# AnonPSI: An Anonymity Assessment Framework for PSI

Bo Jiang
TikTok Inc.
bojiang@tiktok.com

Jian Du
TikTok Inc.
jian.du@tiktok.com

Qiang Yan
TikTok Inc.
yanqiang.mr@tiktok.com

***Abstract*—Private Set Intersection (PSI) is a widely used protocol that enables two parties to securely compute a function over the intersected part of their shared datasets and has been a significant research focus over the years. However, recent studies have highlighted its vulnerability to Set Membership Inference Attacks (SMIA), where an adversary might deduce an individual's membership by invoking multiple PSI protocols. This presents a considerable risk, even in the most stringent versions of PSI, which only return the cardinality of the intersection. This paper explores the evaluation of anonymity within the PSI context. Initially, we highlight the reasons why existing works fall short in measuring privacy leakage, and subsequently propose two attack strategies that address these deficiencies. Furthermore, we provide theoretical guarantees on the performance of our proposed methods. In addition to these, we illustrate how the integration of auxiliary information, such as the sum of payloads associated with members of the intersection (PSI-SUM), can enhance attack efficiency. We conducted a comprehensive performance evaluation of various attack strategies proposed utilizing two real datasets. Our findings indicate that the methods we propose markedly enhance attack efficiency when contrasted with previous research endeavors. The effective attacking implies that depending solely on existing PSI protocols may not provide an adequate level of privacy assurance. It is recommended to combine privacy-enhancing technologies synergistically to enhance privacy protection even further.**

## I. Introduction

Private Set Intersection (PSI) protocols allow two parties to securely compute a function over the intersection of their datasets without directly revealing the intersection itself [3], [6], [11] and have gained significant interest in the industry. Sensitive information within the intersection, such as individual identities, attributes, and memberships at the other party, remains unrevealed.

While the intersection set is not directly disclosed, most two-party secure computation protocols disclose the intersection size may inadvertently leak membership information [14]. PSI-CA (Private Set Intersection-Cardinality) is one such protocol that releases the cardinality of the intersection to one party [13], [12]. This protocol serves as an essential building block for various applications, such as new friend recommendations in online social networks [30] and contagious disease tracking [7]. Another popular PSI protocol is PSI-SUM [16], [20]. In addition to the cardinality of the intersection, PSI-SUM also reveals the summation of payloads associated with members of the intersection. One application of PSI-SUM is the secure computation for advertising measurement between an advertiser and an ad provider. Similar PSI protocols including Private-ID [4] reveal the intersection size.

Recent research indicates that even the disclosure of the intersection cardinality can introduce vulnerabilities [14]. Intuitively, cardinality is not independent of the identities of individuals in the intersected set, especially when the size is small. Therefore, these intersection size revealing protocols enable an adversary to infer whether a targeted individual is in the intersection or not, which is also known as the membership information. Notably, individuals who intersect with the other party's set are classified as positive members, while those outside the intersection are classified as negative members. Depending on the application context, either or both of these positive/negative memberships can be sensitive from a privacy perspective as shown below. Consider, for example, a situation involving COVID-19 testing. Individuals who have tested positive are treated as sensitive information. Another scenario arises when politicians seek to advance their positions and persuade people to support a bill. They may examine existing datasets to identify those who abstained from voting. In this context, these non-voting individuals, representing negative members, are deemed sensitive. In a third scenario, such as advertising measurement [16], [20], the advertising company would be interested in both positive and negative memberships, which refer to an advertising conversion and non-advertising conversion, respectively. The motivation to determine user memberships serves two primary purposes: firstly, to tally positive members for the targeted advertising campaign, and secondly, to engage with negative members through a distinct advertising campaign.

A more efficient attacker can even deduce a subset of individuals' membership through a small number of protocol invocations. Such an attack is also known as Set Membership Inference Attack (SMIA). In certain scenarios, the inference of such information may lead to unintended consequences, such as discrimination, targeted advertising, or even social engineering attacks. An SMIA is initiated by the attacker being

Network and Distributed System Security (NDSS) Symposium 2024
26 February - 1 March 2024, San Diego, CA, USA
ISBN 1-891562-93-2
https://dx.doi.org/10.14722/ndss.2024.241279
www.ndss-symposium.org

one party participating in the PSI running protocol, through the continual submission of varying input sets. Upon observing the outputs, the adversary can infer the membership information of targeted elements.

On the other hand, an effective SMIA is useful in measuring the privacy protection guarantee provided by different PSI protocols and providing guidance in designing better and stricter protocols. The effectiveness of the SMIA is typically determined by the number of individuals' memberships inferred. Beyond the attack efficiency, to assess the protection provided by PSI, the attack algorithm should also be feasible and valid under certain counterattack measures embedded in the PSI protocol. For instance, many companies set daily limits for database queries from the public. For example, it allows only 15 queries per day [24]. Consequently, a feasible attack should be designed to be sufficiently efficient with a limited number of protocol invocation times. Moreover, strict PSI protocols are often fortified with privacy-preserving mechanisms, such as noise-adding mechanisms based on differential privacy [10], [9]. The attack should continue to function effectively against such noise with high confidence.

In this paper, our principal contributions can be summarized in three key areas:

We introduce two strategies for implementing membership inference attacks: deterministic and statistical, which take individuals' membership as deterministic values and random variables respectively. Efficient attack algorithms are proposed under each strategy, and we theoretically illustrate the performance guarantee of each (performance lower bound).

We exhibit the potential of employing auxiliary information to further augment the attack's effectiveness. In this regard, an efficient algorithm specifically tailored for PSI-SUM attacks is proposed for the first time. We amalgamate the attack algorithm in PSI-CA with an offline summation matching algorithm, N-SUM, thereby significantly reducing combination possibilities and enhancing attack efficiency.

We assess the proposed algorithms using two real-world datasets: COVID-tracking systems and TaoBao advertisements. We rigorously test the performance and effectiveness of these algorithms against these datasets, substantiating their practical applicability. This further implies that the PSI protocol alone may not ensure adequate privacy. It's advisable to integrate other privacy-enhancing technologies for strengthened privacy safeguards.

## II. Related works

The concept of Membership Inference Attack (MIA) was introduced by Shokri et al. in 2017 [25], marking a significant milestone in the field of machine learning security. In an MIA, attackers leverage machine learning model outputs such as prediction scores or confidence intervals to deduce whether a specific data point was part of the training set. Intriguingly, these attackers don't require direct access to the model's inner workings or training data; the model's query responses suffice, making these attacks practicable in real-world scenarios where models are often deployed as services. Nasr et al. [22] extended these attacks to collaborative deep learning models, underscoring potential risks in the burgeoning field of federated learning. Theoretical frameworks for comprehending and counteracting these attacks were furnished by Yeom et al. [29]. It's noteworthy, however, that MIAs typically infer a single individual's membership through one observation, while SMIA reasons multiple individual memberships through a sequence of observations. MIAs generally train dedicated models to emulate the target model's behavior, while in SMIA, the attacker alters the input for each protocol call and interprets the results from observations. Defense mechanisms against MIAs, such as adding noise to the model's outputs [15] or employing differential privacy mechanisms [2], [17], remain viable for SMIA.

A distinct but related line of research pertains to linkage attacks, which attempt to re-identify anonymized data by correlating it with other accessible datasets. Introduced by Latanya Sweeney in 2000 [27], linkage attacks gained notoriety when Sweeney demonstrated the possibility of re-identifying individuals in anonymized medical data using publicly available voter registration lists. The rise of big data and computational power has amplified the prominence of linkage attacks. Notably, in 2006, Narayanan and Shmatikov [21] de-anonymized Netflix's movie rating data by linking it with publicly available IMDb ratings. The emerging ubiquity of linkage attacks has underscored the need for robust anonymization techniques and highlighted the limitations of approaches like k-anonymity [19]. SMIA in contexts such as PSI bears similarities to linkage attacks, as it makes inferences using data from a dependent dataset. However, while linkage attacks typically focus on re-identifying individuals, SMIAs target membership information. In this paper, we present an attack algorithm for PSI-SUM that leverages side information, akin to a linkage attack.

Recently, Guo et al. in [14] proposed an attack that enhances the basic "toy attack" (submit one element at a time to the PSI protocol, and observe the membership) efficiency from $O(N)$ to $O(\log N)$ (protocol invocation times to infer $N$ individuals' membership). Nevertheless, this improved attack possesses several limitations:

1) PSI protocols often cap invocation frequencies within a specific number, necessitating highly efficient iteration-based attacks to garner enough information during the limited protocol calls. The method outlined in [14] is inapplicable when the allowable protocol call budget is constrained.
2) When the PSI incorporates privacy-preserving mechanisms, such as differential privacy, the method in [14] becomes impracticable due to its inability to cope with the resultant uncertainty in the attack process.
3) The aforementioned method focuses on one PSI scenario (PSI-CA), disregarding the side information that other applications like PSI-SUM can provide. This side information could prove instrumental during an attack.

In light of these limitations, our work aims to devise a more effective and efficient attack strategy that not only considers multiple PSI scenarios but is also capable of handling the uncertainties introduced by privacy-preserving mechanisms.

## III. Preliminaries

To provide a formal representation of the SMIA for PSI protocols, we define the notations, attack model, and parties involved in the PSI protocol execution. Then we introduce some baseline algorithms for SMIA.

### A. Problem formulation

In a two-party setting, let us represent the dataset of Party 1 as $X = \{x_1, \ldots, x_{|X|}\}$ and the dataset of Party 2 as $Y = \{y_1, \ldots, y_{|Y|}\}$. Here, we use $|\ |$ to denote the cardinality of a dataset. For instance, $|X|$ signifies the total number of elements in Party 1's dataset. The cardinality of the intersection between the two datasets is represented by $|X \setminus Y|$.

In the context of intersection-size-revealing protocols, the party that receives the intersection size obtains a measure of similarity between its own dataset and the other party's dataset. Given that the party has the freedom to select its own dataset for the protocol, it can strategically assess the other party's dataset according to its interests. A relevant question arises when the party has a multitude of target elements: Can the party determine the membership status of these target elements in the other party's dataset by taking advantage of its ability to measure similarity?

**Threat Model:** We assume the attacker is from Party 1, i.e., $X$ belongs to the attacker, and the attacker possesses the following capabilities. The attacker can participate in multiple intersection-size-revealing protocols as a party. During each protocol execution, the attacker can select its input and obtain the intersection size (and some side information if available, such as the summation in PSI-SUM) resulting from its input and $Y$. In practical scenarios, the number of times the protocol can be called might be limited due to time constraints or rate-limiting mechanisms. Therefore, it is assumed that the attacker is permitted to repeatedly engage in protocol invocations under a query budget .

Depending on different purposes, the attacker may be interested in different types of membership information, as we discussed in the introduction. Here we assume the adversary is interested in both the positive members and the negative members. To this end, the adversary adaptively designs his attack strategy to maximize membership information leakage.

**Attacking Strategy:** We consider two types of attack strategies:

Deterministic attack, where the adversary treats each person's membership as a deterministic value, and his attack is based on narrowing down group size iteratively and eventually landing on selected subsets which incurs a return with a result equal to 0 (all negative) or the size of the subset (all positive). The attack may re-identify only a small subset of the victim set under small , but the inferred memberships are accurate and deterministic.

Statistical attack, where the attacker takes each individual's membership as a binary random variable. His attack is based on updating the posterior belief on these random variables to either maximize (reasoning positive membership) or minimize (reasoning negative membership) them until the stopping criterion is matched. This type of attack strategy will make a guess of each membership in the dataset, and his strategy will guarantee his guessing accuracy above a tolerable threshold.

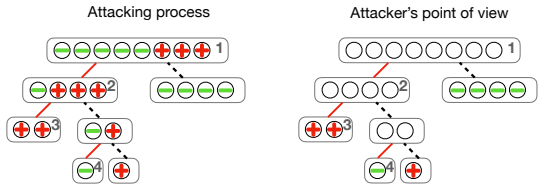


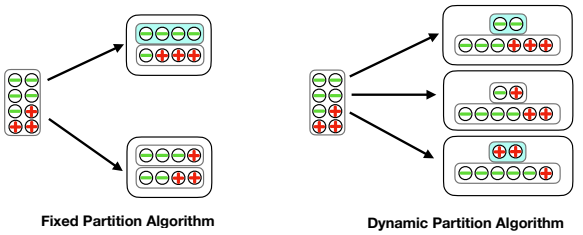

Fig. 1. An Illustration of deterministic hierarchical set membership inference attack. We use "+" to denote positive memberships, i.e., individuals in the intersection, and "-" to denote negative memberships, i.e., individuals in the attacker' set only.

### B. Preliminary attack algorithms

A brute-force attack involves the attacker randomly selecting an individual from $X$ set and checking their membership by invoking the protocol. The attacker then moves on to another individual in $X$. To determine the membership of $|X|$ individuals, the attacker must make $|X|$ protocol calls. This approach is inefficient for two reasons: first, it requires a large number of calls; and second, it can be easily countered by incorporating some protocol output policies. Such as Apple's PSI-AD protocol, which only returns an intersection size greater than a certain threshold.

The algorithm proposed in [14] introduces an attacker that uses a strategy similar to binary search. It constructs a binary tree with target elements at the root, and each non-leaf node's set is divided into two non-empty, disjoint subsets stored in its child nodes. Here node denotes a subset of $X$, and a child node is a subset of its parent node. The attacker then performs a depth-first search (DFS) on the tree, invoking the protocol with the victim to get the intersection size for each visited node's set. The search ends at a node where the stored set's size equals the received intersection size. Unvisited subtrees are queued for future DFS.

The attack in [14] is also optimized by adopting a greedy DFS approach. The attacker should prioritize searching subtrees with the highest 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 the subtree. The attacker can use this ratio as a priority score to sort the subtrees in a priority queue. By doing so, the attacker will first run DFS on the subtree with the highest merge probability, further improving the efficiency of the attack. An illustrative example of this algorithm is shown in Fig. 1.



Fixed Partition Algorithm

| Call # | Partition | Exp.Leak |
|---|---|---|
| 1 | 0 | 0 |
| 2 | 4 | 0.57 |

Dynamic Partition Algorithm

| Call # | Partition | Exp.Leak |
|---|---|---|
| 1 | 0 | 0 |
| 2 | 2 | 0.928 |

Fig. 2. A comparison of different tree partition principles in hierarchical attacks on PSI-CA protocols.

The attack in [14] is more efficient than the toy attack due to its binary-search-like strategy. By constructing a binary tree, the attacker can save at least half of the protocol invocations for every non-root layer. This is because after visiting a child node, the return of the other child node can be deduced from the parent node and the visited child node. The efficiency of the baseline attack is no worse than the toy attack, and it can even be better since it may terminate early when the current node contains all positive or negative members, reducing the number of invocations.

## IV. Deterministic attack

In this section, we propose attacks for the PSI protocols treating each individual's membership as deterministic values. We first illustrate with an example showing the limitations of the aforementioned works. Then we present our DyPathBlazer, a dynamic programming approach for PSI-CA protocol. At the end of this section, we show how auxiliary information in the output improves attack efficiency and propose TreeSumExplorerer, an efficient attack algorithm for PSI-SUM.

### A. Even v.s. uneven tree partition

One possible direction to optimize the hierarchical structural attack is to adopt an uneven tree partition principle rather than divide all nodes evenly. We provide more insights through the following example.

Suppose an attacker possesses a dataset containing eight individuals, three of whom also are positive members and the rest of them are negative members. The adversary can request a PSI call twice. Our exploration focuses on two distinct attack strategies: fixed partitioning and dynamic partitioning.

In the fixed partition strategy, the adversary consistently bisects the tree, employing one of the child nodes for the next round. As shown in the figure, the attacker employing the fixed partition strategy initially executes a PSI run, identifying three positive members in their input but failing to deduce their identities. Then, the attacker randomly selects four individuals from the $X$ for the second PSI run. As depicted in the figure, two potential scenarios arise: The first scenario, with a probability of $\binom{5}{4} = \binom{8}{4}$, allows the attacker to discern the identities of four individuals. The second scenario unfolds with a probability of $\binom{5}{3}\binom{3}{1} = \binom{8}{4}$, in which the attacker cannot infer any identities. Following two PSI calls, the expected number of inferred memberships is 0:57. In contrast, the dynamic partitioning strategy involves an algorithm where the attacker can unevenly split the current branch, which then serves as the input for the next PSI call. The rationale is that the attacker, understanding the constraints of their protocol run time, opts to thoroughly identify a subset of individuals instead of selecting a larger quantity and failing to determine all their memberships. Case 2 from Figure 2 illustrates the attack process using the dynamic partition strategy.

In this process, the attacker initiates a PSI run, identifies three individuals as positive members, and then randomly chooses two individuals for the second round. This can result in three potential outcomes: Both selected individuals are negative members. This scenario, with a probability of $\binom{5}{2} = \binom{8}{2}$, results in a leakage of 2. The two individuals include one positive and one negative member, yielding no leakage. Both selected individuals are negative. This outcome, with a probability of $\binom{3}{2} = \binom{8}{2}$, results in a leakage of 2. The expected leakage after two PSI calls is 0:928, which exceeds that of the fixed partition scenario.

This raises a pertinent question: Given that the dynamic partition algorithm can potentially enhance inference efficiency, how can the adversary determine the optimal partition factor (for example, 2 in the example above)? In the following section, we introduce DyPathBlazer, a dynamic programming solution designed to resolve this optimal partition factor issue.

### B. DyPathBlazer: A bottom-up dynamic programming solution

The proposed algorithm is depicted in Alg.1 and can be summarized as follows. The attacker requests a PSI call for $X$ and observes a return $C_X$, the PSI call budget is also updated to $-1$. These together determine an optimal partition factor $K$ (the determining function will be introduced later). He then randomly divides the tree into two child nodes. One with $K$ elements (denoted as left child), one with $|X| - K$ elements (denoted as right child). He submits a child with a shorter length to the PSI protocol, the left child for example, and observes an outcome $C_L$. Then $C_R = C_X - C_L$. The next step is to compare the expected leakage of the left child and the right child. The child node with a larger expected leakage is treated as the parent node and waiting for $K$ in the next iteration. The child node with smaller expected leakage is pushed into a priority queue. If all elements' membership in the current node is determined (all positive or all negative), the algorithm makes a prediction and dequeues a node with the highest priority as the current node. This process goes on until the protocol call budget is exhausted or all elements' membership in $X$ are determined.

The selection of $K$ is dependent on three parameters: the number of elements in the parent node $|N|$, the number of positive members of the elements $C_N$, and the protocol run budget $\_$. We define the tuple containing these three factors as a *state*: $(|N|, C_N, \_)$.

**Algorithm 1** DyPathBlazer

**Input:** A set X of target elements, pre-calculated Intermediate results , protocol invocation times

**Output:** Classified sets $Z_{pos}$, $Z_{neg}$.

Initialize $Z_{pos} = ?$, $Z_{neg} = ?$
$C_X$ PSI-CA(X; Y), = 1.
queue $f(jXj; C_X; X)g$
**while** queue is not empty **do**
  $(C_N; N)$ queue.dequeue
  **while** $0 < C_N < jNj$ and $> 0$ **do**
    $K =$ $(N; C_N; )$
    $N^L_K$ N[: $K$], $N^R_K$ N[$K$ :]
    $C_L$ PSI-CA($N^L_K$; Y), 1
    $C_R$ $C_N$ $C_L$
    **if** $^L_K$ $^R_K$ **then**
      $(C_N; N)$ $(C_L; N^L_K)$
      queue.enqueue $f(\ ^R_K; C_R; N^R_K)g$
    **else**
      $(C_N; N)$ $(C_R; N^R_K)$
      queue.enqueue $f(\ ^L_K; C_L; N^L_K)g$
    **end if**
  **end while**
  **if** $C_N = jNj$ **then** $Z_{pos}$ $Z_{pos}$ [ N
  **else if** $C_N = 0$ **then** $Z_{neg}$ $Z_{neg}$ [ N
  **end if**
**end while**
**return** $Z_{pos}$, $Z_{neg}$

**Algorithm 2** Memo generating function

**Output:** Memorized space and .

Initialize and as hash tables, where the key of is $(N; C_N; )$, and the key of is $(N; C_N)$
(1,1,0) = 1; (1,0,0)=1;
(1,1) = 0; (1,0) = 1;
**return** ,

**Algorithm 3** Optimal Tree partition algorithm based on dynamic programming

**Input:** Size of the victim set $N$, memberships in the victim set $m$, number of protocol innovation budget

**Output:** , the optimal set partition principle.

**return** $(N; C_N; )$ if exists.
Initiate $(N; C_N; ) = 0$, $(N; C_N; ) = 0$, $(N; C_N) = N$
**for** 1 $k$ $(N + 1)$==2 **do**
  Calculate $^L_k(N; C_N; )$ with (4)
  Calculate $^R_k(N; C_N; )$ with (5)
  $(N; C_N; 1)$ with (6)
**end for**
$(N; C_N; ) = \text{argmax}_k\ ^k(N; C_N; )$
**if** $(N; C_N; )$ N **then**
  $(N; C_N) = \min f\ (N; C_N);\ g$
**end if**
**return** $(N; C_N; )$

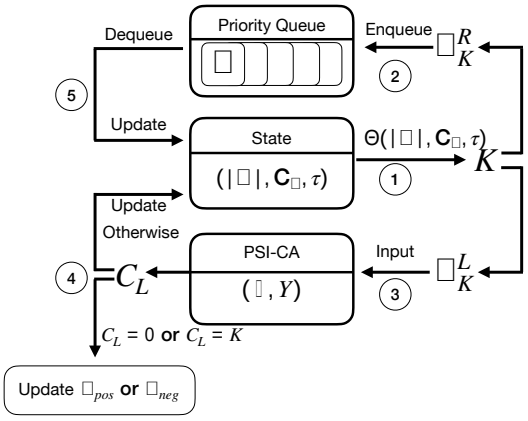



Fig. 3. DyPathBlazer model description

Note that different values of the partition factor $K$ may lead to different membership inference scenarios. Denote $^k(jNj; C_N; )$ as the expected membership leakage when the partition factor is $k$, given the current state. Denote $(jNj; C_N; ) = K$ as the partition rule, then:

$$K = \text{argmax}_k\ _k(jNj; C_N; ): \tag{1}$$

We remove the subscription to denote the best-case expected leakage: $(jNj; C_N; ) = _K(jNj; C_N; )$. In deriving the values of , we also need to introduce another term $(jNj; C_N)$, which denotes the smallest expected protocol run time the adversary needs to infer N target users' identity given $C_N$ of them are positive memberships:

$$(jNj; C_N) = \min_{(jNj; C_N; )\ N} : \tag{2}$$

After the partition, the current input set is divided into two parts with $K$ and $jNj$ $K$ elements respectively. Denote $N^L_K$ as the first set with $K$ elements, and $N^R_K$ as the set with the other $jNj$ $K$ elements. Then the positive members in the first $K$ elements, PSI-CA result of $N^L_K$ and Y can be denoted as a random variable $C_L$, with a probability distribution of:

$$\Pr(C_L = c) = \frac{\binom{C_N}{c}\binom{jNj\ C_N}{K\ c}}{\binom{N}{K}}: \tag{3}$$

We next show that $_K(N; C_N; )$ can be derived by recursive functions: After partition, the total leakage by choosing the left child becomes:

$$^L_K(N; C_N; ) = \begin{cases} E_{C_L}[\ (K; C_L; \quad 1)]; & \text{if} \quad (K; C_L) \quad 1; \\ K + E_{C_L}[\ (jNj \quad K; C_N \quad C_L; \quad (K; C_L))]; & \text{otherwise:} \end{cases} \tag{4}$$

Similarly, the total leakage by choosing the right child becomes:

$$\mathcal{L}_K^R(\mathsf{N}; C_\mathsf{N}; \ ) = \begin{cases} E_{C_L}[\ (j\mathsf{N}j \quad K; C_\mathsf{N} \quad C_L; \quad 1)]; \\ \qquad\qquad \text{if} \quad (j\mathsf{N}j \quad K; C_\mathsf{N} \quad C_L) \qquad 1; \\ j\mathsf{N}j \quad K + E_{C_L}[\ (K; C_L; \qquad (j\mathsf{N}j \quad K; C_\mathsf{N} \quad C_L))]; \\ \qquad\qquad \text{otherwise:} \end{cases} \tag{5}$$

Then,

$$\mathcal{L}_K(j\mathsf{N}j; C_\mathsf{N}; \ ) = \max \quad \mathcal{L}_K^L(j\mathsf{N}j; C_\mathsf{N}; \ ); \ \mathcal{L}_K^R(j\mathsf{N}j; C_\mathsf{N}; \ ) \ : \tag{6}$$

Intuitively, the expected membership leakage given of the current *state* depends on the partition factor $K$. For a given $k$, the algorithm chooses from the left or the right child according to their corresponding expected leakage values: $\mathcal{L}_K^L(j\mathsf{N}j; C_\mathsf{N}; \ )$ and $\mathcal{L}_K^R(j\mathsf{N}j; C_\mathsf{N}; \ )$. The algorithm subsequently selects the branch with the maximum expected leakage value as the new input of the PSI, relegating the other part to the priority queue. The processing framework is shown in Fig.3.

Now suppose the left branch has a larger expected leakage than the right. After dividing N into $\mathsf{N}_K^L$ and $\mathsf{N}_K^R$, the algorithm calls the PSI-CA protocol and receives an observation of the cardinality of the intersection $C_L$. If $C_L = 0$, none of the elements in the input $\mathsf{N}_K^L$ belongs to the intersection and $Z_{neg} = Z_{neg} \ [\ \mathsf{N}_K^L$. On the contrary, if $C_L = K$, all elements belong to the intersection and $Z_{pos} = Z_{pos} \ [\ \mathsf{N}_K^L$. Other than these two cases, the new $C_L$, together with $K$ and 1 form a new state of the attack $(K; C_L; \quad 1)$.

It is worth noting that once the expected membership leakage for the parent node is calculated, all the expected membership leakage of its child nodes are available, as they are the intermediate steps in calculating the parent node. i.e., to calculate $(\mathsf{N}; C_\mathsf{N}; \ )$, all $(\mathsf{N}^0; C_\mathsf{N}^0; \ ^0)$ are required, where $1 \quad \mathsf{N}^0 \quad \mathsf{N}$, $1 \quad C_\mathsf{N}^0 \quad C_\mathsf{N}$ and $0 < \ ^0$ . The same rule also applies to the structure of $(\mathsf{N}; C_\mathsf{N})$. When the algorithm is proceeding, any observations on the real number of positive memberships in the current node are included. Therefore, the policy space of only needs to be calculated once offline and can be directly used for further retrieval. This backtracking structure of building the memorization table of $(\mathsf{N}; C_\mathsf{N})$ in the dynamic programming solution is depicted in Fig. 4.

The following Theorem states the optimality of the proposed algorithm under the hierarchical attack structure.

**Theorem 1.** *The proposed algorithm is optimal in maximizing the number of expected membership inferences under the hierarchical structure.*

*Proof.* The proof of Theorem 1 follows a simple concept containing two steps, in the first step, we show that under a given dividing value, the state of the membership inference attack can be decomposed into an average (expectation) of all possible sub-branches. Then in the second step, we show our algorithm always selects the optimal dividing value according to the maximum number of expectations of the inference.

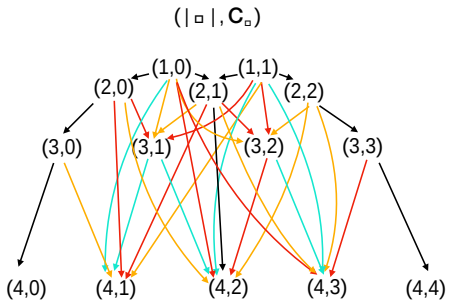



Fig. 4. The backtracking structure of memorization table in the dynamic programming solution

□

**Leakage Lowerbound** We next analyze the worst-case performance of our proposed attack algorithm. For a given dataset, after the initial PSI call, the state variables of are available. Denote them as $(j\mathsf{N}j; C_\mathsf{N}; \ )$. With $(\mathsf{N}; C_\mathsf{N}; \ )$, the partition factor $K$ can be determined. The uncertainty of the attack performance comes from the randomness in picking the $K$ out of the $j\mathsf{N}j$ individuals for the following protocol call. To derive the lower bound of the performance, here we analyze the worst-case scenario of selection. Then the left branch terminates if $K \quad C_L = 0$ or $C_L = 0$, and the right branch terminates if $C_\mathsf{N} \quad C_L = 0$ or $\mathsf{N} \quad K \quad C_\mathsf{N} + C_L = 0$. Note that if any of these conditions approach to 0, the corresponding branch tends to terminate. The worst-case value of $C_L$ satisfies the following condition:

$$\max_{C_L}[(K \quad C_L)^2 + C_L^2 + (C_\mathsf{N} \quad C_L)^2 + (j\mathsf{N}j \quad K \quad C_\mathsf{N} + C_L)^2]: \tag{7}$$

**Proposition 1.** *The worst-case sampled $C_L$ for the partition factor $K$ is $C_L = C_\mathsf{N} K = \mathsf{N}$.*

Then the lower bound can be numerically derived using the dynamic programming algorithm in DyPathBlazer with the following modification: at each node, instead of asking for a PSI run, the adversary calculates $C_L$ according to (7). Therefore, the online attack algorithm can be transferred to a local attack except for the initial protocol run.

**Remark 1.** *Compared with the leakage lower-bound of the attack proposed in [14], it is straightforward $K = j\mathsf{N}j=2$, and the worst case $C_L = C_\mathsf{N}=2$.*

We theoretically compare the lower bound of the membership leakage of the DyPathBlazer and the algorithm in [14] (denoted as USENIX 22) under two cases: case 1) among 100 individuals, 50 are positive members. case 2) among 100 individuals, 20 are postiive members. The comparison results are plotted in Fig. 5.

### C. TreeSumExplorer: Deterministic attack on PSI-SUM

In this part, we propose an attack algorithm to evaluate the privacy leakage of the PSI-SUM protocol. This protocol

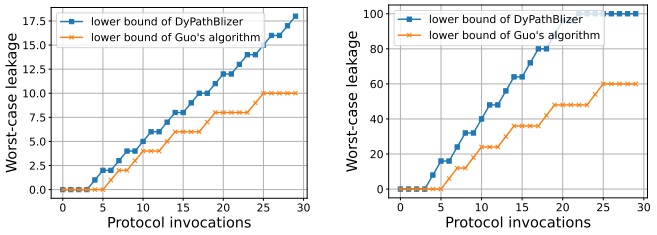



Fig. 5. Performance lower bound comparison (worst-case leakage comparison) of DyPathBlazer and Guo's algorithm in [14]. Both cases consider a dataset of 100 individuals. Case 1) assumes 50 positive members, case 2) assumes 10 positive members. Higher lower bounds from DyPathBlazer guarantees better efficiency in the worst-case scenario. DyPathBlazer's lower bound surpasses that of Guo's algorithm lower bound, ensuring improved efficiency in the worst-case scenario.

is a black box that receives a set $X$ from one party and a table $Y = (y_i; v_i)$ of index-value pairs from the other party. It internally aggregates the values associated with the indices in the intersection $X \setminus Y$ and gets the sum $\sum_{y_i \in X \setminus Y} v_i$. Then, the protocol sends this sum and the intersection size $|X \setminus Y|$ to the party that inputs the table.

Intuitively, the leakage of the PSI-SUM should be greater than the leakage of PSI-CA, as more information (summation of the intersection) is observable during each protocol run. We next summarize the steps to take advantage of this information and conduct a more efficient privacy evaluation.

The basic idea behind this approach is to offline search all possible combinations that match the summation result. For example, after a protocol is called, the adversary observes the following: input size $|N|$, returned number of members $C_N$, and the summation of the intersected members' value SUM. Instead of directly applying the cardinality-based attack, the adversary is capable of traversal all possible combinations of the $C_N$ out of $|N|$ users in the dataset, and checking if the summation of their data values matches SUM. So the membership identity of these $|N|$ users can be inferred in one shot (best case depending on the number of possible combinations). Such a problem is defined as an N-SUM problem (summation of $C_N$ elements in the array, denoted as Arr, matches a SUM) [26], and a common algorithm for solving the N-SUM problem is summarized as follows:

Sort the array: The Arr is sorted in ascending order.

Check the Base Case: If $C_N = 2$, the problem is reduced to a 2-SUM problem. This is solved by initializing two pointers at both ends of the array and moving them toward each other until they meet. If the sum of the numbers at the pointer indices equals the target, then record this pair. The pointers are adjusted based on whether the current sum is less than or greater than the target.

Handle the Recursive Case: If $C_N > 2$, treat the problem as an $(C_N - 1)$-SUM problem for each element in the array. This involves iterating over the array, and for each element, recursively solving an $(C_N - 1)$-SUM problem with a new target that is equal to the original target minus the current element.

The computation complexity of the N-SUM solution described above is $O(|N|^{C_N - 1})$, which depends on the length of the input and the number of positive membership individuals. There is also a line of work improving this complexity [5], [18], [23], however, is out of the scope of this paper.

There could be multiple combinations of users whose summation matches SUM returned. The possibilities form a candidate set for the input of the next iteration. The adversary, then, randomly picks one of these combinations $N'$ as the input to call for another protocol run. If the returned $SUM'$ matches the previous SUM, the selected combination are all positive memberships. Further, all memberships in N are determined. Otherwise, the adversary has selected an incorrect combination as input. The adversary observes and obtains two sub N-SUM problems: a. N-SUM($N'$, SUM', $C_N$) and b. N-SUM($N \cap N'$, SUM-SUM', $C_N - C_{N'}$). For each subproblem, the priority the sub-problem a or b is calculated as follows:

$$\text{Priority} = \frac{|N|}{|\text{Arr}|}; \tag{8}$$

The adversary selects $\max\{\text{Priority}_a; \text{Priority}_b\}$ and continues his inference with the branch with higher priority, then pushes the lower priority branch to the priority queue. The algorithm for TreeSumExplorer is summarized in the Alg. 4.

**Algorithm 4** Privacy evaluation algorithm for PSI-SUM

**Input:** A set $X$ of target elements, protocol invocation times , maximum set size $N$.

**Output:** Predicted sets: $Z_{pos}; Z_{neg}$.

```
Initialize Z_pos = ?, Z_neg = ?
Find a subset N according to the computation power
while 0 <  do
    (C_N; SUM)  PSI-SUM(N; Y),  1
    Arr = N-SUM(N; SUM; C_N)
    Priority  |N|/|Arr|.
    queue.enqueue {Priority; C_N; SUM; N; Arr}
    while queue is not empty do
        (C_N; SUM; N; Arr)  queue.dequeue()
        while 1 < |Arr|, and 0 <  do
            N_L = Arr[0], N_R = N ∩ N_L
            (C_L; SUM_L)  PSI-SUM(N_L; Y),  1
            C_R  C_N  C_L, SUM_R  SUM  SUM_L
            Arr_L = N-SUM(N_L; SUM_L; C_L)
            Arr_R = N-SUM(N_R; SUM_R; C_R)
            Compute priorities according to (8)
            Continue with the high-priority branch
            Push low-priority branch to the queue
        end while
        if |Arr| = 1 then
            Z_pos  Z_pos [ Arr, Z_neg  Z_neg [ N=Arr
        end if
    end while
end while
return Z_pos; Z_neg
```

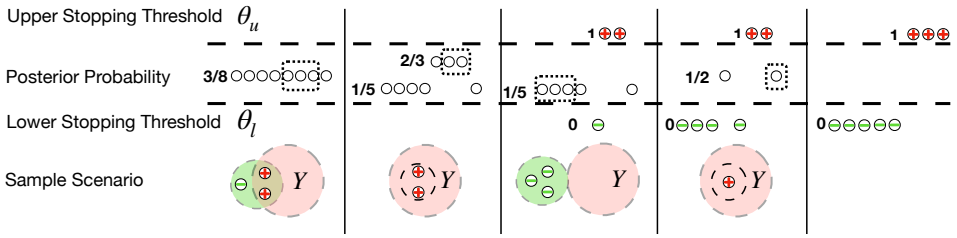



Fig. 6. An illustration of statistical membership inference attack.

## V. ActBaysian: Statistical Attack on PSI-CA

Different from the deterministic attack, where the adversary takes each individual's membership as constant. In the context of a statistical attack, each individual's membership is regarded as a binary random variable. This allows the adversary to design a soft-stopping criterion to finish his guessing. Therefore, the attacker's goal is to select subsets of $X$ which helps accurately infer the identity of individual members. Denote $L_i$ as a binary random variable that $L_i = 1$ if $x_i \in X \setminus Y$. ($L_i = 0$ otherwise). Then the memberships in $X$ can be specified as a random vector $p$, where $p[i]$ denotes the probability that $\Pr(L_i = 1)$. Suppose for PSI attack runtime $t \in [1; \ldots; \ ]$, the attacker selects a subset $S_t$ of $X$ as the input of PSI, let $O_1; \ldots; O$ denote the release from the PSI protocol, then $O_t = |S_t \setminus Y|$.

In this section, we propose a statistical attack algorithm based on Bayesian posterior update and adaptive learning. We call this algorithm ActBaysian algorithm. The active learning process guarantees the adversary selecting the most informative subset of $S_t$ at each time $t$. The posterior update process enables the adversary to update his belief on $L_i$ based on the observations.

*1) Posterior belief update:* After each PSI-CA call, the adversary observes the cardinality of the positive members contained in the input dataset. Based on this observation, the adversary updates their posterior belief. The update rule follows a maximum likelihood criterion. For a given pair of PSI input/output, denoted as $S_t$ and $O_t$ respectively, the updated prior for each individual in the subset becomes:

$$p[i] = \frac{O_t}{|S_t|}. \tag{9}$$

This update reflects the ratio of positive members observed ($O_t$) to the cardinality of the input dataset ($|S_t|$). The higher the observed positive count relative to the size of the input subset, the higher the posterior belief assigned to each individual.

On the other hand, all sets containing $S_t$ can also update their posterior beliefs. Let $S_t$ be the subset such that $S_t \setminus S_t = ?$, and the result of PSI-CA($S_t \cap S_t$) is known, denoted as $O_{tot}$. After observing the output $O_t$, the posterior belief of all individuals' membership in $S_t$ becomes:

$$p[i] = \frac{O_{tot} \quad O_t}{|S_t|}. \tag{10}$$

This update is based on the remaining positive count ($O_{tot} \quad O_t$) after subtracting the observed positive count $O_t$ from the known total positive count $O_{tot}$. The posterior belief is calculated by dividing this remaining positive count by the cardinality of $S_t$. By updating the prior and posterior beliefs based on the observed positive counts and known total positive counts, the adversary can refine their belief about the membership status of each individual in the respective subsets. This iterative belief update process enables the adversary to incorporate new information and adjust their inference based on the observed results.

**Stopping criterion:** In the context of a statistical attack, the adversary has the flexibility to enhance their inference power at the cost of sacrificing some inference accuracy. This is achieved through the use of a soft-stopping criterion, which involves setting an upper threshold $\quad_u$ and a lower threshold $\quad_l$. The adversary classifies an individual's membership as positive if their posterior probability $p[i]$ is greater than or equal to $\quad_u$, and as negative if $p[i]$ is less than or equal to $\quad_l$. It is important to note that the selection of these thresholds plays a crucial role in determining the performance of the attack. The threshold values significantly influence the trade-off between inference accuracy and the number of individuals' memberships inferred: If the upper threshold $\quad_u$ is set close to 1 and the lower threshold $\quad_l$ is set close to 0, the inference attack achieves high accuracy. However, there may be a limited number of individual memberships inferred. This occurs because the adversary requires a higher level of certainty before classifying an individual as a positive or negative member. Consequently, only individuals with posterior probabilities close to the extremes will be confidently classified, potentially leading to a reduced number of identified individuals. On the other hand, if both the upper threshold $\quad_u$ and the lower threshold $\quad_l$ are set close to 0.5, a larger proportion of individuals' memberships can be inferred. However, the inference accuracy may be compromised. With lower thresholds, even samples with relatively uncertain posterior probabilities will be classified, resulting in a higher number of individuals being identified. Nevertheless, the trade-off is that the accuracy of these classifications may be lower compared to using more stringent thresholds.

*2) Active learning for input set selection:* The active learning contains two phases: 1. constructing candidate input sets by comparing the absolute distance between each posterior

and the threshold. 2. determine the input set based on the minimized Manhattan distance.

**Minimize absolute distance:** During the attack process, the adversary aims to select individuals who are likely to meet the stopping criterion as the input for the PSI protocol. This selection is done in two directions: individuals whose prior probabilities are close to the upper threshold $_u$ and individuals whose priors are close to the lower threshold $_l$. The absolute distance between each individual's posterior and the upper/lower threshold is defined as follows:

$$\begin{aligned} d_i^u &= jp[i] \quad {}_u j \\ d_i^l &= jp[i] \quad {}_l j: \end{aligned} \tag{11}$$

For a given iteration, the minimized distance in the posterior vector is defined as $d^u_{\min} = \min_i d_i^u$, and $d^l_{\min} = \min_i d_i^l$, respectively. However, directly selecting individuals with the highest or lowest priors introduces no randomness to the output, potentially leading to a dead loop in the attack. To mitigate this issue, two factors are introduced: the sampling rate $r$ and the tolerance factor $tol$.

The sampling rate and tolerance factor together determine the grouping principles for selecting the input individuals. The candidate input for the next iterations includes:

$$\begin{aligned} S^u &= \underset{d_i^u[i] \quad d^u_{\min} \quad tol}{[} x_i \quad \mathbf{1}_{sample} \\ S^l &= \underset{d_i^l[i] \quad d^l_{\min} \quad tol}{[} x_i \quad \mathbf{1}_{sample}; \end{aligned} \tag{12}$$

where $\mathbf{1}_{sample}$ is an indicator function that is determined by the sampling result. Intuitively, a larger tolerance factor $tol$ results in grouping more individuals who are not prone to meet the stopping criterion. This allows for a broader exploration of individuals in the search for those whose posterior probabilities may cross the thresholds. However, a larger candidate size increases the number of individuals whose memberships can be inferred in one iteration. On the other hand, a small sampling rate $r$ tends to narrow down the inference scope quickly. The attack can rapidly infer a smaller portion of individuals with positive or negative memberships. However, this approach may lead to decreased overall inference efficiency since the algorithm becomes prone to depth-first search (DFS) behavior. The choice of the sampling rate and tolerance factor depends on the specific attack objectives and constraints. By carefully tuning these parameters, the adversary can balance the trade-off between the efficiency of the attack, the number of inferences made in each iteration, and the overall accuracy of the inference process. We present a detailed analysis in numerical evaluation.

**Minimize Manhattan distance:** As the adversary keeps calling the PSI protocol, each individual's posteriors are pushing either towards $_u$ or $_l$. Given a tolerant factor $tol$, before each PSI call, the adversary has the choice to select from $S_u$ and $S_l$ as his input for the next PSI-CA call. This can be achieved by comparing the minimized averaged Manhattan distance between $S_u$ and $_u$, $S_l$ and $_l$:

$$\begin{aligned} D_u &= \sum_{i:x_i \in S_u} d_i^u / |S_u| \\ D_l &= \sum_{i:x_i \in S_l} d_i^l / |S_l|: \end{aligned} \tag{13}$$

Then, the adversary selects the input with a smaller distance toward the threshold. The algorithm is summarized in Alg. 5.

**Algorithm 5** Statistical Membership Inference Attack

**Input:** Victim set X, $_u$ upper threshold, $_l$ lower threshold, PSI call budget , sampling rate $r$, tolerance factor $tol$.
**Output:** Prediction of $Z_{pos}$ and $Z_{neg}$

```
Initialize prior distribution for each user fp[i] = 0.5g_{i=1}^N
while > 0 and |Z_pos + Z_neg| < |N| do
    S_u ← S_{ fd_i^u[i] ≤ d^u_min + tol g x_i, sampling with r
    S_l ← S_{ fd_i^l[i] ≤ d^l_min + tol g x_i, sampling with r
    Calculate D_u, D_l with (13).
    if D_u ≤ D_l then
        S ← S_u
    else
        S ← S_l
    end if
    C_S = PSI-CA (S; Y)
    Update posteriors according to (9) and (10)
    for 1 ≤ i ≤ |N| : do
        Z_pos = Z_pos [ x_i if p[i] < _l
        Z_neg = Z_neg [ x_i if p[i] > _u
    end for
end while
return Z_pos, Z_neg
```

### A. Error bound

The statistical attack incurs classification error corresponding to the upper and the lower stopping threshold: A group of individuals is classified as positive members if most of them are positive members. Similarly, a group of individuals are classified as negative members if most of them are negative members. Specifically, we define Type I, Type II, and misclassification rate for an individual $x_i$ as follows:

$$\begin{aligned} P^e_{typeI} &= \Pr(x_i \in X \setminus Y | x_i \in Z_{neg}); \\ P^e_{typeII} &= \Pr(x_i \notin X \setminus Y | x_i \in Z_{pos}); \\ P^e_{mis} &= P^e_{typeI} \Pr(x_i \in Z_{neg}) + P^e_{typeII} \Pr(x_i \in Z_{pos}): \end{aligned} \tag{14}$$

When the stopping state containing $x_i$ is $(N; C_N; )$, from algorithm 5, the upper bound of these error probabilities are summarized in proposition 2.

**Proposition 2.** *The probability upper bounds of Type I, Type II error, and the misclassification rate are as follows:*

$$\begin{aligned} P^e_{typeI} &\le 1 - C_N/|N| \text{ if } {}_u - C_N/|N| \le 1; \\ P^e_{typeII} &\le C_N/|N| \text{ if } 0 - C_N/|N| \le {}_l; \\ P^e_{mis} &\le 1 - {}_u + {}_l: \end{aligned} \tag{15}$$

It is straightforward to extend the error probabilities to correct guessing probabilities:

$$P_{TP} = 1 \quad P^{e}_{typeI} \quad C_N=jNj \text{ if } _u \quad m=N \quad 1;$$
$$P_{FP} = 1 \quad P^{e}_{typeII} \quad 1 \quad C_N=jNj \text{ if } 0 \quad C_N=jNj \quad _l: \tag{16}$$

where $P_{TP}$ stands for true positive probability and $P_{FP}$ stands for false positive probability.

## VI. Experiments

In this section, we evaluate real data to compare different attack algorithms. We divide the evaluation into two subsections, for the first part, we compare the efficiency of different deterministic attacks, then in the second part, we show the performance of the statistical attack.

**Attacks considered** For the deterministic attacks, we perform several membership inference attacks on PSI-CA alike protocols. Specifically, we compare the following attack algorithms: (a) baseline attack in [14] which is denoted as "Guo's algorithm-CA"(b) DyPathBlazer, as described in Section IV-B. (c) TreeSumExplorer in Section IV-C, and (d) The USENIX22 solution but takes SUM as the output of the PSI denoted as Guo's algorithm-SUM.

For the statistical attacks, we show the impact of different parameters on the attack efficiency and accuracy. Then we present attack performance against Differentially Private PSI protocols.

**Datasets:** We consider two real-world membership-sensitive datasets for evaluation. The first dataset in our experiment is Taobao's dataset of ad display/click records [28]. 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. This dataset is used for leakage quantification to measure ad conversion revenue. The second dataset is COVID-19 dataset of tested individuals in Israel4 [1]. This dataset was collected by Israel Ministry of Health. The COVID-19 dataset includes 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. This dataset is used for the leakage quantification in COVID-19 contact tracing.

### A. Deterministic attacks

**Offline attack evaluations** We first consider an offline setting using the Taobao dataset. We select a brand with id number 185, which has 504 ads (for another set of experiments, we select brand id 279 with 403 ads). Each ad is associated with a unique price for the related product and a list of buyers who may or may not have clicked the ad. We let party A, the company of the brand, include the list of buyers labeled by the total amount of money they have spent on this company's product. For party B, the ads platform, we assume they possess the list of individuals who have clicked the ads from this company. Party A requests a PSI-SUM protocol and wants to know how many people have clicked the ads they put on Party B's platform before buying their products and how much they spend on their products. On the other hand, Party A is also interested in re-identifying the persons in the intersection for targeted advertising. After preprocessing, there are 22615 persons who have purchased the products from brand 185. Among them, 4762 have clicked the ads from this brand. For brand 279, there are 15424 persons who have purchased the products, and 332 users have clicked the ads.

The offline attack evaluations are shown in Fig. 7, where the membership leakage percentages using different attacks are compared according to different protocol call limitations. Further, we consider two subcases when the adversary is only interested in the identity of individuals in the intersection or those who are not in the intersection, respectively. It is worth noting that our DyPathBlazer and TreeSumExplorer algorithms can both be adapted to each setting by designing different goals in the objective functions. Observe that our DyPathBlzaer outperforms the attack in Guo et al. in efficiency, while the PSI-SUM attack achieves the highest efficiency.

**Online attack evaluations:** In the second scenario, we consider an online setting using the COVID-19 dataset. Party A is a local community that provides COVID testing services. Party B is the Lab that processes the testing results. It is assumed that Party B directly publishes testing results to each individual, so Party A is unclear which individual tests positive for COVID. On the other hand, Party A keeps monitoring the trend of total positive individuals in the community and is also interested in inferring the identity of each positive individual for targeted control. Party A calls for a PSI-CA protocol with Party B daily. We evaluate different attacks for PSI-CA protocol by comparing the total number of positive individuals they infer according to the testing timeline.

The online attack evaluations are shown in Fig.8. Note that we slightly modified the DyPathBlazer to estimate and maximize only the expected number of positive members. The dataset is assumed to be updated daily but stays the same for one day. Party A's maximal protocol call limit is 10 times per day. Observe that our DyPathBlazer achieves significantly higher efficiency compared to the Guo et al.

### B. Statistical attacks

**For PSI-CA without protection** To illustrate the effectiveness of statistical attacks on PSI-CA, we initially adopt a similar context as with the deterministic attacks. However, we employ the attack methodology delineated in Section V. This algorithm is parameterized by $_u$, $_l$, $tol$, $r$, and . We assign default values to each of these parameters as follows: $_u = 0.9$, $_l = 0.1$, $tol = 0.1$, $r = 0.5$, $= 20$. Subsequently, while maintaining other values constant, we vary each parameter in turn and evaluate the resulting performance of the attack algorithm. The performance metrics under consideration include the number of correctly inferred individuals and the Type I and Type II error rates. The result is shown in Table I.

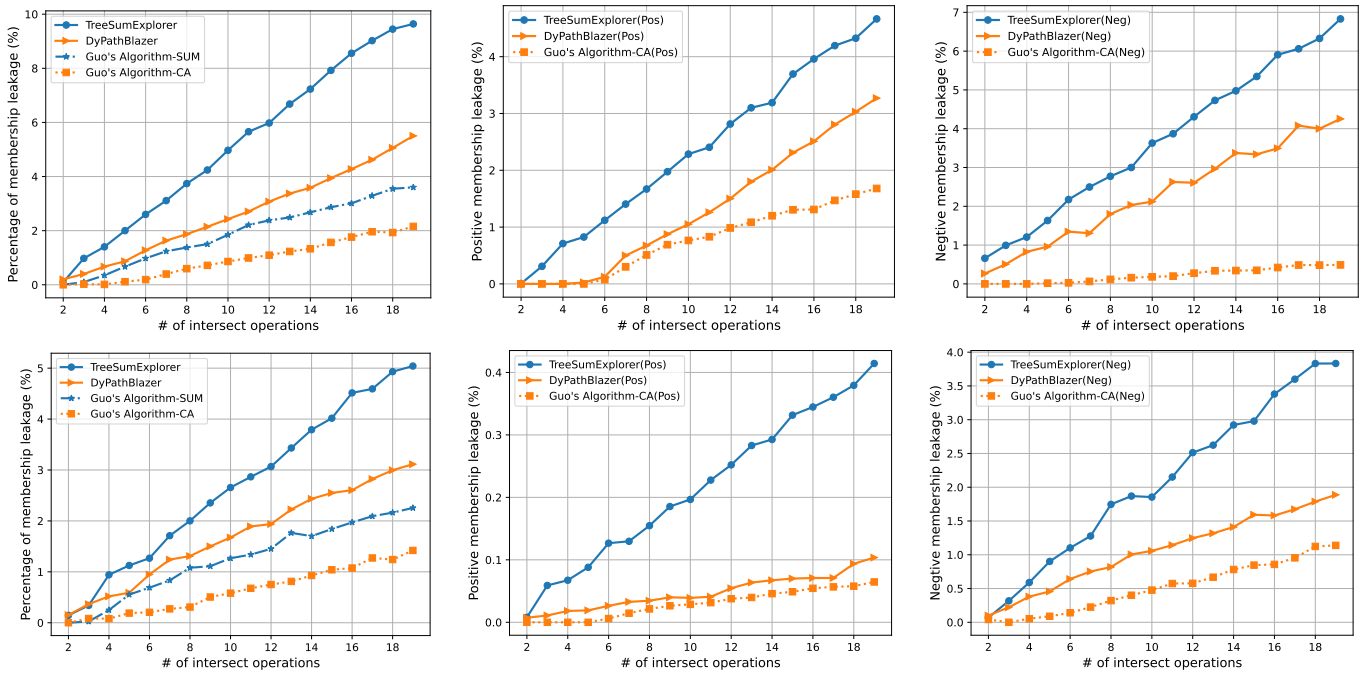



Fig. 7. Membership information leakages in PSI-SUM under different attacks. Column 1 displays the overall membership leakage, while Column 2 represents positive membership leakage, and Column 3 represents negative membership leakage. The first row corresponds to cases where the product company targets the advertising company, and the second row corresponds to cases where the advertising company targets the product company.

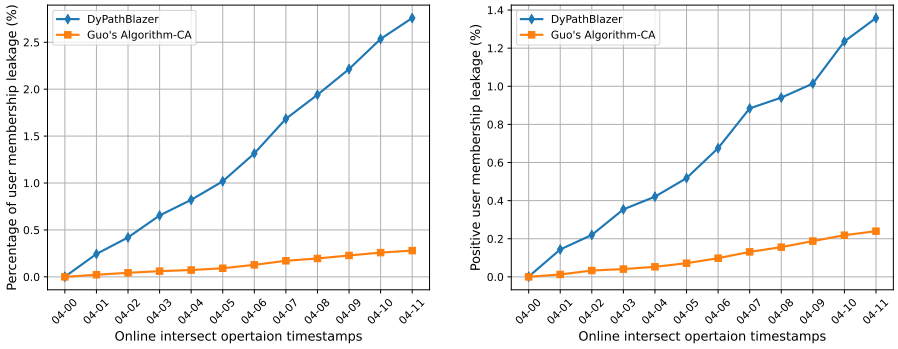



Fig. 8. Membership information leakages in PSI-CA under different attacks. Subcase (a) showing total membership leakage (positive member + negative member); (b) showing positive membership leakage only

Key insights can be derived from the results. As $_u$ increases, fewer individuals are classified as positive members, which in turn leads to a decrease in the true positive percentage as well as the Type I error rate. In contrast, reducing $_u$ allows a larger number of individuals to be classified as positive. While this results in an increase in true positive individuals correctly inferred, it also leads to a rise in the Type I error rate. The impact of $_l$ mirrors this pattern. The tolerance factor $tol$ regulates the quantity of individuals whose priors are updated in one PSI call. Lowering this value prompts the algorithm to focus on exploitation, resulting in insufficient individuals being inferred given a particular PSI call budget. Conversely, excessively increasing this value tilts the system towards exploration, causing a gradual narrowing of its scope and a consequent reduction in the final count of inferred individuals. The sampling rate $r$ determines the proportion of individuals to be included in the subsequent PSI input. An even division is typically advantageous unless is small, in which case a lower $r$ accelerates the narrowing down rate, analogous to our DyPathBlazer. Finally, for larger values of , the attack is capable of inferring most of the individuals' membership, and the error rate concurrently decreases.

**For PSI-CA with Differential Privacy** Recent studies have revealed the potential to enhance the privacy guarantee of PSI through its integration with DP. DP offers robust privacy protection by adding calibrated random noise to the raw response to a query, as defined in [8]. In the PSI context, a DP-incorporated PSI-CA protocol can be seen as introducing randomness to the published intersection size, and this randomness has been proven to achieve -DP.

TABLE I
STATISTICAL ATTACK: EFFECTS OF VARIOUS PARAMETERS ON RESULTS

| | Default | $_u$(0.8 / 1) | $_l$(0=0:2) | tol (0 / 0.2) | $r$ (0.3 / 0.9) | (10 / 50) |
|---|---|---|---|---|---|---|
| True Positive Percentage | 0.08 | 0.14 / 0.06 | 0.11 / 0.02 | 0.05 / 0.07 | 0.05 / 0.04 | 0.02 / 0.17 |
| True Negative Percentage | 0.27 | 0.25 / 0.29 | 0.22 / 0.31 | 0.23 / 0.25 | 0.21 / 0.22 | 0.12 / 0.83 |
| Type I error rate | 0.083 | 0.15 / 0 | 0.09 / 0.04 | 0.072/ 0.084 | 0.084 / 0.08 | 0.12 /0.064 |
| Type II error rate | 0.085 | 0.087 / 0.092 | 0 / 0.17 | 0.082 / 0.085 | 0.085 / 0.082 | 0.16 / 0.055 |

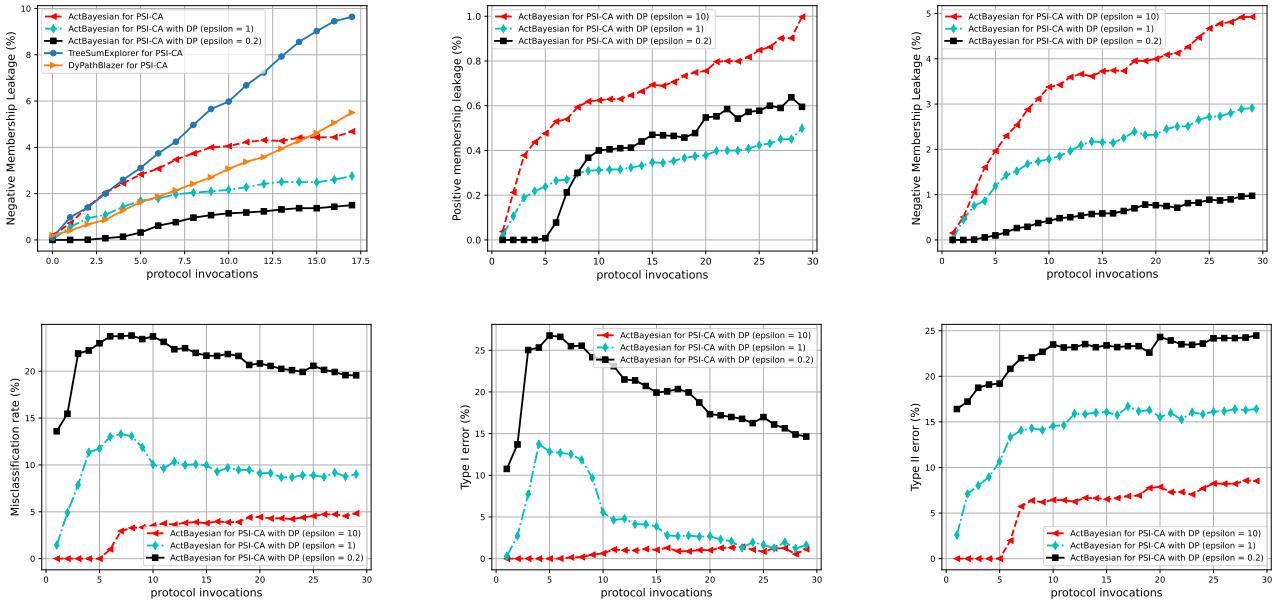



Fig. 9. Membership information leakages with statistical attack, ads company

Subsequent experiments aim to evaluate the membership information leakage from an -DP-protected PSI protocol. We are considering a Laplacian DP mechanism, wherein the scale of the Laplacian noise is defined as:

$$= \frac{f}{i} = -: \tag{17}$$

In this study, we derive Equation (17) under a basic DP composition theorem: the composition of consecutive privacy-preserving mechanisms, each satisfying -DP, complies with -DP. We set the sensitivity of DP, $f$ (as depicted in Equation (17)), to 1 by default, analogous to a counting query. It is crucial to note that deterministic attacks fall short in measuring privacy leakage under DP protection, primarily due to their exclusion of randomness and error considerations.

We incorporate Laplacian random noise into each PSI result in our ensuing experiments using the TaoBao dataset. We emulate settings akin to deterministic attacks but consider different values (1, 5, and 10) with = 30. Our evaluations of statistical attacks under DP-protected PSI protocol are illustrated in Fig.9. Fig. 9 features the Taobao dataset, processed identically to Fig.7. We include prior results from DyPathBlazer and TreeSumExplorer for comparative evaluation, examining positive and negative membership leakage, misclassification rate, and Type I and Type II errors across different subcases. A ROC curve comparison for different types of memberships is shown in Fig.10.

Key insights from the figures are summarized as follows: Without DP protection, TreeSumExplorer is generally more efficient than other attack algorithms, as the fewer combination possibilities in matching the returned SUM significantly narrow the targeted individuals' membership.

The statistical attack outperforms DyPathBlazer for small . For these lower values, the statistical attack algorithm infers more individual memberships, as its looser stopping criterion surpasses DyPathBlazer in efficiency at the cost of some accuracy. However, as increases, dynamic programming algorithms reveal their superiority in optimally dividing the input set, thus maximizing returns.

With DP protection, attack efficiency is diminished, with the reduction corresponding to the DP mechanism's strength ( ). DP-induced randomness makes updated posteriors less accurate, thereby decreasing inference accuracy. The error rate in DP-protected mechanisms rises with increasing . The misclassification rate first rises and then falls as increases. Initially, fewer inferred memberships result in a lower error

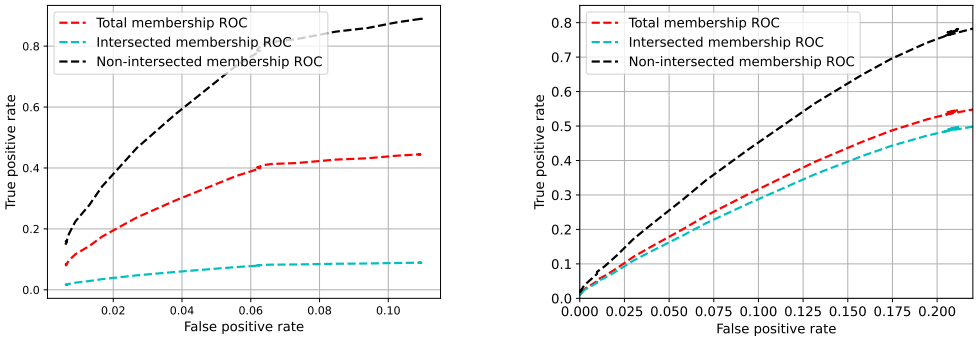



Fig. 10. ROC curves of the ActBaysian algorithm with DP protection, = 0:2

rate. As the number of inferred members grows, the error rate consequently increases. Eventually, as increases, the randomness introduced by DP is mitigated (due to consecutive queries weakening DP), rendering subsequent inferences more accurate.

## VII. Conclusion & Future Works

Private Set Intersection (PSI) protocols that reveal the size of the intersection may unintentionally disclose membership information regarding each parties' sets. While this doesn't directly breach the intended security assurance of PSI, which is to maintain the confidentiality of each party's input set, such PSI protocols can divulge extra details about whether members of one set are part of the other set or not.

In this study, we have delved into the realm of anonymity assessment frameworks specifically designed for intersection-size revealing PSI protocols. Our exploration has led to the development of two innovative strategies for deducing individual memberships within the intersecting set. These strategies include a deterministic attack algorithm supported by dynamic programming, which offers a theoretical performance guarantee and is further enhanced through the incorporation of side information. Additionally, we propose a statistical attack method based on Bayesian principles derived from active learning, which can augment information leakage with minimal compromise to accuracy. We also demonstrate, through real-world data, that our proposed methodology exhibits superior performance when compared to the most relevant prior research.

Given the de-anonymization concerns associated with PSI protocols discussed earlier, there is a pressing need for an innovative privacy-enhanced PSI protocol. This protocol should aim to minimize privacy leakage in situations where two parties must compute intersection-related statistics from their confidential datasets.

More importantly, when considering real-world applications, the demand for multi-ID PSI is frequently encountered. In this context, revealing the intersection size for each ID match can potentially expose significantly more information compared to the single-ID scenario. Effectively mitigating membership leakage in such cases presents a more formidable challenge and remains an open problem, especially when factoring in practical constraints related to communication and computation overhead in the implementation of such a PSI system.

## References


[1] Machine learning-based prediction of covid-19 diagnosis based on symptoms, url: https://github.com/nshomron/covidpred.

[2] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS '16, page 308–318, New York, NY, USA, 2016. Association for Computing Machinery.

[3] 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, SIGMOD '03, page 86–97, New York, NY, USA, 2003. Association for Computing Machinery.

[4] Prasad Buddhavarapu, Andrew Knox, Payman Mohassel, Shubho Sengupta, Erik Taubeneck, and Vlad Vlaskin. Private matching for compute. Cryptology ePrint Archive, Paper 2020/599, 2020. url-https://eprint.iacr.org/2020/599.

[5] Xi Chen, Yaonan Jin, Tim Randolph, and Rocco A. Servedio. Subset sum in time $2^{n=2}$=poly($n$), 2023.

[6] Emiliano De Cristofaro, Paolo Gasti, and Gene Tsudik. Fast and private computation of cardinality of set intersection and union. In Josef Pieprzyk, Ahmad-Reza Sadeghi, and Mark Manulis, editors, Cryptology and Network Security, pages 218–231, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[7] Thai Duong, Duong Hieu Phan, and Ni Trieu. Catalic: Delegated psi cardinality with applications to contact tracing. In Advances in Cryptology – ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part III, page 870–899, Berlin, Heidelberg, 2020. Springer-Verlag.

[8] Cynthia Dwork. Differential privacy. In Encyclopedia of Cryptography and Security, 2006.

[9] Cynthia Dwork. Differential privacy: A survey of results. In Manindra Agrawal, Dingzhu Du, and Zhenhua Duan, editors, Theory and Applications of Models of Computation: 5th International Conference, TAMC, pages 1–19. 2008.

[10] Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Calibrating noise to sensitivity in private data analysis. In Theory of Cryptography: Third Theory of Cryptography Conference, pages 265–284. 2006.

[11] 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, CCS '17, page 2295–2312, New York, NY, USA, 2017. Association for Computing Machinery.

[12] Michael J. Freedman, Carmit Hazay, Kobbi Nissim, and Benny Pinkas. Efficient set intersection with simulation-based security. J. Cryptology, 29:115–155, 2016.

[13] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan L. Camenisch, editors, Advances in Cryptology - EUROCRYPT 2004, pages 1–19, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[14] Xiaojie Guo, Ye Han, Zheli Liu, Ding Wang, Yan Jia, and Jin Li. Birds of a feather flock together: How set bias helps to deanonymize you via revealed intersection sizes. In 31st USENIX Security Symposium (USENIX Security 22), pages 1487–1504, Boston, MA, August 2022. USENIX Association.

[15] Yangsibo Huang, Zhao Song, K. Li, and Sanjeev Arora. Instahide: Instance-hiding schemes for private distributed learning. In International Conference on Machine Learning, 2020.

[16] 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, 2020.

[17] Bo Jiang, Mohamed Seif, Ravi Tandon, and Ming Li. Context-aware local information privacy. IEEE Transactions on Information Forensics and Security, 16:3694–3708, 2021.

[18] J. C. Lagarias and A. M. Odlyzko. Solving low density subset sum problems. In 24th Annual Symposium on Foundations of Computer Science (sfcs 1983), pages 1–10, 1983.

[19] Latanyasweeney. k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 10, 05 2012.

[20] Peihan Miao, Sarvar Patel, Mariana Raykova, Karn Seth, and Moti Yung. Two-sided malicious security for private intersection-sum with cardinality, 2020. urlhttps://eprint.iacr.org/2020/385.

[21] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In 2008 IEEE Symposium on Security and Privacy (sp 2008), pages 111–125, 2008.

[22] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In 2019 IEEE Symposium on Security and Privacy (SP), pages 739–753, 2019.

[23] Bartosz Przydatek. A fast approximation algorithm for the subset-sum problem. International Transactions in Operational Research, 9, 05 2000.

[24] Ryan M. Rogers, Adrian Rivera Cardoso, Koray Mancuhan, Akash Kaura, Nikhil T. Gahlawat, Neha Jain, Paul Ko, and Parvez Ahammad. A members first approach to enabling linkedin's labor market insights at scale. ArXiv, abs/2010.13981, 2020.

[25] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In 2017 IEEE Symposium on Security and Privacy (SP), pages 3–18, 2017.

[26] Nei Yoshihiro Soma and Paolo Toth. An exact algorithm for the subset sum problem. European Journal of Operational Research, 136(1):57–66, 2002.

[27] Latanya Sweeney. Simple demographics often identify people uniquely. 2000.

[28] Tianchi. Taobao display advertisement click-through rate prediction dataset, url: https://tianchi.aliyun.com/dataset/datadetail?dataid=56, 2018.

[29] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In 2018 IEEE 31st Computer Security Foundations Symposium (CSF), pages 268–282, 2018.

[30] Yongjun Zhao and Sherman S.M. Chow. Can you find the one for me? In Proceedings of the 2018 Workshop on Privacy in the Electronic Society, WPES'18, page 54–65, New York, NY, USA, 2018. Association for Computing Machinery.