health authority, who has a set of infected patients' tokens. The person receives the intersection size of the two sets and learns that it has contacted so many infected patients.

In *location-based* contact tracing systems (e.g., COVID-Alert [66, 67]), the health authority maintains a set of the locations that COVID-19 patients have visited. A person can test its exposure by using the set of its visited locations to run PSI-CA[1] with the health authority. The intersection size given to the person shows the number of locations where it may have contacted any infected patients.

Measurement of ad conversion revenue from PSI-SUM. In this scenario, an advertiser (e.g., a retailer) wants to calculate the revenue contributed by the persons who have clicked its ad displayed by a publisher. Google's Private Join and Compute [46, 47, 52] facilitates this calculation by using PSI-SUM, where the advertiser inputs a table of (`personal_id`, `spending`) pairs and the publisher inputs a set of ad clickers' `personal_id`s. The advertiser will receive the aggregate conversion revenue from PSI-SUM and the number of `personal_id`s shared by the two parties.

Here, `personal_id` serves as the index to identify the persons who have clicked the ad and purchased the product. A `personal_id` in advertising campaigns can be a hashed email address, a hashed phone number [29, 50], or an International Mobile Equipment Identity (IMEI) number [26]. It is usually linked to a user profile and can identify a real-world person.

Measurement of ad conversion lift from Private-ID. Facebook's Private Lift [40, 41, 62] realizes A/B testing [1] to tell an advertiser how many *incremental* conversions were generated because persons have seen its ad displayed by a publisher. Here, the publisher selected some persons who were interested in the ad. During the A/B testing, the publisher randomly assigned each person to either a treatment group or a control group. The persons in the treatment group would see the ad, while those in the control group were qualified to see the ad but would not see it from the publisher [13, 23].

The first step to measure the conversion lift concerning the two groups is that the advertiser and the publisher invoke Private-ID [28] to align their sets of `personal_id`s. Specifically, the advertiser inputs a set of customers' `personal_id`s, and the publisher inputs a set of the `personal_id`s of all persons in the two groups. Then, Private-ID assigns a fresh unique tag to each `personal_id` in the *union* of the two parties' sets. At the end of this step, the two parties will receive from Private-ID the tags of all `personal_id`s in the union and the number of the `personal_id`s shared by the two parties. Facebook's Private Lift further uses these tags in the secure downstream computation of conversion lift.

A note on the attacker's abilities in these scenarios. We note that the attacker's abilities assumed in Section 2.2 are satisfied in the above scenarios. First, the attacker can be (i) the person who tests its exposure in COVID-19 contact tracing, (ii) the advertiser in the measurement of ad conversion revenue, or (iii) the advertiser in the measurement of ad conversion lift. Thus, the attacker in each scenario is supposed to receive intersection size from the underlying protocol. Second, these scenarios do need to invoke the intersection-size-revealing protocols regularly. In COVID-19 contact tracing, the person should periodically test its exposure to COVID-19. In the two advertising scenarios, the advertiser is allowed to request the statistics on an hourly or daily basis[2] (e.g., [24, 51, 71]) to make informed marketing decisions.

A note on the attacker's target elements and background knowledge in these scenarios. We note that (i) the attacker can collect valid elements used in the scenarios so that its target elements chosen from them are also valid, and (ii) the attacker can learn some features of the target elements.

In COVID-19 contact tracing, we focus on the two token-based systems. Such systems are based on the automatic exchange of tokens and can be used with other COVID-19 control measures (e.g., temperature screening [17, 18, 19, 20] and symptom screening [21, 22]). The attacker can set up its mobile phones in specific places to collect personal tokens and record the person(s) showing up at the moment when each token was received. This technique is known in the linkage attack [14] against the tokens of COVID-19 patients. Given this token-person record, if the attacker also has access to the data collected in those control measures, it can learn some symptom features (e.g., fever and cough) of each recorded person and associate these features with the personal token.

In the two advertising scenarios, the advertiser can obtain

[1] According to the technical explanation [67], COVID-Alert uses Homomorphic Encryption to compute the intersection size of two sets of GPS coordinates. Thus, this system is based on PSI-CA.

[2] For the readers familiar with secure computation, there is a reusability issue of the tags generated by Facebook's Private-ID protocol. We discuss this issue in Appendix A.

personal_ids from (i) its first-party data of customers and (ii) third-party marketplaces [10, 11]. These personal_ids are immediately linked to real-world persons and are associated with many personal features (e.g., demographics).

4.2 Data Sources

To quantify the leakage threat faced by the victim in the three scenarios, we use the following three public datasets, each of which is related to one scenario.

- **COVID-19 dataset of tested individuals in Israel[3].** This dataset was collected by Israel Ministry of Health [7] and the authors [73] translated it into English. After data cleaning, there are 255,668 distinct individuals who were tested for COVID-19 from March 22, 2020, to April 30, 2020. Each individual record has a test date and is associated with eight features. This dataset is used for the leakage quantification in COVID-19 contact tracing.
- **Taobao's dataset of ad display/click records[4].** This dataset was collected on Taobao, a Chinese online shopping platform owned by Alibaba Group. This platform allows small businesses and individual entrepreneurs to open online retail stores and sell their products. After data cleaning, there are 25,029,435 ad display/click records concerning 827,009 ads and 1,061,768 individuals. The records were collected from May 6, 2017, 00:00:00 AM to May 14, 2017, 00:00:00 AM. Each individual is associated with eight features. This dataset is used for the leakage quantification in the measurement of ad conversion revenue.
- **Tencent's dataset of ad display records[5].** This dataset was collected by Tencent for the 2019 Tencent Advertising Algorithm Competition. It contains 102,386,695 ad display records concerning 509,280 ads and 1,341,958 individuals. The records were collected from February 17, 2019, to March 20, 2019. Each individual is associated with ten features. This dataset is used for the leakage quantification in the measurement of ad conversion lift.

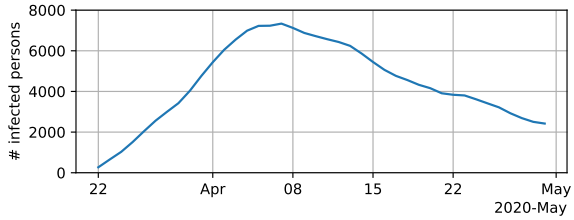
A note on the two ad datasets. Due to a large number of ads, we only present the evaluation results of the ad with the most records. The ad_ids are 710164 and 320379 for the two datasets, respectively.

Ethics. Datasets used in this work are publicly available after the proper data anonymization that their owners performed. Thus, we cannot deduce any Personal Identifiable Information (PII) of real-world individuals from these datasets. In our evaluation, we only exploited the time-dependent distribution of the population of each dataset and measured the statistical leakage concerning artificial IDs.

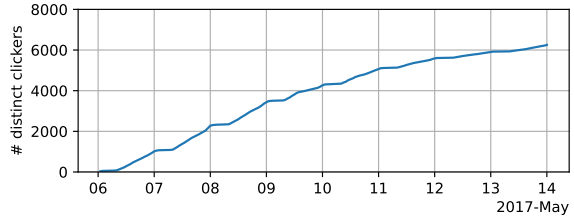
[3] https://github.com/nshomron/covidpred

[4] https://tianchi.aliyun.com/dataset/dataDetail?dataId=56&lang=en-us

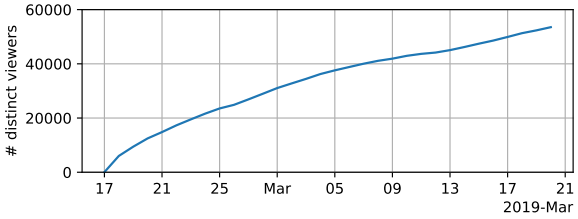
[5] https://algo.qq.com/archive.html?&lang=en



(a) COVID-19 contact tracing.



(b) Measurement of ad conversion revenue.



(c) Measurement of ad conversion lift.

Figure 2: The size change of the victim's dynamic set.

Table 2: Parameters in the simulation of the attacker's input.

Scenario	**Attack Time** (β_0)	$\lvert X\rvert$	β
COVID-19 contact tracing	April 4, 2020 (0.0273)	512	$\{\beta_0, 5\beta_0, 10\beta_0\}$
		1024	β_0
		2048	β_0
Measurement of ad conversion revenue	May 9, 2017, 00:00:00 AM (0.0032)	512	$\{\beta_0, 5\beta_0, 10\beta_0\}$
		1024	β_0
		2048	β_0
Measurement of ad conversion lift	February 19, 2019 (0.0070)	512	$\{\beta_0, 5\beta_0, 10\beta_0\}$
		1024	β_0
		2048	β_0

4.3 Characteristics of the Victim's Set

We can extract the victim's realistic sets in the scenarios from the given datasets. A set of the victim is supposed to include (i) the tokens of infected patients in COVID-19 contact tracing, (ii) the personal_ids of the persons who have clicked the ad in the measurement of ad conversion revenue, or (iii) the personal_ids of the persons who were qualified to see the ad (i.e., to participate the A/B testing) in the measurement of ad conversion lift. We artificially assign unique IDs to the persons in the datasets. These IDs are regarded as the tokens or the personal_ids in the scenarios.

We represent the victim's dynamic set as a time series of *snapshot sets*. The victim uses each snapshot set in a time-

Table 3: Set bias with respect to each feature, measured in mutual information.

Scenario	`feature name` (MI value)
COVID-19 contact tracing	`fever` (0.0168), `cough` (0.0099), `gender` (0.0004)
Measurement of ad conversion revenue	`age` (0.0010), `gender` (0.0007), `shopping_level` (0.0002), `work` (0.0002), `consumption_ability` (0.0001), `city_level` (0.0001)
Measurement of ad conversion lift	`marriage_status` (0.0013), `education` (0.0012), `consumption_ability` (0.0012), `age` (0.0009), `work` (0.0005), `gender` (0.0001)

specific protocol invocation, and a snapshot set contains the elements that the victim held in a past *time window*. That is, outdated elements will be excluded from the current snapshot set. Note that this time window is introduced by the concerned scenarios. In COVID-19 contact tracing, the health authority is suggested to keep the tokens of the infected patients in the last 14 days [70], which equals the maximum incubation period of COVID-19. In the two online advertising scenarios, such a time window is known as the "conversion window" or "attribution window", capturing the number of days between when a person clicked or viewed an ad and subsequently took action. Its size is chosen by the advertiser (i.e., the attacker) and can range from 1 to 90 days [49]. The publisher is asked to keep the `personal_id`s of the persons who clicked or viewed the ad during this period. Since the time span of each advertising dataset is less than 90 days, we simply set the window size of the dataset to be its time span.

We extract a time series of snapshot sets for each dataset by sliding a time window over the time span of the dataset. A snapshot set is formed from all elements in its time window. In Figure 2, we show the size change of the victim's set.

4.4 Simulation of the Attacker's Input

To evaluate the set membership leakage caused by our attacks, we need to simulate the attacker's input (i.e., a set of target elements, and the features used by the feature-aware attacker) to be forwarded into Algorithm 1.

The simulation of the set X of target elements works as follows. Note that the attacker's choice of target elements eventually affects the proportion $\beta \in [0,1]$, defined as the number of the target elements in the snapshot set *at the beginning of the attack* divided by the total number of the target elements. Since we cannot exhaust the attacker's choices of target elements, we turn to simulate its sets with different set sizes and values of β. Given the time to launch the attack and the snapshot set at this moment, we simulate a set X for the set size $|X|$ and the proportion β as follows.

- According to the attack time and the snapshot set, divide the population U of the dataset into two disjoint sets. The *positive set* A is just the snapshot set itself, while the *negative set* $B := U \backslash A$ is the complement of A.
- Uniformly sample $\lceil \beta \cdot |X| \rceil$ persons from the positive set and $|X| - \lceil \beta \cdot |X| \rceil$ persons from the negative set. The set X is formed from the artificial IDs of the sampled persons.

We define $\beta_0 := |A|/|U|$ and summarize in Table 2 the parameters used in the simulation of the set X. We explain our choices of β in Appendix B.1.

We proceed to consider the features that the feature-aware attacker can use. Since an artificial ID is assigned to each real-world person, we can associate the corresponding personal features with the ID. Our evaluation only uses the features that the feature-aware attacker can easily identify in real life. In the COVID-19 dataset, the attacker can learn (`fever`, `cough`, `gender`). In the two advertising scenarios, the attacker only learns some demographic features, i.e., (`age`, `gender`, `consumption_ability`, `shopping_level`, `work`, `city_level`) in Taobao's dataset and (`age`, `gender`, `education`, `consumption_ability`, `marriage_status`, `work`) in Tencent's dataset. The preprocessing of the features is given in Appendix B.2.

A note on the number of protocol invocations. This number equals the number of invocations per time unit multiplied by the number of the time units after the beginning of the attack. In COVID-19 contact tracing, the time unit is a day, and we set the number of invocations per day to five, which is practical for the attacker. In the measurement of ad conversion revenue, the time unit is an hour, and the number of invocations per hour is one due to hourly reporting [51]. In the measurement of ad conversion lift, the time unit is a day, and the number of invocations per day is one due to daily reporting [24, 71].

4.5 Metrics

We repeat the evaluation in each parameter setting 20 times and use the toy attack (in our Introduction) for comparison. We consider the following metrics for each attack.

- # total: the total number of the target elements whose set membership is determined.
- # members: the number of the target elements diagnosed as the members of the victim's set.
- # TP, # FP, # TN, and # FN: the number of (i) true positives, (ii) false positives, (iii) true negatives, and (iv) false negatives. These metrics are measured according to Section 3.3 where a positive denotes a set member and a negative

Table 4: # total breakdown at the end of the attack in COVID-19 contact tracing. T: toy attack, B: baseline attack, K: feature-aware attack instantiated with `k-means`.

	# TP	# FP	# TN	# FN	% initial TP
	$\lvert X\rvert = 512, \beta = \beta_0 = 0.0273, \lceil \beta \cdot \lvert X\rvert \rceil = 14$				
T	2.6 (1.1)	0.0 (0.0)	128.4 (2.4)	4.0 (1.8)	n/a
B	12.0 (2.1)	2.6 (1.6)	486.2 (2.8)	11.1 (3.0)	0.63 (0.16)
K	13.4 (1.3)	1.7 (1.0)	488.1 (2.7)	8.8 (2.3)	0.69 (0.11)
	$\lvert X\rvert = 1024, \beta = \beta_0 = 0.0273, \lceil \beta \cdot \lvert X\rvert \rceil = 28$				
T	2.5 (1.3)	0.0 (0.0)	128.6 (2.2)	3.9 (1.7)	0.52 (0.41)
B	18.1 (2.9)	9.1 (2.8)	923.1 (58.8)	32.5 (3.6)	0.43 (0.08)
K	22.4 (2.3)	6.7 (2.2)	964.4 (5.5)	29.9 (4.6)	0.57 (0.07)
	$\lvert X\rvert = 2048, \beta = \beta_0 = 0.0273, \lceil \beta \cdot \lvert X\rvert \rceil = 56$				
T	3.0 (1.4)	0.0 (0.0)	127.5 (2.4)	4.5 (2.0)	n/a
B	17.6 (2.2)	17.8 (4.8)	769.9 (116.3)	33.6 (8.8)	0.43 (0.08)
K	25.9 (3.8)	15.3 (4.2)	791.8 (266.9)	32.7 (10.1)	0.67 (0.07)
	$\lvert X\rvert = 512, \beta = 5\beta_0 = 0.1367, \lceil \beta \cdot \lvert X\rvert \rceil = 70$				
T	6.7 (2.1)	0.0 (0.0)	113.7 (3.8)	14.7 (3.4)	0.77 (0.13)
B	24.9 (3.3)	34.6 (3.9)	396.0 (5.1)	56.0 (3.6)	0.75 (0.09)
K	40.2 (4.1)	24.3 (3.4)	402.9 (17.7)	40.5 (5.0)	0.84 (0.06)
	$\lvert X\rvert = 512, \beta = 10\beta_0 = 0.2734, \lceil \beta \cdot \lvert X\rvert \rceil = 140$				
T	12.9 (2.7)	0.0 (0.0)	95.0 (3.9)	27.1 (4.1)	0.88 (0.10)
B	49.0 (4.4)	64.5 (4.8)	295.9 (7.6)	101.0 (5.7)	0.87 (0.05)
K	72.7 (4.9)	53.4 (9.5)	306.5 (12.5)	77.0 (5.3)	0.93 (0.03)

denotes a non-member. Note that # total = # TP + # FP + # TN + # FN and # members = # TP + # FP.

- % initial TP: the number of the true positives, which come from the victim's snapshot set at the beginning of the attack, divided by the total number of true positives.

The first two metrics show the speed of set membership leakage, and we plot these metrics as functions of the number of protocol invocations. The other metrics can only be measured at the end of the attack due to the victim's dynamic set and are written in "`mean` (`standard deviation`)".

5 Evaluation Results

5.1 Set Bias Measurement

We measure the set bias in each scenario using Mutual Information (MI) [16] between the used features and whether an element appeared in the victim's snapshot set at the beginning of the attack. There is an MI value between each feature and the set membership. The higher the MI value, the stronger the set bias concerning this feature. In Table 3, we sort the MI values in the three scenarios in descending order. These values are measured at the beginning of the attack.

5.2 Set Membership Leakage in the Scenarios

5.2.1 Scenario 1: COVID-19 contact tracing

The characteristics of this scenario are that (i) the attacker invokes PSI-CA many times (i.e., 5 per day $\times$ 27 days), and (ii) the features used by the attacker are correlated with the set bias. We expect that facing these characteristics, our feature-aware attack can cause the most severe set membership leakage, followed by our baseline attack and finally the toy attack.

Figure 3 visualizes that our two attacks can determine the set membership of more target elements than the toy attack. In particular, they can find more historical set members in the victim's dynamic set as expected. We also present in Table 4 the accuracy metrics of the three attacks. As discussed in Section 3.3, there are false positives and false negatives in our two attacks due to the set change in Figure 2a. Although the set members inferred from our attacks are not perfectly correct like the toy attack[6], our attacks outperform the toy one in terms of the number of true positives. This number determines whether the attacker can identify many true set members in the limited number of PSI-CA invocations. Considering this metric and the leakage speed in Figure 3, we conclude that our two attacks lead to a more severe set membership leakage than the toy attack.

Table 4 shows that our feature-aware attack has fewer false positives and false negatives than our baseline attack. Recall that the two attacks follow the same blueprint and only differ in the usage of features. Thus, the difference in attack performance must owe to the bias-correlated features. In fact, we can see from Figure 3 that these features help to increase leakage speed. This result coincides with our expectation that the attacker with bias-correlated features can infer set membership in batches and reduce the number of PSI-CA invocations. This reduction prevents a significant change in the victim's set, and the set change is the cause of false positives and false negatives. Thus, the feature-aware attack is more accurate than the baseline attack. The insignificant set change also ensures that the feature-aware attack can find more set members in the victim's initial snapshot set, as indicated by % initial TP. To sum up, the feature-aware attacker does cause a more severe leakage than the baseline one.

However, the feature-aware attacker's advantage in inferring set members comes at the cost of reducing the number of non-members obtained in the early stage of the attack. This phenomenon is apparent when $\lvert X\rvert \in \{512, 1024, 2048\}$ and $\beta = 0.0273$. The reason is that the binary tree is unbalanced in our feature-aware attack and the DFS passes early in the attack fill the priority queue with more subtrees of small size. These subtrees may contain fewer non-members but increase the number of PSI-CA invocations to empty the queue.

[6] The toy attack has no false positive since, in this attack, the set membership of one element does not affect that of another one.

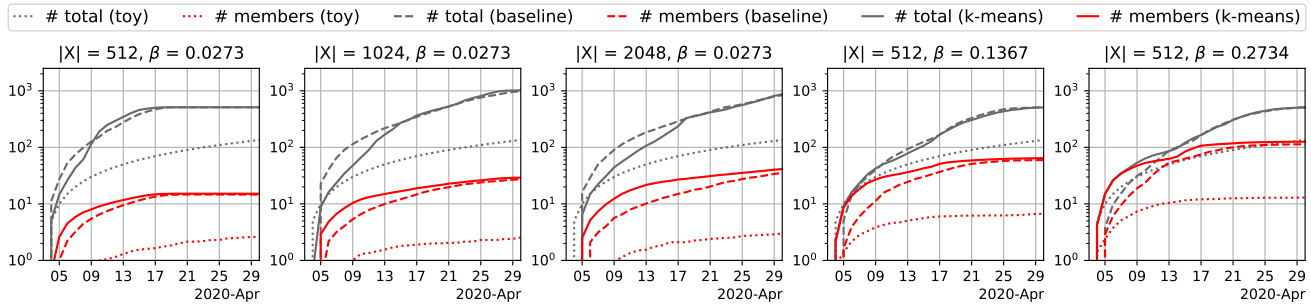


Figure 3: Set membership leakage in COVID-19 contact tracing, measured in # total and # members. We plot the two metrics after the last (i.e., 5-th) PSI-CA invocation per day.

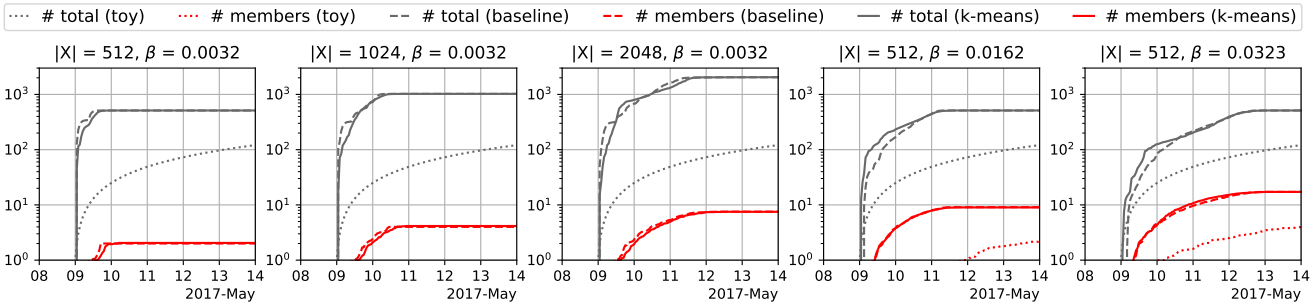


Figure 4: Set membership leakage in the measurement of ad conversion revenue, measured in # total and # members.

Table 5: # total breakdown at the end of the attack in the measurement of ad conversion revenue. T: toy attack, B: baseline attack, K: feature-aware attack instantiated with `k-means`.

	# TP	# FP	# TN	# FN	% initial TP
	$\lvert X\rvert = 512, \beta = \beta_0 = 0.0032, \lceil\beta \cdot \lvert X\rvert\rceil = 2$				
T	0.3 (0.5)	0.0 (0.0)	120.6 (0.5)	0.1 (0.2)	n/a
B	2.0 (0.0)	0.0 (0.0)	509.6 (0.6)	0.3 (0.6)	1.00 (0.00)
K	2.0 (0.2)	0.0 (0.0)	509.6 (0.6)	0.3 (0.6)	0.88 (0.20)
	$\lvert X\rvert = 1024, \beta = \beta_0 = 0.0032, \lceil\beta \cdot \lvert X\rvert\rceil = 4$				
T	0.8 (0.7)	0.0 (0.0)	120.0 (0.7)	0.2 (0.4)	n/a
B	4.0 (0.0)	0.0 (0.0)	1019.5 (0.7)	0.5 (0.7)	0.97 (0.07)
K	4.2 (0.4)	0.0 (0.0)	1019.3 (0.9)	0.6 (0.7)	0.95 (0.10)
	$\lvert X\rvert = 2048, \beta = \beta_0 = 0.0032, \lceil\beta \cdot \lvert X\rvert\rceil = 7$				
T	0.3 (0.6)	0.0 (0.0)	120.5 (0.7)	0.1 (0.5)	n/a
B	7.5 (0.9)	0.0 (0.0)	2037.3 (1.7)	3.1 (1.4)	0.82 (0.14)
K	7.5 (0.7)	0.0 (0.0)	2037.0 (1.7)	3.5 (1.5)	0.80 (0.12)
	$\lvert X\rvert = 512, \beta = 5\beta_0 = 0.0162, \lceil\beta \cdot \lvert X\rvert\rceil = 9$				
T	2.2 (1.0)	0.0 (0.0)	118.7 (1.0)	0.1 (0.3)	n/a
B	9.1 (0.4)	0.0 (0.0)	502.4 (0.9)	0.6 (0.9)	0.97 (0.06)
K	9.0 (0.0)	0.0 (0.0)	502.4 (0.9)	0.7 (0.9)	0.97 (0.06)
	$\lvert X\rvert = 512, \beta = 10\beta_0 = 0.0323, \lceil\beta \cdot \lvert X\rvert\rceil = 17$				
T	4.0 (1.5)	0.0 (0.0)	117.0 (1.5)	0.0 (0.0)	0.96 (0.08)
B	17.2 (0.4)	0.0 (0.0)	493.8 (0.9)	1.1 (0.9)	0.96 (0.04)
K	17.2 (0.4)	0.0 (0.0)	493.9 (0.8)	0.9 (0.7)	0.96 (0.04)

5.2.2 Scenario 2: Measurement of ad conversion revenue

In this scenario, (i) the attacker invokes PSI-SUM many times (i.e., 24 per day $\times$ 5 days), and (ii) the features used by the attacker are *not* strongly correlated with the set bias.

According to Figure 4, our two attacks still outperform the toy attack in terms of leakage speed and the number of the target elements whose set membership is determined in the given PSI-SUM invocations. Here, the toy attack even fails to identify any set member if only a few target elements appeared in the victim's snapshot set at the beginning of the attack (e.g., $|X| \in \{512, 1024, 2048\}$ and $\beta = 0.0032$). In contrast, our two attacks can still spot set members in this case. We see from Table 5 that the number of obtained set members roughly equals the number of the target elements in the initial snapshot set. Since the victim's set grows over time (Figure 2b), there is no false positive in the obtained set members as expected. The above analysis shows that using our two attacks, the attacker can cause a more severe set membership leakage in this scenario.

In contrast to COVID-19 contact tracing, our two attacks cause similar set membership leakage in this scenario. More concretely, they have comparable leakage speed (Figure 4) and almost the same accuracy (Table 5). These results are due to the weak correlation between the used features and the set bias. The demographic features provide little help in this scenario since most ads are primarily based on personal interest instead of demographic features. When only a few target elements are set members (e.g., $|X| \in \{512, 1024, 2048\}$

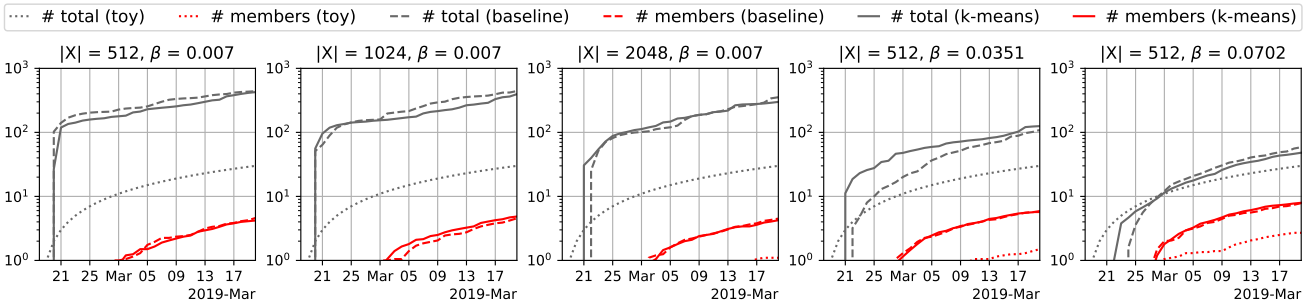


Figure 5: Set membership leakage in the measurement of ad conversion lift, measured in # total and # members.

Table 6: # total breakdown at the end of the attack in the measurement of ad conversion lift. T: toy attack, B: baseline attack, K: feature-aware attack instantiated with `k-means`.

	# TP	# FP	# TN	# FN	% initial TP
	$\|X\| = 512, \beta = \beta_0 = 0.0070, \lceil\beta \cdot \|X\|\rceil = 4$				
T	0.3 (0.5)	0.0 (0.0)	29.2 (1.0)	0.5 (0.8)	n/a
B	4.6 (0.7)	0.0 (0.0)	422.8 (81.7)	11.8 (4.1)	0.40 (0.20)
K	4.2 (1.1)	0.0 (0.0)	409.4 (101.7)	11.7 (4.4)	0.50 (0.25)
	$\|X\| = 1024, \beta = \beta_0 = 0.0070, \lceil\beta \cdot \|X\|\rceil = 8$				
T	0.9 (0.7)	0.0 (0.0)	28.9 (0.7)	0.1 (0.4)	n/a
B	4.5 (0.8)	0.0 (0.0)	423.6 (162.0)	14.0 (7.6)	0.38 (0.17)
K	4.8 (1.1)	0.0 (0.0)	377.5 (236.0)	13.1 (9.0)	0.41 (0.19)
	$\|X\| = 2048, \beta = \beta_0 = 0.0070, \lceil\beta \cdot \|X\|\rceil = 15$				
T	1.1 (0.7)	0.0 (0.0)	28.7 (0.8)	0.2 (0.4)	n/a
B	4.5 (0.7)	0.0 (0.0)	337.9 (153.9)	11.3 (8.2)	0.40 (0.23)
K	4.2 (1.2)	0.0 (0.0)	283.6 (186.7)	11.4 (7.5)	0.38 (0.24)
	$\|X\| = 512, \beta = 5\beta_0 = 0.0351, \lceil\beta \cdot \|X\|\rceil = 18$				
T	1.5 (1.4)	0.0 (0.0)	28.2 (1.4)	0.3 (0.6)	n/a
B	5.7 (0.6)	0.0 (0.0)	99.5 (36.9)	3.8 (3.5)	0.71 (0.26)
K	5.9 (0.9)	0.0 (0.0)	113.8 (73.1)	5.6 (5.8)	0.76 (0.17)
	$\|X\| = 512, \beta = 10\beta_0 = 0.0702, \lceil\beta \cdot \|X\|\rceil = 36$				
T	2.7 (1.5)	0.0 (0.0)	26.8 (1.5)	0.5 (0.7)	n/a
B	7.8 (1.8)	0.0 (0.0)	48.6 (16.7)	2.2 (2.5)	0.85 (0.12)
K	8.0 (1.7)	0.0 (0.0)	38.8 (16.1)	1.4 (1.2)	0.89 (0.09)

and $\beta = 0.0032$), Algorithm 3 cannot use the features to early separate set members from others and place them at the tree nodes of low depth. The higher the node depth, the lower the leakage speed. The above discussion explains why the feature-aware attack does not work better than the baseline attack in the early stage.

However, the small number of set members also means that the attacker can find so many members in the limited number of PSI-SUM invocations. We find that the number of the set members found by Algorithm 1 must be not less than the initial number, i.e., $\lceil\beta \cdot |X|\rceil$. The reason is that (i) there is no false positive for an increasing set of the victim, (ii) the underestimation from Inequality 3 will not reduce the current number of set members, and (iii) the number of found set members equals the current number of set members by the time Algorithm 1 terminates after *sufficient* invocations. Certainly, some found set members may appear in a snapshot set after the beginning of the attack rather than the initial one. The above analysis is consistent with the results in Table 5.

5.2.3 Scenario 3: Measurement of ad conversion lift

This scenario has the following characteristics: (i) the attacker can only invoke Private-ID *a few* times (i.e., 1 per day $\times$ 30 days), and (ii) the features used by the attacker is *not* strongly correlated with the set bias. These characteristics are the most unfavorable ones for set membership inference attacks.

Due to the limited number of Private-ID invocations, the following phenomena are more apparent in Figure 5. First, the number of non-members found in the feature-aware attack can be less than that in the baseline attack. This phenomenon is also observed in COVID-19 contact tracing. It happens in this scenario since the attack is still in its early stage until the last Private-ID invocation. In this stage, the number of found non-members is less stable. Second, even if the attacker chooses more target elements, the number of found set members cannot be increased like the other scenarios. This is due to the saturation of leakage speed for the same β. Third, our attacks start to determine set membership later than the toy attack. The reason is that our attacks require several invocations to finish one DFS. However, this latency is short.

Even with the latency, our two attacks cause a more severe set membership leakage than the toy attack in general (see Figure 5 and Table 6). Here, we again find that the toy attack fails to identify any set member like our attacks when the proportion β is small. In this case, it is unlikely for the toy attacker to choose at least one set member in the limited Private-ID invocations. In contrast, this number is greater than the tree height in our attacks. Therefore, the attacker can finish at least one DFS pass and find a set member(s).

Similar to the other advertising scenario, the weak correlation between the features and the set bias makes our attacks have comparable performance in identifying set members.

6 Discussion

6.1 Real-world Implications of Our Results

COVID-19 contact tracing. In essence, our set membership inference attacks narrow the range of the tokens of infected patients. For example, the probability that the attacker can guess one set member in its set of target tokens is $\lceil \beta \cdot |X| \rceil / |X|$, which is small. In contrast, our attacks significantly improve this probability since the attacker can simply pick one of the target elements diagnosed as set members (including true positives and false positives). We can see this improvement from Table 4. Moreover, the higher numbers of true positives indicate that our attacks allow the attacker to guess more tokens in the health authority's set.

The attacker can misuse this advantage in guessing the tokens of infected patients to identify these patients. Note that the attack [14] that links tokens to their associated persons is known in the token-based COVID-19 contact tracing systems that do NOT hide intersecting tokens (e.g., Apple and Google's system [12]). This linkage attack collects personal tokens through mobile phones placed in some locations and records (e.g., via cameras) the person(s) showing up at the moment when each token was received. The attacker can deduce some infected patients from this record and the revealed tokens in the health authority's set. Although Epione and Catalic try to address the linkage attack by using PSI-CA to reveal only the number of intersecting tokens, our attacks show that the attacker's advantage in guessing the tokens of infected patients is still non-negligible. The linkage attack can still work in the two augmented systems by using our attacks as subroutines to get some tokens in the health authority's set. The attacker's ability to identify any infected patient is dangerous. It was reported that the patients had suffered severe harassment [2, 3, 4, 5].

Measurement of ad conversion revenue. In this scenario, our attacks can significantly narrow the range of ad clickers' `personal_id`s from the following aspects. First, realistic ad click-through rates (CTRs) [6] ensure that (i) a large proportion of target `personal_id`s are unlikely to be in the victim's set at the beginning of the attack, and (ii) the change of this set is unlikely to make some target elements that were not originally set members appear in the victim's set subsequently. Second, the realistic number of protocol invocations allows the attacker to find as many set members from its target elements as there were at the beginning of the attack. For example, there were 7 target elements in the victim's set at the beginning, and the attacker found 7.5 set members on average by the end of the attack. Third, the found set members are perfectly correct since the attacker can make the victim's set grow by setting a large time window. The above arguments show that our attacks can infer member `personal_id`s with very high precision and recall.

The attacker (i.e., the advertiser) can use the inferred member `personal_id`s to deduce the interest of the persons associated with these `personal_id`s. The reasons for this interest disclosure are that (i) clicking the ad is a sign showing the ad clicker's interest in the advertised product [54], (ii) our attacks infer some ad clickers' `personal_id`s in this scenario, and (iii) the advertiser can translate these `personal_id`s into their associated persons. Although some ad clickers may end up purchasing the product and thus implicitly agree to reveal their interest to the advertiser, there are also those who *have clicked the ad but did not make a purchase*. In this scenario, most ad clickers fall into the latter category. From a privacy perspective, the advertiser should never collect the interest of these ad clickers since they have not given explicit consent to the collection of their interest. This unauthorized interest collection is known to be a cause of uncomfortable personalized recommendations [9]. When the advertised product is privacy-sensitive (e.g., HIV home test kits), it is also a privacy threat that the advertiser learns the interest of the persons who did not even make a purchase.

Measurement of ad conversion lift. In this scenario, the effect of our attacks on narrowing the range of ad viewers' `personal_id`s is not as significant as its counterpart in the previous scenario. The reason is that realistic ad view-through rates (VTRs) [8] can cause a higher β and such a change in the victim's set that some non-member `personal_id`s at the beginning of the attack will be included in the victim's set subsequently. However, the found set members are still perfectly correct. In other words, our attacks can still infer member `personal_id`s with very high precision.

In this scenario, the set membership leakage also allows the attacker (i.e., advertiser) to infer the interest of the persons associated with target `personal_id`s. Ideally, the advertiser should not know which persons are associated with the target `personal_id`s in the publisher's set. The reason is that the advertiser knows that the publisher categorized these persons as interested in the ad and invited them to the A/B testing. Thus, revealing whether a `personal_id` is in the publisher's set immediately leaks the associated person's interest. However, our attacks show that the leakage of the `personal_id`s in the publisher's set does exist. Consequently, the advertiser can covertly use the publisher to find the persons interested in the ad and acquire some of them. This idea of inferring personal interest with the publisher's help is similar to [54]. Here, merely viewing the ad without purchasing does not mean that the person allows the advertiser to know its personal interest. Such persons should account for the vast majority of all persons who have or would have seen the ad in the testing.

6.2 Possible Defenses

Here, we discuss the possible defenses from a technical perspective and the limitations.

Limiting the number of protocol invocations. A simple

defense is to limit the number of protocol invocations, i.e., the number of intersection sizes revealed to the attacker.

However, it is unclear how to properly choose the threshold of this number. The choice of the threshold should depend on (i) the target elements, (ii) the attacker's background knowledge, (iii) the allowed set membership leakage in the scenario, (iv) the frequency of collaboration tasks in the scenario, and (v) the number of protocol invocations per task. Although (iii), (iv), and (v) can be estimated, the victim is unaware of (i) and (ii). Thus, the victim can only use a heuristic threshold.

This defense cannot fully prevent set membership leakage unless the threshold is very small. Our evaluation in the measurement of ad conversion lift confirms this conclusion. The conclusion is also applied to the other two scenarios. In most scenarios, allowing only a few protocol invocations will harass the functionality of high-level systems.

Auditing intersection size. If the attacker launches a set membership inference attack according to Algorithm 1, there will be a size pattern in the sequence of revealed intersection sizes. Therefore, the victim can detect our attacks by auditing these intersection sizes to find out the pattern.

However, this defense faces some challenges. First, it is non-trivial for intersection-size-revealing protocols to return a precise intersection size to both entities *while preserving the provable security against a malicious attacker* (e.g., see [61, 68]). Without the sequence of intersection sizes, the victim cannot check the size pattern. Second, the attacker can make malicious invocations intermittently and, in each invocation, pad some elements whose set membership is known to skew the intersection size. It is unclear whether this defense will work if the attacker adopts the strategy.

Auditing the size of the attacker's input table. In most intersection-size-revealing protocols, the victim is allowed to see the size of the attacker's input table. In other words, the victim learns how many elements are input by the attacker. Since there is also a size pattern of the subsets used in Algorithm 1, it seems possible for the victim to detect the attacks by auditing the table size in each invocation.

Unfortunately, the attacker can easily bypass this defense. Suppose that Algorithm 1 asks the attacker to input $\{T_1, ..., T_n\}$ to the protocol in n invocations sequentially. To hide the size pattern of these subsets, the attacker constructs another n subsets $\{D_1, ..., D_n\}$ such that (i) these new subsets are of the same size, (ii) each D_i is derived from T_i by padding T_i with $|D_i| - |T_i|$ dummy elements. These dummy elements will appear in the victim's set Y with negligible probability. Thus, by inputting D_i instead of T_i to the protocol, the attacker can still learn $|T_i \cap Y| = |D_i \cap Y|$ while hiding the actual size of T_i. Due to the security of intersection-size-revealing protocols, the victim cannot tell whether there are dummy elements in the attacker's input table. Therefore, auditing the size of the attacker's table does not help.

Applying differential privacy. Differential Privacy (DP) [37, 39, 59] is a rigorous definition of privacy. It ensures that whether a record is a part of a mechanism's input does not significantly affect the output distribution of the mechanism. This significance is bounded by DP parameters (ε, δ). The high-level intuition behind the DP-based defense is to add noise to revealed intersection sizes. Such noisy intersection sizes are expected to reduce the accuracy of our attacks.

This DP-based defense works as follows. Recall that, given the attacker's set X of target elements and the victim's set Y, inferring the target elements' set membership is equivalent to reconstructing the intersection $I := X \cap Y$. This intersection can be represented as a binary vector $\mathsf{bin}(I) \in \{0,1\}^{|X|}$, where $\mathsf{bin}(\cdot)$ is a bijective mapping such that the i-th coordinate of $\mathsf{bin}(I)$ is one if and only if $x_i \in Y$. In Algorithm 1, the intersection size between a counterfeit set $Q \subseteq X$ and the set Y equals the inner product of $\mathsf{bin}(Q)$ and $\mathsf{bin}(I)$ since

$$\begin{aligned} |Q \cap Y| &= |(Q \cap X) \cap Y| = |Q \cap (X \cap Y)| \\ &= |Q \cap I| = \langle \mathsf{bin}(Q), \mathsf{bin}(I) \rangle. \end{aligned} \tag{5}$$

From the DP perspective, the attacker measures the binary string $\mathsf{bin}(I)$ using the query $\mathsf{bin}(Q)$ in each protocol invocation. [27, 39, 48, 64] showed that, if the number of invocations is limited (e.g., sublinear in $|X|$), adding small noise to the inner products in Equation 5 can achieve differential privacy.

However, this defense has the following limitations. First, the choice of DP parameters depends on the size $|X|$, which is unknown until the first protocol invocation. In other words, the two entities have difficulty in agreeing on how much additive noise is needed to reach the desired degree of differential privacy. Second, how to efficiently incorporate differential privacy into the intersection-size-revealing protocols is still an open problem. To date, no such protocol is known that can be used to replace those vulnerable to our attacks. Third, even if there is a differentially private alternative of the protocols, this alternative can only distort the set membership of each person rather than perfectly hiding it from the attacker. After several protocol invocations, it is still probable for the attacker to correctly infer the set membership of some persons. Future work is required to explore the relationship between concrete DP parameters and this probability.

6.3 Limitations

First, in our feature preprocessing (see Appendix B.2), the symptoms of the persons who are not infected with COVID-19 at the beginning of attack are set to `fever` $= 0$ and `cough` $= 0$. This preprocessing may distort the feature distribution of recorded persons since even uninfected persons can have fever or cough. However, such persons are expected to represent a small fraction of all uninfected persons, unless there is a concurrent epidemic of another disease that can cause these symptoms. Thus, we believe that this distortion is mild and will not significantly affect our results.

Second, Tencent's dataset was not collected from ad conversion lift campaigns. However, we note that the distribution of ad viewers in this dataset is similar to the distribution of tested persons in the measurement of ad conversion lift. The reason is that the tested persons are also qualified to see the ad, like the ad viewers in Tencent's dataset. Private-ID takes as input all these persons regardless of whether they have actually seen the ad (i.e., in the treatment group). In essence, the two distributions are of the persons who are identified by the publisher as being interested in the ad. Therefore, it makes sense for us to use the distribution of ad viewers in Tencent's dataset as the distribution of tested persons. The analysis results of Tencent's dataset are supposed to be consistent with those in the measurement of ad conversion lift.

Third, this work considers, for each scenario, the data distribution under certain conditions (e.g., time span and data source). Our results show how severe set membership leakage is under these data distributions. We stress that the set membership leakage caused by our attacks changes with the underlying data distribution in a scenario. When using our attacks to quantify the set membership leakage in a given scenario, one should take into account the current data distribution, instead of simply applying our analysis results.

6.4 Revisiting Set Membership Leakage in Intersection-size-revealing Protocols

We note that, in realistic scenarios, the set membership leakage from intersection-size-revealing protocols may be more severe than that studied in this work. The reason is that the leakage in this work is only resulted from intersection size returned by these protocols. However, some of intersection-size-revealing protocols reveal more information than this size. Such additional information may help the attacker to infer set membership more efficiently.

An example is PSI-SUM, which additionally aggregates the values associated with the intersecting indices and reveals this aggregate value. In PSI-SUM, the attacker can counterfeit a table $\{(y_i, v_i)\}_{y_i \in Y, |Y|=n}$ where the sequence $(v_1, ..., v_n)$ is *superincreasing* (i.e., $v_{i+1} > \sum_{1 \le j \le i} v_j$ for $1 \le i < n$). Given the victim's set Y, PSI-SUM reveals the intersection size $|\{k_i\}_i \cap Y|$ and the following subset sum to the attacker

$$\sum_{1 \le i \le n,\ k_i \in Y} v_i \equiv \sum_{1 \le i \le n} b_i \cdot v_i \pmod{p}, \tag{6}$$

where p is a public large modulus and $b_i = 1$ if and only if $k_i \in Y$. Inferring each k_i's set membership regarding the victim's set Y is equivalent to determining b_i in Equation 6, or rather, solving the subset sum problem. It is known in the literature [60] that, when the sequence is superincreasing, the attacker can easily solve this subset sum problem by locally running a greedy algorithm. That is, the attacker can even infer the set membership of all target elements with only ONE PSI-SUM invocation, if n is small. If n is super-logarithmic in p so that the aggregate value wrap out p, the attacker can instead divide n elements into several bins and solve the problem for each bin. In this way, the attacker can determine all elements' set membership using at most $n/\log p$ invocations.

Some other intersection-size-revealing protocols, such as Private-ID or (not deployed) circuit-based Private Set Intersection [55], are combined with downstream secure computation that additionally returns the statistics needed by high-level systems. It is left for future work to see whether the attacker can use these statistics in conjunction with revealed intersection size to cause more severe set membership leakage.

7 Related Work

Recall that, as shown in Equation 5, our attacks essentially aim to reconstruct the binary representation $\mathsf{bin}(I) \in \{0,1\}^{|X|}$ of the intersection I at the beginning of attack. The works most relevant to this work are those investigating how many inner product responses are needed to reconstruct a binary database $D = (d_1, ..., d_n) \in \{0,1\}^n$. The first reconstruction attack by Dinur and Nissim [32] showed that, if the noise added to each response is $o(\sqrt{n})$, the database D can be approximately recovered from the noisy version of $O(n \log^2 n)$ responses $\langle Q_i, D \rangle$, where the query $Q_i \in \{0,1\}^n$. Dwork et al. [38] showed that the attacker can reconstruct almost the whole database if there are $O(n)$ noisy responses and at least 0.761 fraction of them are with $o(\sqrt{n})$ bounded noise. As noted in [38], the reconstruction problem also has a connection with compressed sensing [30, 33, 34], which aims to find possible solutions to undetermined linear systems.

This work differs from the related ones in the following aspects. First, our attacks use *noise-free* responses (aka intersection sizes) given by intersection-size-revealing protocols, while the related works consider noisy ones from a differentially private mechanism. Second, the intersection (aka the vector of set membership bits) is recovered "bit-by-bit" in our attacks. However, the attacker in the related works needs to run a reconstruction algorithm after receiving all responses. Third, and most importantly, this work use features correlated with set bias to produce queries while the related works focus on random queries. We show that set bias, which is not considered in the existing works, can improve reconstruction efficiency. Fourth, the related works are of theoretical interest and pay little attention to the database reconstruction in realistic scenarios, especially when the database is dynamic.

8 Conclusion

This paper initiates the first study on the set membership leakage in intersection-size-revealing protocols. This leakage results from the intersection sizes revealed by these protocols. We propose the baseline and feature-aware attacks to show that the attacker can infer some target elements' membership

of the victim's set. In particular, our feature-aware attack can exploit set bias to aggravate the leakage.

In addition to the theoretical analysis, we evaluate our attacks in three realistic scenarios: (i) COVID-19 contact tracing, (ii) the measurement of ad conversion revenue, and (iii) the measurement of ad conversion lift. The results show that our two attacks can cause more severe leakage than a heuristic attack in these scenarios. Notably, the feature-aware attack outperforms the baseline attack in COVID-19 contact tracing since there is a non-negligible bias in the victim's set. We also discuss the real-world implications of our results and some possible defenses against our attacks.

Note that some intersection-size-revealing protocols (e.g., PSI-CA, PSI-SUM, and Private-ID) have attracted much attention from the industry and can be used in many collaborative computation scenarios. This work is helpful in analyzing the privacy issues raised by the protocols in their scenarios.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. This work is supported by the National Natural Science Foundation of China (No. 62032012). Yan Jia is also funded by China Postdoctoral Science Foundation (No. 2021M691673) and National Natural Science Foundation of China (No. 62102198).

References

[1] A/B testing. https://en.wikipedia.org/wiki/A/B_testing.

[2] Covid-19 patients suffered harassment and abuse. https://www.rnz.co.nz/news/national/428581/covid-19-patients-suffered-harassment-and-abuse.

[3] Argentina's coronavirus patients, medical workers harassed. https://www.aljazeera.com/features/2020/4/28/argentinas-coronavirus-patients-medical-workers-harassed.

[4] Coronavirus patients shamed and harassed in Nepal. https://www.dw.com/en/coronavirus-patients-shamed-and-harassed-in-nepal/a-55017037.

[5] 'Gathering to kill me': Coronavirus patients in Haiti fear attacks, harassment. https://www.reuters.com/article/uk-health-coronavirus-haiti-stigma-insig-idUKKBN22N1BD.

[6] Clickthrough rate (CTR): Definition. https://support.google.com/google-ads/answer/2615875.

[7] Israeli Ministry of Health official covid-19 dataset. https://data.gov.il/dataset/covid-19.

[8] View rate: Definition. https://support.google.com/google-ads/answer/6293479.

[9] How Target figured out a teen girl was pregnant before her father did. https://www.forbes.com/sites/kashmirhill/2012/02/16/how-target-figured-out-a-teen-girl-was-pregnant-before-her-father-did.

[10] B2c contact data - global contact list - reach b2c audience globally with our verified 200m+ b2c contact data. https://datarade.ai/data-products/b2c-contact-data-global-contact-list-reach-b2c-audience-g-blue-mail-media.

[11] Us consumers database | b2c contact data by every market media. https://datarade.ai/data-products/consumers-b2c-every-market-media.

[12] Privacy-preserving contact tracing. https://covid19.apple.com/contacttracing.

[13] Conversion lift. https://www.facebook.com/business/m/one-sheeters/conversion-lift.

[14] Does covid-19 contact tracing pose a privacy risk? Your questions, answered. https://www.wired.com/story/apple-google-contact-tracing-strengths-weaknesses/.

[15] Which US states and cities have the highest ad click rates? https://www.marketingprofs.com/charts/2016/29556/which-states-and-cities-have-the-highest-ad-click-rates.

[16] Mutual information. https://en.wikipedia.org/wiki/Mutual_information.

[17] Guidance for businesses and employers responding to coronavirus disease 2019 (covid-19). https://www.cdc.gov/coronavirus/2019-ncov/community/guidance-business-response.html.

[18] Managing covid-19 risks: temperature screening. https://www.worksafe.vic.gov.au/managing-coronavirus-covid-19-risks-temperature-screening.

[19] Coronavirus: How much does your boss need to know about you? https://www.bbc.com/news/business-53109207.

[20] Employers rush to adopt virus screening. the tools may not help much. https://www.nytimes.com/2020/05/11/technology/coronavirus-worker-testing-privacy.html.

[21] Ford covid-19 screening attestation. https://covid19survey.ford.com/uk.

[22] Sample employee covid-19 health screening questionnaire. https://www.osha.gov/sites/default/files/publications/OSHA4132.pdf.

[23] Google Ads. Paving the path to proven success: Your playbook on experimentation. https://www.thinkwithgoogle.com/_qs/documents/11543/playbook_of_marketing_experiment_principles_methodology_and_tools.pdf.

[24] Google Ads. Make every marketing dollar count with attribution and lift measurement. https://blog.google/products/ads-commerce/attribution-lift-measurement/.

[25] Rakesh Agrawal, Alexandre Evfimievski, and Ramakrishnan Srikant. Information sharing across private databases. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 86–97, 2003.

[26] AppsFlyer. [Legacy] Tencent AMS (GDT & Wechat MP) campaign configuration. https://support.appsflyer.com/hc/en-us/articles/360000264689-Tencent-AMS-GDT-WeChat-MP-campaign-configuration.

[27] Aditya Bhaskara, Daniel Dadush, Ravishankar Krishnaswamy, and Kunal Talwar. Unconditional differentially private mechanisms for linear queries. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1269–1284, 2012.

[28] Prasad Buddhavarapu, Andrew Knox, Payman Mohassel, Shubho Sengupta, Erik Taubeneck, and Vlad Vlaskin. Private matching for compute. *IACR Cryptol. ePrint Arch.*, 2020. https://eprint.iacr.org/2020/599.

[29] Facebook Business Help Center. About advanced matching for web. https://www.facebook.com/business/help/611774685654668?id=1205376682832142.

[30] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001.

[31] Emiliano De Cristofaro, Paolo Gasti, and Gene Tsudik. Fast and private computation of cardinality of set intersection and union. In *International Conference on Cryptology and Network Security*, pages 218–231. Springer, 2012.

[32] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 202–210, 2003.

[33] David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

[34] David L Donoho. For most large underdetermined systems of linear equations the minimal ℓ_1-norm solution is also the sparsest solution. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 59(6): 797–829, 2006.

[35] Thai Duong, Duong Hieu Phan, and Ni Trieu. Delegated psi cardinality & contact tracing. https://github.com/nitrieu/delegated-psi-ca.

[36] Thai Duong, Duong Hieu Phan, and Ni Trieu. Catalic: Delegated psi cardinality with applications to contact tracing. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 870–899. Springer, 2020.

[37] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–503. Springer, 2006.

[38] Cynthia Dwork, Frank McSherry, and Kunal Talwar. The price of privacy and the limits of lp decoding. In *Proceedings of the thirty-ninth annual ACM Symposium on Theory of Computing*, pages 85–94, 2007.

[39] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[40] Facebook. Private lift games pseudocode. https://github.com/facebookresearch/fbpcs/blob/main/docs/PrivateLift.md.

[41] Facebook. Private randomized controlled trial. https://github.com/facebookresearch/fbpcf/blob/main/docs/PrivateRCT.md.

[42] Facebook. What are privacy-enhancing technologies (PETs) and how will they apply to ads? https://about.fb.com/news/2021/08/privacy-enhancing-technologies-and-ads.

[43] Ellis Fenske, Akshaya Mani, Aaron Johnson, and Micah Sherr. Distributed measurement with private set-union cardinality. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2295–2312, 2017.

[44] Michael J Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *International conference on the theory and applications of cryptographic techniques*, pages 1–19. Springer, 2004.

[45] Michael J Freedman, Carmit Hazay, Kobbi Nissim, and Benny Pinkas. Efficient set intersection with simulation-based security. *Journal of Cryptology*, 29(1):115–155, 2016.

[46] Google. Helping organizations do more without collecting more data. https://security.googleblog.com/2019/06/helping-organizations-do-more-without-collecting-more-data.html.

[47] Google. Private join and compute. https://github.com/Google/private-join-and-compute.

[48] Moritz Hardt and Kunal Talwar. On the geometry of differential privacy. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 705–714, 2010.

[49] Google Ads Help. About conversion windows. https://support.google.com/google-ads/answer/3123169.

[50] Google Ads Help. About enhanced conversions. https://support.google.com/google-ads/answer/9888656.

[51] Search Ads 360 Help. About hourly reporting. https://support.google.com/searchads/answer/6293151.

[52] Mihaela Ion, Ben Kreuter, Ahmet Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, Mariana Raykova, David Shanahan, and Moti Yung. On deploying secure computing: Private intersection-sum-with-cardinality. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 370–389. IEEE, 2020.

[53] Lea Kissner and Dawn Song. Privacy-preserving set operations. In *Annual International Cryptology Conference*, pages 241–257. Springer, 2005.

[54] Aleksandra Korolova. Privacy violations using microtargeted ads: A case study. In *2010 IEEE International Conference on Data Mining Workshops*, pages 474–482. IEEE, 2010.

[55] Phi Hung Le, Samuel Ranellucci, and S Dov Gordon. Two-party private set intersection with an untrusted third party. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2403–2420, 2019.

[56] Bisheng Liu and Urs Hengartner. Privacy-preserving social recommendations in geosocial networks. In *2013 Eleventh Annual Conference on Privacy, Security and Trust*, pages 69–76. IEEE, 2013.

[57] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[58] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[59] Frank D McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 19–30, 2009.

[60] Ralph Merkle and Martin Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE transactions on Information Theory*, 24(5):525–530, 1978.

[61] Peihan Miao, Sarvar Patel, Mariana Raykova, Karn Seth, and Moti Yung. Two-sided malicious security for private intersection-sum with cardinality. In *Annual International Cryptology Conference*, pages 3–33. Springer, 2020.

[62] Mahnush Movahedi, Benjamin M Case, James Honaker, Andrew Knox, Li Li, Yiming Paul Li, Sanjay Saravanan, Shubho Sengupta, and Erik Taubeneck. Privacy-preserving randomized controlled trials: A protocol for industry scale deployment. In *Proceedings of the 2021 on Cloud Computing Security Workshop*, pages 59–69, 2021.

[63] Marcin Nagy, Emiliano De Cristofaro, Alexandra Dmitrienko, N Asokan, and Ahmad-Reza Sadeghi. Do i know you? efficient and privacy-preserving common friend-finder protocols and applications. In *Proceedings of the 29th Annual Computer Security Applications Conference*, pages 159–168, 2013.

[64] Aleksandar Nikolov, Kunal Talwar, and Li Zhang. The geometry of differential privacy: the sparse and approximate cases. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 351–360, 2013.

[65] Ben Niu, Xiaoyan Zhu, Jie Liu, Zan Li, and Hui Li. Weight-aware private matching scheme for proximity-based mobile social networks. In *2013 IEEE Global Communications Conference (GLOBECOM)*, pages 3170–3175. IEEE, 2013.

[66] OpenMined. Private set intersection for covid-19 corona contact tracing apps. https://blog.openmined.org/private-set-intersection.

[67] OpenMined. Covid alert. https://github.com/OpenMined/covid-alert.

[68] Peter Rindal and Mike Rosulek. Malicious-secure private set intersection via dual execution. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1229–1242, 2017.

[69] Ni Trieu, Kareem Shehata, Prateek Saxena, Reza Shokri, and Dawn Song. Epione: Lightweight contact tracing with strong privacy. https://sunblaze-ucb.github.io/privacy/projects/epione.html.

[70] Ni Trieu, Kareem Shehata, Prateek Saxena, Reza Shokri, and Dawn Song. Epione: Lightweight contact tracing with strong privacy. *IEEE Data Eng. Bull.*, 43(2):95–107, 2020.

[71] Twitter. Optimizing campaigns with conversion lift reports. https://blog.twitter.com/en_us/a/2016/optimizing-campaigns-with-conversion-lift-reports.

[72] Jaideep Vaidya and Chris Clifton. Secure set intersection cardinality with application to association rule mining. *Journal of Computer Security*, 13(4):593–622, 2005.

[73] Yazeed Zoabi, Shira Deri-Rozov, and Noam Shomron. Machine learning-based prediction of covid-19 diagnosis based on symptoms. *npj digital medicine*, 4(1):1–5, 2021.

A A Note on Facebook's Private-ID

In the measurement of ad conversion lift, there is a subtle issue in Facebook's Private-ID protocol when the union of the two parties' sets is dynamic. The issue is that the assigned tags for the t-th union cannot be reused for the $(t+1)$-th union. Otherwise, it will *trivially* leak set membership.

Consider a counterexample where the $(t+1)$-th union has only one more element x than the t-th union. If the tags of the elements in the t-th union remain the same, and the two parties only assign a fresh tag to x, the party holding x can check whether x's tag equals any *old* tag for the t-th union. If so, x must be a member of the other party's set. In contrast, if they re-invoke Private-ID to assign fresh tags to *all* elements in the $(t+1)$-th union, no set membership will be leaked.

This counterexample indicates that, if the union of the two parties' sets is dynamic (e.g., in the measurement of ad conversion lift), Private-ID should be re-invoked for the current two sets, and thus the attacker can learn an intersection size from each Private-ID invocation.

B More Details about Experimental Setup

B.1 On the Choices of β

We choose the values of β for the following reasons. $\beta = \beta_0$ corresponds to that, at the beginning of attack, the attacker uniformly samples $|X|$ target elements from the population without replacement. The number Z of set members follows the hypergeometric distribution

$$\Pr[Z=k] = \binom{|A|}{k} \cdot \binom{|U|-|A|}{|X|-k} \Big/ \binom{|U|}{|X|},\ k = 0,1,...,|X|,$$

and the expectation $\mathbb{E}(\beta) = \mathbb{E}(Z)/|X| = |A|/|U| = \beta_0$.

However, in reality the attacker will choose its set of target elements on purpose. Recall that the more elements in this set are the members of the victim's set, the more likely it is that the attack will cause a severe leakage. Therefore, the attacker is motivated to make its set contain as many members on the victim's set as possible. In COVID-19 contact tracing, it can do this by deliberately collecting personal tokens from epidemic areas, where its mobile phone is more likely to receive the tokens of COVID-19 patients. In the two advertising scenarios, the advertiser can target its ad [15] to the areas with higher click-through rates (CTRs) [6] or view-through rates (VTRs) [8] and choose target `personal_ids` from these areas. Since it is known that such rates vary from area to area [15], the proportion β resulted from this strategy may have a higher value than β_0, which is based on the whole population.

Therefore, we recommend that the victim should consider a conservative (or rather, larger) value of the proportion when empirically evaluating the leakage of its set. This is to estimate the worst-case set membership leakage. In our evaluation, we also consider two such values, i.e., $5\beta_0$ and $10\beta_0$.

B.2 Feature Preprocessing

The main purpose of our feature preprocessing is to prevent `k-means`, which was called at the beginning of attack, from using the personal features that have not yet appeared at this moment. In COVID-19 contact tracing, it is almost impossible for the feature-aware attacker to know, on April 4, 2020, the symptoms of patients who were diagnosed with COVID-19 in the future. Here, we assume that the interval between when a person had COVID-19 symptoms and when it got test result is negligible. Under this assumption, those diagnosed with COVID-19 in the future are amended to have no symptom (i.e., `fever` $= 0$ and `cough` $= 0$) at the beginning of attack. For the two online advertising scenarios, we assume that the considered features are fixed during attack. Thus, we do not amend the features in the two datasets.

In addition to the above preprocessing, we encode personal features using standard techniques.