Private Projections & Variants

Xavier Carpent  
University of California, Irvine  
xcarpent@uci.edu

Sky Faber  
University of California, Irvine  
fabers@uci.edu

Tomas Sander  
Hewlett Packard Labs  
tomas.sander1001@gmail.com

Gene Tsudik  
University of California, Irvine  
gts@ics.uci.edu

ABSTRACT

There are many realistic settings where two mutually suspicious parties need to share some specific information while keeping everything else private. Various privacy-preserving techniques (such as Private Set Intersection) have been proposed as general solutions.

Based on timely real-world examples, this paper motivates the need for a new privacy tool, called Private Set Intersection with Projection (PSI-P). In it, Server has (at least) a two-attribute table and Client has a set of values. At the end of the protocol, based on all matches between Client's set and values in one (search) attribute of Server's database, Client should learn the set of elements corresponding to the second attribute, and nothing else. In particular the intersection of Client's set and the set of values in the search attribute must remain hidden.

We construct several efficient (linear complexity) protocols that approximate privacy required by PSI-P and suffice in many practical scenarios. We also provide a new construction for PSI-P with full privacy, albeit slightly less efficient. Its key building block is a new primitive called Existential Private Set Intersection (PSI-X) which yields a binary flag indicating whether the intersection of two private sets is empty or non-empty.

1. INTRODUCTION

A Private Section Intersection (PSI) protocol allows two parties: Server (S) and Client (C) with respective input sets A and B, to privately compute their intersection. As a result, Client learns only A ∩ B and |A|, while Server learns nothing beyond |B|. PSI has been widely studied and many concrete techniques have been proposed. One reason for PSI’s appeal is that it enables a wide range of privacy-agile applications: from personal genomics to querying cloud-resident databases. Also, unlike many other cryptographic protocols, PSI has been deployed in the real world. For example, Google uses a PSI variant in its advertising business.¹

¹This might be the very first instance of a Secure Function Evaluation (SFE) technique being used in the real world [29].

Different applications motivate several PSI variations, such as PSI-CA, where Client learns only the size of the intersection: |A ∩ B|. Another is PSI with Data Transfer (PSI-DT) [20] where, for each element in the intersection, Client learns one or more associated attribute(s). PSI-DT is particularly useful in privacy-preserving database applications where Server has a database DB and A = \{a₁,...,aₙ\} corresponds to one attribute (column) in DB. Such applications are quite realistic considering current popularity of cloud storage. In the simplest two-attribute case, Server's input is DB = \{(a₁, d₁),..., (aₙ, dₙ)\}, while Client's input is B = \{b₁, ..., bₘ\}. At the end of the protocol, Client learns N = \{(aᵢ, dᵢ) | ∃(i, j) ∋ bⱼ = aᵢ\}.

Unfortunately, PSI-DT generally allows Client to learn the exact relationship between the two attributes, i.e., each \((aᵢ, dᵢ)\) tuple. This also yields auxiliary information, e.g., frequency distributions (histograms) of both attributes. In this paper, motivated by some real-world scenarios, we explore a range of practical techniques that offer more privacy than PSI-DT. In particular, for the ideal privacy case, we introduce a new primitive called PSI with Projection (PSI-P) which allows the server to only learn the projection of the second attribute based on the view² formed by N. While the set-up is the same as in PSI-DT, Client only learns \(P = \{dᵢ | ∃(i, j) ∋ bⱼ = aᵢ\}\). Unlike PSI-DT, A ∩ B remains hidden. PSI-P is useful in scenarios where: (i) Client needs to learn all distinct values of one attribute for all matches of another attribute, and (ii) Server wants to keep secret which values of the latter attribute result in a match.

1.1 Specific Motivation

Indicators of Compromise (IOCs) are network- or host-based artifacts, which – when observed – indicate that a cyber-intrusion took place. Equivalently, IOCs consist of observables associated with malicious activity. Popular network-based IOCs include: IP addresses (plus port data), domain names, URLs, email addresses, user agents, ASN, ISP, netblocks and SSL certificates. A common strategy for an organization to monitor its networks for a breach involves collecting a wide range of event data (firewall logs, web proxy logs, IDS/IPS alerts, etc.) and matching it against known IOCs. If a match is found, an alert is issued and a (human) incident responder investigates further. To respond effectively, in addition to learning that an IOC was detected, an incident responder must learn what this IOC means. For example, this might be the very first instance of a Secure Function Evaluation (SFE) technique being used in the real world [29].

²We use the term view in the database sense, i.e., a virtual table or an intermediate result of a query.
if detected IOCs are associated with a nation state target-
ing intellectual property (IP), effective response would be
different than if the IOCs point to an e-crime syndicate tar-
geting a company’s credit card database. Information that
a responder needs is IOC context, which includes: means of
mitigation (e.g., which systems to patch), vulnerabilities ex-
loited, associated malware families, threat actors and their
motivations, modus operandi, tooling, active campaigns, as
well as additional IOCs for which a responder should search.

In this scenario, a set of IOCs needs to be matched with
network event data. Additional data transferred for each
matching IOC is its context. The novel requirement is that
which indicators match must remain secret. This is appar-
ent in cases where IOCs themselves are highly sensitive, e.g.,
associated with a nation state engaging in industrial or po-
litical espionage, or terror groups planning to sabotage criti-
cal infrastructures, or e-crime syndicates targeting PII. Such
sophisticated attackers are routinely tracked by government
agencies and security vendors specializing in high-value, so-
called “threat intelligence”. Also, cyber-security teams in
targeted organizations collect such IOCs in their own inves-
tigations. Often, these IOCs and their contexts are classi-
fied.

Leakage of IOCs can tip off the attacker that the intru-
sion is detected and cause it to change the infrastructure
and tooling, thus setting back the investigation [12]. IOC
leakage also gives away methods and capabilities of the in-
vestigators. Aside from malicious disclosure, receivers of
sensitive IOCs need to be trusted not to mishandle them. If
IOCs are accidentally blocked at the firewall or searched on
the Web, this can signal to the attacker that the intrusion
is detected, thus compromising gathered intelligence. Un-
derstandably, organizations are extremely reluctant to share
IOC-related data. The unfortunate consequence is that par-
ties who would otherwise greatly benefit from information
sharing (e.g., potential victims of massive IP theft) are un-
prepared and not protected because the intelligence never
reaches them. For that reason, the US private sector has
long requested better information sharing. In response, the
President issued an Executive Order requiring US govern-
ment to improve its information sharing practices [27].

As shown in the rest of this paper, PSI with Projection ad-
dresses the above challenge. It can be used to match highly
sensitive IOCs, without disclosing them. Contextual data
associated with each IOC is carefully crafted such that it
is both useful and actionable to Client, while not giving
away extra information. For example, a given IOC context
might advise an organization to patch certain systems or put
specific controls in place, without disclosing further details
about the attack campaign that these measures are sup-
posed to thwart. At the same time, privacy of the receiving
organization (Client’s) is preserved: it does not disclose its
event logs to a third party or the government – something
few organizations are willing to do.

1.2 Roadmap and Contributions

In this paper, we construct several practical protocols for
PSI with Projection and related capabilities. Our first ap-
proach is based on well-known Oblivious Polynomial Eval-
uation (OPE) PSI and PSI-DT protocols. It achieves optimal
round complexity of two and has the advantage of being con-
ceptually simple. However, it falls short in two important
ways: (1) OPE-based protocols are not the most efficient
PSI and PSI-DT techniques, and (2) it involves some infor-
mmation leakage, which we discuss below.

Our second protocol also achieves linear-time complexity.
It is a modified variant of the basic PSI-CA protocol from [7]
that performs data transfer using Oblivious Pseudo-Random
Function (OPRF) techniques [20]. The resulting protocol
hides the set intersection from Client, which is the main
privacy property we want to achieve. It is fast, practical
and sufficient for many application scenarios. Nonetheless,
it has the same information leakage as the first.

Next, we address the issue of information leakage. In an
ideal PSI-P, Client must only learn the true projection \( P = \{ d_i \mid \exists (i,j) \ni b_j = a_i \} \). (Note that \( P \) is a proper set,
not a multi-set). However, in our OPE- and OPRF-based
protocols Client actually learns:

\[ P' = \{ (d_i, \#(d_i)) \mid \exists (i,j) \ni b_j = a_i \} \]

where \#(d_i) is the number of times \( d_i \) occurs as the second
attribute (or contextual data field) in all records where a
match happens, i.e., any \( b_j = a_i \). Extra information in \( P' \)
beyond that in \( P \) – corresponds to a frequency distribution
or a histogram for the contextual data.

Revealing this extra information is not always a problem;
but, in fact, it might even be desirable. However, suppose that
Server’s contextual data amounts to a single message “Call
NSA." If Client learns that this message was triggered 1,000
times, it also learns that exactly 1,000 elements of Client’s
input set are on the NSA IOC watch-list. That gives away
much more information than simply: “one or more elements
of Client’s input are on the watch-list.”

Furthermore, security event logs can be massive. For ex-
ample, in one specific large enterprise (with which the au-
thors are familiar) collects on the order of 5 Billion events
per day for security monitoring. Clearly, these logs must
be aggressively filtered before being fed to our algorithms.
The size of filtered event-logs will influence how well match-
ing indicators are hidden within Client’s input and can be
negotiated between Server and Client if needed.

Consider another example illustrating the risks of disclos-
ing frequencies: the number of a given IOCs sightings de-
notes how often that IOC was seen on the Internet. It re-
resents valuable feedback for threat intelligence providers
to receive sightings information from customers for IOCs
provided to them. A threat intelligence provider can draw
conclusions about which types of organizations are at risk,
which IOCs are used in live attack campaigns, etc. Mean-
while, for an organization, it is less sensitive to disclose that
an IOC was sighted in its infrastructure. However, disclosing
that it was seen 10,000 times is more sensitive, since it im-
plies that the organization had a serious breach. Organiza-
tions are extremely reluctant to share breach data which ex-
poses them to the risk of regulatory fines and reputation loss,
or gives away vulnerabilities of their infrastructure. There
are several ways to deal with IOC sightings in practice; how-
ever, we stress that frequencies of occurrence must be dealt
with carefully. It is thus highly desirable to have protocols
for matching threat intelligence data that do not leak this
information.

Motivated by the above, our third protocol is a modifica-
tion of the second. By slightly increasing run-time (at most
by a factor of 2) it improves privacy by only disclosing raw
frequency distribution, while hiding the mapping between
the set of contextual data attribute values and their frequen-
cies. For example, if the projection is \( P = \{d_1, d_2\} \) where \( d_1 \) occurs 2, and \( d_2 - 10,000 \) times, Client learns \( \{2, 10,000\} \) and not which of the two elements occurred 10,000 times. Breaking the association between frequencies and projected elements is a significant privacy improvement.

However, our ultimate goal is to design a protocol that leaks no information at all about frequencies, i.e., realizes PSI with Projection with full privacy. This turns out to be hard. Short of reverting to general Secure Function Evaluation (SFE) techniques, current PSI protocols do not support deduplication needed to achieve PSI with Projection.

However, we approach the problem by reducing PSI-P to a primitive we call Existential PSI (PSI-X): for Server and Client with sets \( A \) and \( B \) PSI-X returns 1 if \( A \cap B \) is non-empty, and 0 otherwise. Except \(|A|\) and \(|B|\), no other information about \( A \), \( B \) or \( A \cap B \) is revealed to either party. This privacy requirement exceeds that of a PSI-CA that outputs \(|A \cap B|\) with sets of elements is a significant privacy improvement. Breaking the association between frequencies and projected values, i.e., \( PSI-P-H \), we now define PSI-X using BGN encryption.

We denote a given \( L \) variant by appending its symbol to the string “PSI”, e.g., for \( L = \{I, H\} \) we write PSI-\( L \) as PSI-P-H. We now make some observations on this notation and relationships among protocol variants, summarized in Figure 1.

We give a \( L \) variant by appending its symbol to the string “PSI”, e.g., for \( L = \{P, H\} \) we write PSI-\( L \) as PSI-P-H. We now make some observations on this notation and relationships among protocol variants, summarized in Figure 1.

### Table 1: Protocol Notation

<table>
<thead>
<tr>
<th>( L )</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( I )</td>
<td>Intersection ( I = A \cap B )</td>
</tr>
<tr>
<td>( CA )</td>
<td>Cardinality of ( A \cap B ), ( CA =</td>
</tr>
<tr>
<td>( DT )</td>
<td>Subset of records in ( DB ) corresponding to elements in ( A \cap B ): ( DT = {(a, d) \in DB \mid a \in A \cap B} )</td>
</tr>
<tr>
<td>( P )</td>
<td>Projection of ( DT ) onto second attribute of ( DB ): ( P = {d \mid (a, d) \in DT} )</td>
</tr>
<tr>
<td>( H )</td>
<td>Histogram of second attribute in ( DT ), i.e., ( \forall d \in P ), frequency of occurrence of ( d ) in ( DT ): ( H = {(d, f_d) \mid d \in P, f_d =</td>
</tr>
<tr>
<td>( F )</td>
<td>Frequency distribution of second attribute in ( DT ), equivalent to random permutation of second attribute of ( H ): ( F = \Pi_2({f_d \mid (d, f_d) \in H}) ) for random permutation ( \Pi )</td>
</tr>
<tr>
<td>( X )</td>
<td>One-bit value indicating if the intersection is empty, ( b \in {0, 1} ) where ( b = 1 \iff I = A \cap B \neq \emptyset )</td>
</tr>
</tbody>
</table>

![Figure 1: Relationships between information disclosed by various PSI protocols.](image-url)

- Standard PSI protocols that compute the set intersection \( A \cap B \) are called PSI-I in our notation.
- Our notions of PSI-CA and PSI-DT coincide with their common prior use in the literature.
- Our representation is not unique, e.g., PSI-DT can also be written as PSI-DT-I-P-H.
- Notation \( X \rightarrow Y \) means: “knowledge of \( X \) implies knowledge of \( Y \)”.
- PSI-DT discloses the maximum amount of information among all protocols in this family.
- PSI-X discloses the minimal amount of information.

The strategy for the rest of this paper is as follows. Our main goal is to construct a PSI-P protocol. We start with (known) PSI-DT protocols and modify them. The first two protocols in Section 3 remove knowledge of the intersection from Client and yield PSI-P-H protocols. Next, the third protocol realizes PSI-P-F; it achieves better privacy than...
the first two by removing any linkage between elements in $P$ and their frequencies, i.e., we progress from a protocol disclosing $H$ to a protocol disclosing $F$ and $P$. In Section 4, we remove knowledge of $F$ from Client and construct a PSI-P protocol. This is the hardest step. The key building block for a PSI-P protocol is a sub-protocol for PSI-X.

### 2.2 Privacy Advantages of Projection Variants

Before proceeding to concrete protocols, we consider degrees of privacy attainable with various envisaged PSI-based projection techniques, compared to PSI-DT.

#### 2.2.1 PSI-P

Recall that $m$ and $n$ are respective sizes of Client and Server inputs. From the privacy perspective, in an ideal PSI-P protocol, Client learns:

$$P = \{d_i \mid \exists(a_i, A, b_j \in B) \supset b_j = a_i\}$$

However, in some extreme cases, PSI-P offers no privacy over PSI-DT. For example, if $w = |P| = 0$, then $A \cap B = 0$ and no additional information is learned in either protocol. Also, if $m = 1$, then $w \in \{0,1\}$ and both PSI-P and PSI-DT yield equivalent knowledge. Nonetheless, in general, PSI-P that results in Client learning $P$ offers more privacy to Server than PSI-DT which lets Client learn $N=\{(a_i, d_j) \in A \mid \exists(b_j \in B) \supset b_j = a_j\}$. In other words, PSI-P offers no privacy over PSI-DT if $m = 1$ or $w = 0$. The same is true when $w = m$.

Aside from these corner cases, PSI-P can offer a significant privacy advantage over PSI-DT. In general, we can express the number of possible mappings of elements in $B$ to elements in $P$ as:

$$M_P(m, w) = \left\lbrack \frac{m + 1}{w + 1} \right\rbrack \cdot w!$$

where $\left\lbrack \frac{m}{n} \right\rbrack$ represents the Stirling number of the second kind, i.e., the number of ways to partition $n$ objects into $k$ non-empty subsets. Eq. (1) is the number of partial surjective functions from $B$ to $P$. The mappings are partial because some elements in $B$ might not be in $A \cap B$, and surjective because each element in $P$ has at least one corresponding element in $B$. Let $M_B(m, w)$ be the number of total surjective functions from $B$ to $P$. This is known to be: $\left\lbrack \frac{m}{w} \right\rbrack \cdot w!$; see for instance the twelve-fold way [20]. For each non-total function $f$, let $f' : B \rightarrow P \cup \{\epsilon\}$ where $f'(x) = \epsilon$ for every $x$ undefined through $f$. The number of such functions is $M_F(m, w + 1)$. Adding to this number the partial functions that are already summed up gives:

$$M_F(m, w + 1) = M_F(m, w) = \left\lbrack \frac{m}{w + 1} \right\rbrack \cdot (w + 1)! + \left\lbrack \frac{m}{w} \right\rbrack \cdot w!$$

by the recurrence relation of Stirling numbers of the second kind $\left\lbrack \frac{n+1}{k} \right\rbrack = k \left\lbrack \frac{n}{k} \right\rbrack + \left\lbrack \frac{n}{k-1} \right\rbrack$. This expression also assumes (perhaps ideallyistically) that Client has no extra information about DB and its second attribute is uniformly distributed.

Given $M_F(m, w)$, we can capture the exact degree of privacy ($D_{prev}$) that PSI-P offers over PSI-DT as:

$$D_{prev} = 1 - \frac{1}{M_F(m, w)}$$

As reflected by aforementioned “corner cases”, $D_{prev} = 0$ for $m = 1$ or $w = 0$. We also note that perfect privacy ($D_{prev} = 1$) is unattainable for $m \geq 1$. Indeed, $D_{prev} = 1$ only if $m = 0$, i.e., $B = \emptyset$.

#### 2.2.2 PSI-P-H

PSI-P-H allows Client to learn the histogram $H$ (see Table 1) which also leaks $CA = |A \cap B|$. It thus yields more information than $P$ given by PSI-P, though less information than $N$ given by PSI-DT.

Let $M_H(m, w, H)$ be the number of possible mappings from elements in $B=\{b_1, \ldots, b_m\}$ to elements in $P$, given projection histogram $H$. Let $f_i$ relate to the frequency of the $i$-th element in $P$, that is such that $(P_i, f_i) \in H$. The number of such mappings is the product of the number of ways to choose $f_1$ elements among $m$ and the number of ways to choose $f_2$ elements among $m - f_1$, etc. That is:

$$M_H(m, w, H) = \prod_{k=1}^{w} \left( \frac{m - \sum_{i=1}^{k-1} f_i}{f_k} \right)$$

$$= \prod_{k=1}^{w} \frac{(m - \sum_{i=1}^{k-1} f_i)!}{f_k!(m - \sum_{i=1}^{k-1} f_i)!}$$

$$= \frac{m!}{(\prod_{k=1}^{w} f_k!)(m - \sum_{i=1}^{k} f_i)!}$$

Note that this number does not rely on the order in which projection elements are considered.

#### 2.2.3 PSI-P-F

PSI-P-F allows Client to learn $P$ and $F$, both of size $w$ (see Table 1). It thus offers more privacy than PSI-P-H, over PSI-DT. The number of mappings from elements in $B=\{b_1, \ldots, b_m\}$ to projection set elements, given the randomly permuted projection histogram $F$, is:

$$M_F(m, w, F) = M_H(m, w, H) \cdot \pi_F,$$

with $\pi_F$ the number of possible histograms related to $F$. That is, the number of permutations with repetition of indistinguishable objects:

$$\pi_F = \frac{w!}{\prod_{i=1}^{n_i} i!}$$

with $n_i = |\{i \in F\}|$, the number of occurrences of a frequency $i$ in $F$. For example, with $m = 8, w = 4, H = (1, 3, 1, 1)$, we have:

- $M_H(8, 4, (1, 3, 1, 1)) = \frac{w!}{\prod_{i=1}^{n_i} i!} = 6720$
- $M_F(8, 4, (1, 1, 1, 3)) = 6720 \cdot \frac{4!}{3!} = 26880$
- $M_F(8, 4) = 6951 \cdot 4! = 166824$

In the corner case, when $m = w$, and $F = H = (1, 1, \ldots, 1)$, we have: $M_F(m, w, F) = M_H(m, w, H) = M_P(m, w) = m!$. Finally, note that the same measure of privacy $D_{prev}$ defined for PSI-P can be used for PSI-P-H and PSI-P-F, since in each case all mappings are equally likely.

### 3. PROTOCOLS

We now proceed to construct a series of protocols with variable degrees of privacy, falling into the range between PSI-DT and PSI-P.

#### 3.1 Initial PSI-P-H Protocol

Our initial goal is for Client to learn the second attribute of $DT$ without learning $A \cap B$. As a result, Client learns projection $P$ and histogram $H$, as defined in the previous
section. We can achieve this easily by modifying any OPE-based\(^3\) PSI protocol (e.g., [14]), as described below.

Setup: Server and Client respective inputs are:

\[ DB = \{(a_1,d_1),\ldots,(a_n,d_n)\} \quad \text{and} \quad B = \{b_1,\ldots,b_m\}, \quad \text{respectively.} \]

\[ A = \{a_1,\ldots,a_n\} \quad \text{is the first attribute of} \quad DB = \{(a_1,d_1),\ldots,(a_n,d_n)\}. \]

Let \( E \) be an additively homomorphic encryption scheme, e.g., Paillier [24], and \( Enc \) be a symmetric authenticated encryption scheme (AKE), e.g., [2].

1. **Client** generates a new public/secret key-pair for \( E \).

   **Client** encodes each element from \( B = \{b_1,\ldots,b_m\} \) as a root of a polynomial \( P := \prod_{i=1}^{n} (X - b_i) \). Next, **Client** encrypts the coefficients of \( P \) using \( E \) and sends the resulting list of encrypted coefficients to **Server**, together with the new public key.

2. For each \( a_i \in A = \{a_1,\ldots,a_n\} \), **Server** chooses, at random, symmetric key \( k_i \) and identifier \( a_i \). It then computes \( q_i := Enc_{k_i}(d_i) \) and forms \( Q = \{q_1,\ldots,q_n\} \).

3. For each \((a_i,d_i) \in DB\), **Server** computes \( E(r_i \cdot P(a_i) + k_i \| a_i) \), with \( r_i \) random numbers, using the homomorphic properties of \( E \) and adds resulting elements to a list \( L \). **Server** shuffles \( L \) and \( Q \) and sends the shuffled sets to **Client**.

4. **Client** decrypts elements in \( L \) and parses the result into \( k \| o \). If an element in \( Q \) is marked \( o \), **Client** decrypts it with \( k \). If the integrity check holds, **Client** obtains \( d \). Otherwise, **Client** discards this element.

Because of an AKE scheme, **Client** can distinguish whether decryption of \( q_i \) in Step 4 is a random value or an actual attribute (that it needs to learn) associated with an element in \( A \cap B \).

Computation cost is dominated by \( O(nm) \) exponentiations, if we use the Paillier scheme. By applying Horner’s rule and binning techniques we can reduce computation to \( O(n \log \log(m)) \) exponentiations [14].

In this protocol, **Client** learns the histogram \( H \) without learning \( A \cap B \). Recall that our ultimate goal is to hide the frequencies, or equivalently, to deduct the projection. However, this is difficult in protocols that work “point-by-point”, such as the one above. For example, if \( A = \{a_1,a_2\} \) and \( DB = \{(a_1,d_1),(a_2,d_2)\} \), **Client** learns two independent outputs that could yield further information, depending on \( B \). What is missing is the ability to aggregate two independent outputs, such that the cases of \(|A \cap B| = 1\) and \(|A \cap B| = 2\) are indistinguishable to **Client**. This illustrates a basic limitation of such “point-by-point” protocols in attaining PSI-P with maximal privacy. In contrast, protocols in Section 4 below provide this kind of aggregation capability.

### 3.2 A More Practical PSI-P-H Protocol

We now present a PSI-P-H protocol with a linear computation cost, which is much more efficient than the initial protocol. The intuition is to modify the basic PSI-CA protocol from [7]\(^4\) to perform data transfer with techniques similar to traditional OPRF-based PSI-DT protocols, such as [20]. Specifically, as in PSI-CA, **Client** and **Server** evaluate an Oblivious Pseudo-Random Function (OPRF) on each of **Client’s** set elements. Then, for every record in \( DB \), **Server** transfers its second-attribute value encrypted under a unique key derived from the pre-image of the OPRF evaluated over the value of the first attribute of the same record. This way, **Client** only decrypts second-attribute values \((d_i,s)\) that correspond to elements in the intersection.

This protocol needs a semantically secure symmetric-key encryption function \( E_k(\cdot) \), a corresponding key derivation function \( KDF(\cdot) \), two random oracles \( H(\cdot)^5 \) and \( H'(\cdot) \), random permutation \( \Pi(\cdot) \), and a keyed pseudorandom function \( F_k(x) = H'(g^k H(x)) \), based on the following group parameters: primes \( p \) and \( q \) (where \( p = 2q^2 + 1 \)) and a generator \( g \) of order \( q \) in a subgroup of \( \mathbb{Z}_n^* \). The resulting PSI-P-H protocol is shown in Figure 2. The notation is largely self-explanatory, except for \( H[d] \) which denotes an element \((d,f) \in H \) where \( d \) is a data item and \( f \) is its frequency up to now. \( H[d,f] \) denotes the frequency attribute/field of element \((d,f) \).

On its own, this protocol comes very close to the desired PSI-P functionality. Unfortunately, it does not address the deduplication problem. At the same time, for some applications where this information leakage is acceptable or desired, this is the most efficient protocol that we identified thus far.

It is also easy to see that this protocol, being a minor variation of the original PSI-CA, leaks no information beyond the histogram, assuming that all encrypted data values \( d_j \) are of uniform size. See Section 3.6 for more details.

### 3.3 PSI-DT with Deduplication

The main challenge in constructing an ideal PSI-P protocol is how to handle data **deduplication** to avoid privacy leakage. Another issue is bandwidth: ideally, the protocol should transfer associated data values (corresponding to the intersection computed on the first attribute) only once. This is particularly the case if these values are large, e.g., photos or video clips. We now present a trick for reducing privacy leakage from frequencies, and also lowering bandwidth consumption.

We construct a protocol based on two rounds of PSI-DT. First, we introduce a randomly generated tag, \( t_s \leftarrow \{0,1\}^L \), for each unique data value \( d_i \). This produces \( DB^A \), a mapping between \( A \) and randomly generated tags, where distinct elements in \( A \) mapping to the same data value in \( DB \) are associated with the same tag in \( DB^A \). Another data structure \( DB^T \) maps distinct tags to data values.

Then, **Client** and **Server** run PSI-DT on these (potentially duplicated) tags instead of data values. Note that while the number of tags and distinct data values is the same, the size of the former is uniform and may be significantly smaller.

Next, **Client** filters all duplicate tags and then pads the resulting set of tags with randomly generated tags, up to the size of \( \min(m,n) \). This is necessary to preclude **Server** from learning \( CA \). Although padding incurs additional computation in PSI-P-H, dummy tags need not be blinded by **Client**; they should only be indistinguishable from actual blinded elements.

Then, **Client** runs PSI-DT for the second time, its input being the deduplicated padded set of unique tags. **Server** input in this round includes the tags and their mapping to data items. This way, **Server** only sends each encrypted data item once, regardless of the number of set elements which map to it. Details are shown in Figure 9 in Appendix A.

---

\(^3\)OPE: Oblivious Polynomial Evaluation.

\(^4\)Note that in [7], the protocol contains an additional DH-like key exchange that was subsequently removed in the latest version [8]. Private communication with the authors clarified that this key exchange is not required.

\(^5\)The range of \( H \) is the set of elements from the group generated by \( g \).
At the end of the protocol Client learns the same information as in PSI-DT, in particular, the histogram $H$. However, each encrypted data item is only transferred once, in the second round. If data items are large this can result in appreciable bandwidth savings. However, these savings come at increased computation by a factor of at most 2, since PSI-DT is run twice.

Also, Client learns $k$ – the number of distinct data elements in $DB$. This can be avoided (see 3.5) at the cost of negating bandwidth savings. Indeed, disclosure of $k$ is inherent to each data element being sent only once.

3.4 Practical PSI-P-F Protocol

In our previous protocols Client learns $H$ – the histogram of data items in projection $N$. We now show how to construct an efficient (linear-time) PSI-P-F protocol. As argued earlier, disclosing only the frequencies to Client (and not their linkage to elements in the projection set) is a major practical privacy gain.

To construct a practical PSI-P-F protocol, we use the deduplication technique described in Section 3.3 and reflected in Figure 9, except, instead of PSI-DT, we run PSI-P-F twice. The resulting protocol is a practical, linear-time instantiation of PSI-P-F. Its offers better privacy since Client only learns the histogram of tags in the first execution of PSI-P-H. The second execution of PSI-P-H removes the mapping between tags and second-attribute values, thereby also removing any association between frequencies and second-attribute values. The protocol is shown in Figure 3.

Note that, as with PSI-DT with deduplication, Client learns $k$ – the number of distinct second-attribute values in $DB$. This is not the case in PSI-P-H from Section 3.2. See Section 3.5 for a discussion.

3.5 Hiding Number of Distinct Data Elements

The number of distinct data elements $k = |\{d \mid (a, d) \in DB\}|$ is revealed in both PSI-DT with deduplication and PSI-F. This is inherent to transmitting each (encrypted) data element (aka second-attribute value) only once. This leakage in PSI-P-H can be avoided by running a simple PSI-DT protocol; however the sole advantage of PSI-DT with deduplication over PSI-DT is bandwidth savings.

In PSI-P-F, leakage can be avoided by having Server pad $DB$ to $n \geq k$ pairs, using random tags (indistinguishable from real tags by Client) and dummy encryptions. Although this would result in no bandwidth savings, it might be useful in scenarios where data elements are small and/or privacy of $k$ is important.

Given that this leakage can be avoided, it is omitted from the final output of PSI-DT with deduplication as well as PSI-P-F in Section 3.6. Finally, this leakage is absent in PSI-P presented later in Section 4.

3.6 Correctness & Privacy

**Definition 2 (Correctness of PSI-L).** If both parties are honest, at the end of protocol execution on respective inputs $[(B), (A, DB)]$, Client outputs $(n, \ell)$ and Server outputs $m$.

**Definition 3 (Privacy of PSI-L).** If both parties are honest, at the end of protocol execution on respective inputs
Our PSI-P-H construction is substantially based on the PSI-CA technique from [7]. The argument for its privacy is therefore very similar, and relies on the very same assumptions, i.e., semi-honest (Hbc) participants, DDH, One-More-DH, and Gap-One-More-DH assumptions; we refer to [7] for further details.

**Theorem 3.** PSI-P-H is private.

**Proof (Sketch).** Server privacy: The main addition of PSI-P-H over PSI-CA is that, in Server’s reply, encrypted data elements are associated with hashed blinded elements. Client can not learn anything from encrypted elements for which it does not know the decryption keys, based on security of $E(\cdot)$. An decryption key can be learned only if the corresponding element is in the intersection.

Client privacy: Server learns nothing more than in PSI-CA, which is private. □

**Theorem 4.** PSI-P-F is private.

**Proof (Sketch).** Server privacy: Based on the definition of PSI-P-H and its privacy (Theorem 3), Client learns $P$ and $H$ related to $B$ and $DB^H$, and (padded) $T'$ and $DB^T$, respectively.

In the first exchange, this reveals (from $DB^A$) tags mapped to items in $A \cap B$ and their histogram. Nothing is learnt from tag values since they are selected at random. Even though the histogram $H^A$ leaks frequency distribution of tags, it conveys no information about the histogram of data items, since their order is randomly permuted.

The second exchange reveals the data items mapped to tags received in the first exchange. The histogram $H^T$ carries no useful information since mappings are unique in $DB^T$. 

---

**Figure 3:** PSI with Projection and Frequency distribution (PSI-P-F).
It is computationally infeasible for Client to learn the tags corresponding to items outside the intersection and learn additional data items.

Client privacy. Server learns nothing beyond \( m \) from either interaction (by Theorem 3). In particular, in the second exchange, since Client input is padded to \( \min(m, n) \), Server can not learn CA. It also can not distinguish its own tags from padding tags generated by Client, since all tags are blinded by the latter.

4. PSI-P & PSI-X: HIDING FREQUENCIES

Our strategy in this paper is to progress from a PSI-DT (=PSI-DT-I-P-H-F) protocol to a PSI-P protocol by removing the characteristic functions efficiently. Instead, we store the blinded by the latter.

It is computationally infeasible for the latter. In particular, the second exchange, since Client input is padded to \( \min(m, n) \), Server can not learn CA. It also can not distinguish its own tags from padding tags generated by Client, since all tags are blinded by the latter.

4. PSI-P & PSI-X: HIDING FREQUENCIES

Our strategy in this paper is to progress from a PSI-DT (=PSI-DT-I-P-H-F) protocol to a PSI-P protocol by removing the characteristic functions efficiently. Instead, we store the blinded by the latter.

It is computationally infeasible for the latter. In particular, the second exchange, since Client input is padded to \( \min(m, n) \), Server can not learn CA. It also can not distinguish its own tags from padding tags generated by Client, since all tags are blinded by the latter.
Theorem 6. The number of ciphertext multiplication operations $C_X$ in PSI-X is minimized when

$$N = \frac{mn \log 2}{\log^2 2},$$

giving

$$C_X = \frac{mn \log P_X}{\log^2 2},$$

for a desired total probability of false positive $P_X$. For this choice of $N$ the number of rounds is $R = \frac{\log P_X}{\log 2}$, with $P$ the probability of a false positive in one round.

Proof. See Appendix B □

We now show PSI-X has the desired privacy properties. PSI-X uses as subprotocols an OPRF protocol [15, 19] and the SFE protocol for evaluating 2-DNF formulas from [3]. The 2-DNF protocol is secure under the Subgroup Decision Assumption.

Theorem 7. Given a secure OPRF protocol and under the Subgroup Decision Assumption PSI-X is private.

Proof (Sketch). PSI-X has three steps. First Client and Server engage in an OPRF protocol. Second there is a hashing step with a randomly chosen 2-universal hash function $h$. Third there is a secure evaluation of a 2-DNF formula Eq. (2) on the outputs from the hashing step. Hashing and 2-DNF formula evaluation are repeated in $R$ independent rounds.

Client privacy The OPRF protocol does not leak information to the Server about Client’s input due to its privacy guarantees. In Step 3 Server additionally receives a set of BGN-encrypted values from Client. Under the Subgroup Decision Assumption BGN encryption is semantically secure and thus Server does not learn anything from these ciphertexts.

Server privacy We show that a simulator $C^*$ can simulate the Client’s view of the protocol from its input and the protocol output. The Client’s view consists of OPRF($B$), i.e. OPRF applied to its input $B$. In the $i$’th round of the protocol Client sees a randomly chosen 2-universal hash function $h$, the result from the secure evaluation of the 2-DNF formula applied to $h_i(\text{OPRF}(B))$ and the binary output $o_i := \text{BGN-Is-Zero}_{priv}(X_{o_i})$ of the 2-DNF computation.

If the PSI-X protocol outputs 1 then all the $o_i$ have been 1 by definition of the protocol. In the simulation $C^*$ generates a set of random values in the range of the PRF. This set is indistinguishable from the set OPRF($B$) without knowledge of $k_{prf}$. In addition he sets each bit in bit vector $UA$ to be 1 at every location, and then proceeds as would the Server of the ideal model. This simulation is indistinguishable from the execution of the real protocol.

If the PSI-X protocol outputs 0 we know that Client and Server input sets $A$ and $B$ are disjoint. $C^*$ simulates the Client’s view of the protocol by first generating a set of random values for OPRF($B$) as before. Then the bit vector $UA_i$ is created randomly so that $X_i = s(UA_i \cdot EUB_i)$ is non-zero with probability $P$ (the error probability) and 0 otherwise. Then $X_i$ is sent as before. Again the OPRF pro-
tocol and the secure 2-DNF evaluation protocol are given as in the ideal model. This simulation is computationally indistinguishable from the Client’s view of the real PSI-X protocol execution.

4.2 Full PSI-P Protocol

The main idea is to use PSI-X to determine, for each second attribute value \( d_i \), whether the elements in \( A \) that map to each second attribute value \( d_i \) intersect with \( B \). Such elements of the second attribute are then transferred to Client using PSI-DT. The full protocol is depicted in Fig. 5 and described below.

Let \( D = \{ d_i \mid (a_i, d_i) \in DB \} \) denote a set of distinct second-value attribute values, such that \( |D| = k \leq n \). Let \( \epsilon \) denote a dummy value. Server first builds a vector \( V \) comprised of elements of \( D \), padded with \( n - k \) dummy values, such that \( |V| = n \), and then shuffles it. For each \( d = V_i \), Server computes its support, defined as: \( \text{support}(d) = \{ a_i \in A \mid (a_i, d_i) \in DB \} \), with \( \text{support}(\epsilon) = 0 \). Then, support is padded such that the size of the ensuing vector is \( n \). The padding scheme pad() that we suggest is to use elements from a subset of \( U \) disjoint from the possible values in \( A \) and \( B \) and agreed upon by Client and Server. Next, Client and Server engage in a PSI-X protocol with respective inputs \( B \) and \( \text{pad}([], \text{support}(V_i)) \). After this is done for each element of \( V \), Client and Server engage in a final PSI-DT protocol for the elements of \( V \) that resulted in PSI-X output 1.

False positives in PSI-X executions result in elements of \( D \) being incorrectly transferred. This can be alleviated by tuning the number of rounds in PSI-X.

**Theorem 8.** Let Server’s input be \( DB = \{ (a_1, d_1), \ldots, (a_n, d_n) \} \) and Client’s input be \( B = \{ b_1, \ldots, b_m \} \). PSI-P is a randomized algorithm so that at the end of the protocol Client learns a set \( Pr \) such that \( Pr \supset \{ d_i \mid \exists (i, j) \ni b_j = a_i \} \). With probability at least \( 1 - nP^R \) equality holds, i.e. \( Pr = \{ d_i \mid \exists (i, j) \ni b_j = a_i \} \), where \( P^R \) is the error probability of the PSI-X sub-protocol.

The proof follows directly from the prior discussion and Theorem 5.

PSI-P uses as subroutines PSI-X and PSI-DT. It now follows from Theorem 7.

**Theorem 9.** Given a secure OPRF protocol, a private PSI-DT protocol and under the Subgroup Decision Assumption, PSI-P privately computes \( Pr \supset \{ d_i \mid \exists (i, j) \ni b_j = a_i \} \). With probability at least \( 1 - nP^R \), no additional element beyond the actual projection is revealed, i.e. \( Pr = \{ d_i \mid \exists (i, j) \ni b_j = a_i \} \), where \( P^R \) is the error probability of the PSI-X sub-protocol.

There are \( n \) executions of PSI-X with sets of size \( m \) and \( n \) and one execution of PSI-DT with a set of size \( n \).

**Theorem 10.** The number of ciphertext multiplication operations \( C_p \) is given by

\[
C_p \approx \frac{mn^2 \log n}{\log^2 2} + C_{DT}(n) \in O\left( mn^2 \log n \right),
\]

for a desired total probability of false positive \( P_r \), and with \( C_{DT} \) the cost of the PSI-DT execution.

**Proof.** See Appendix B

Although considerably more expensive than the construction given in Sect. 3.4, PSI-P protects privacy of both Server and Client and does not disclose any other information than the projection.

Some elements in its construction help achieve that. Firstly, there are \( n \) rounds of PSI-X instead of \( k \) (although \( n - k \) of these rounds are dummy rounds) so that client does not learn \( k \), the number of different data elements. Additionally, the support of (possibly dummy) data elements are padded to sets of size \( n \) so that client does not learn the histogram of data items. Finally, the data transfer operation is done after the \( n \) rounds of PSI-X so as to not give information on the server about the result of the individual PSI-X executions.

5. EXPERIMENTS

5.1 PSI-P-H and PSI-P-F

We implemented PSI-P-H (Fig. 2) and PSI-P-F (Fig. 3) in order to have a practical measure of their efficiency. We used a custom fast implementation of PSI-DT that uses OPRF as a building block for PSI-P-F. Experiments were run on an Intel i7-4710HQ CPU.

Figures 6, 7, and 8 represent the cost of PSI-P-H and PSI-P-F for different values of \( m \) and \( n = 100 \) for when the intersection \( A \cap B \) is respectively empty, half the size of \( B \), and the entire set \( B \). The overhead of PSI-P-F with respect to PSI-P-H corresponds to the second instance of PSI-P-H within PSI-P-F. This second instance has a small fixed cost, and depends linearly on the size of the intersection. For PSI-P-H however, the size of the intersection does not matter.

![Figure 6: Comparison of the cost of PSI-P-H and PSI-P-F for varying m with n = 100 and |A ∩ B| = 0.](image)

5.2 PSI-P

For PSI-X (Fig. 4) and PSI-P (Fig. 5), we used Relic [1], an efficient library in C for cryptographic protocols. It is used notably for the BGN encryption scheme in PSI-X. PSI-P also uses our OPRF-based PSI-DT implementation. Experiments are again run on an Intel i7-4710HQ CPU.

PSI-P is orders of magnitude slower than PSI-P-H and PSI-P-F. The clear bottleneck is the number of BGN ciphertext multiplications. Table 2 shows time data for different parts of PSI-P, under different parameters. The number of rounds is fixed to \( R = 7 \) (this corresponds to a \( P_{fail} < 1\% \) for PSI-X). The intersection size \(|A ∩ B|\) is set to 1 arbitrarily,
time [ms]
0 20 40 60 80 100
m
0
100
200
300
400
500
600
700
800
time [ms]
PSI-P-H
PSI-P-F

Figure 7: Comparison of the cost of PSI-P-H and PSI-P-F for varying m with n = 100 and |A ∩ B| = m/2.

Figure 8: Comparison of the cost of PSI-P-H and PSI-P-F for varying m with n = 100 and |A ∩ B| = m.

but this has no impact on the protocol efficiency (otherwise this would constitute a leakage of CA). “BGN dot product” corresponds to the EUA_r · EUB_r operation in Fig. 4.

“Other BGN” corresponds to BGN encryptions, decryptions, and initialization in PSI-X. “PSI-DT” is the final operation in PSI-P. Time reported are aggregated for the Client and Server, but Server does most of the work, since it computes the BGN dot products.

6. PSI-P WITH THRESHOLD CONDITIONS

In some scenarios it will be useful to transfer the additional item d only if there has been a match of Client’s input B={b_1, ..., b_m} for at least t elements with Server’s input A={a_1, ..., a_n} for a threshold t. For example A may contain 300 domain names which are associated with a particular attack campaign. Although connecting with one domain in A may be inconclusive, if there were more than 20 connections from the Client’s network this might be strong evidence that an intrusion took place and consequently that additional data d should be sent to Client.

Our framework can handle threshold conditions naturally by applying a t-out-of-n Shamir secret sharing scheme to d. For each matching indicator, using PSI-P-H, a (different) share of d is transferred to Client. Client can then reconstruct d if and only if he learned at least t shares.

This yields a practical threshold scheme for (1) transferring additional data if and only if a threshold condition is met and (2) not revealing the matching indicators themselves. The privacy guarantees of this solution are likely sufficient in many practical scenarios.

However there is still some information leakage. In addition to d (the only data Client should learn if conditions are met), Client also learns threshold t and how many shares for reconstructing d he actually received (which could be more than t). We believe the question how to reduce information leakage in threshold scenarios is an interesting, open problem for future work.

7. RELATED WORK

The initial set of PSI and related protocol constructs was suggested by Freedman et al. [14]. It was based on obli-
Table 2: Times (in seconds) of different parts of the PSI-P protocol.

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>Total time</th>
<th>BGN dot product</th>
<th>Other BGN</th>
<th>PSI-DT</th>
<th>Remaining operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>3.05</td>
<td>2.29 (75.01%)</td>
<td>0.67 (21.86%)</td>
<td>0.10 (3.12%)</td>
<td>0.00 (&lt;0.01%)</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>13.79</td>
<td>11.33 (82.20%)</td>
<td>2.35 (17.08%)</td>
<td>0.10 (0.72%)</td>
<td>0.00 (&lt;0.01%)</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>107.43</td>
<td>91.35 (85.04%)</td>
<td>15.96 (14.85%)</td>
<td>0.12 (0.11%)</td>
<td>0.00 (&lt;0.01%)</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>843.76</td>
<td>722.90 (85.68%)</td>
<td>120.74 (14.31%)</td>
<td>0.12 (0.01%)</td>
<td>0.00 (&lt;0.01%)</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>53.45</td>
<td>45.41 (84.97%)</td>
<td>7.88 (14.74%)</td>
<td>0.16 (0.29%)</td>
<td>0.00 (&lt;0.01%)</td>
</tr>
<tr>
<td>20</td>
<td>5</td>
<td>211.56</td>
<td>179.96 (85.06%)</td>
<td>31.53 (14.91%)</td>
<td>0.07 (0.03%)</td>
<td>0.00 (&lt;0.01%)</td>
</tr>
</tbody>
</table>

8. CONCLUSION

This paper introduces the concept of Private Set Intersection with Projection (PSI-P). We construct several practical, linear time protocols that approximate PSI-P’s privacy guarantees and suffice in many practical scenarios. We also provide a new construction for PSI-P with full privacy that is slightly less efficient. The key building block is a new primitive we call Existential Private Set Intersection (PSI-X). PSI-X answers the basic question whether two sets A and B have an element in common without revealing anything else.

These constructions address important challenges in collaborative network security by making the sharing and protective use of sensitive IOCs (Indicators of Compromise) less risky.

9. REFERENCES


APPENDIX

A. PSI-DT WITH DEDEPLICATION

The PSI-DT with Deduplication protocol presented in Section 3.3 is depicted in Figure 9.

B. PROOFS

**Proof of Theorem 6.** We have $C_X = RN$. We also have that $P_X = P^R$, and therefore $R = \frac{\log P_X}{\log P}$, with $P$ the false positive probability for one round. This is given by

$$P = 1 - \left(1 - \frac{1}{N}\right)^{mn} \approx 1 - e^{-mn/N}.$$  

This approximation is good for $mn$ large. We can thus assume (for large $mn$) that

$$C_X = \frac{\log P_X}{\log(1 - e^{-mn/N})} N$$  

Deriving $C_X$ with respect to $N$ gives

$$\frac{\partial C_X}{\partial N} = \log P_X \left[ \frac{mn}{N(1 - e^{mn/N})} \log^2(1 - e^{-mn/N}) \right. + \left. \frac{1}{\log(1 - e^{-mn/N})} \right],$$

which is 0 when $N = \frac{mn}{\log 2}$. For this choice of $N$ we have $P = 1/2$. Substituting in Eq. (3) gives the claimed result.

To complete the proof we need to show that Eq. (3) has a global minimum for $N = \frac{mn}{\log 2}$. To see this it suffices to show that Eq. (3) is convex. Note that the function $f_1 : x \mapsto \log(1 - e^{-x})$ is concave (its second derivative $-\frac{1}{(e^{x} - 1)^2}$ is negative). $f_2 : x \mapsto \log(1 - e^{-x})$ is concave (composition a concave function $(f_1)$ with a concave, non-increasing function $(y \mapsto \frac{1}{y}$ on the negatives, $f_1$ being negative when $x$ is positive)). $f_3 : x \mapsto \frac{1}{\log(1 - e^{-x})}$ is convex (multiplication of a concave function by a negative number). $f_4 : (x, N) \mapsto Nf_3(x/N) = N\log P_X$ is the perspective of $f_3$ and is convex (the perspective of a convex function is convex).

**Proof of Theorem 10.** With $P_X$ the individual false positive probability of the PSI-X invocations, we have

$$P = 1 - (1 - P_X)^n.$$  

For a total false positive probability of $P$, we thus have

$$P_X = 1 - (1 - P)^\frac{1}{n}.$$  

The cost is given by

$$C_P = nC_X(m, n, P_X) + C_{DT}(n) = -\frac{mn^2 \log \left(1 - (1 - P)^\frac{1}{n}\right)}{\log^2 2} + C_{DT}(n)$$

with $C_X$ computed as in Theorem 6. For large values of $n$, we further have

$$\lim_{n \to \infty} -\log \left(1 - (1 - P)^\frac{1}{n}\right) = \log n - \log \log \frac{1}{1 - P} + O \left(\frac{1}{n}\right) \approx \log \frac{n}{\log \frac{1}{1 - P}},$$

which concludes the proof.
### PSI-DT with Deduplication

**on input:** two party protocol PSI-DT(·,·)

---

**Client on input:**

\[ B = \{b_1, \ldots, b_m\} \]

---

**Server on input:**

\[ A = \{a_1, \ldots, a_n\}, \]
\[ DB = \{(a_1, d_1), \ldots, (a_n, d_n)\} \]

---

**Tag Generation**

\[ DB^A = \{\}; DB^T = \{\} \]

**for** \( j = 1, \ldots, n \):  

- if \( \not\exists (t, d_j) \in DB^T \)  
  - \( t \leftarrow \{0, 1\}^\ell \)
  - add \((t, d_j)\) to \(DB^T\)
  - add \((a_j, t)\) to \(DB^A\)

**First instance of PSI-DT**

\[ DT^A = \{(a, t) \in DB^A \mid a \in B\}\]
\[ T' = \{t \mid \exists (a, t) \in DT^A\}\]

---

**Second instance of PSI-DT**

\[ DT^T = \{(t, d) \in DB^T \mid \exists (a, t) \in DT^A\}\]
\[ DT = \{(a, d) \mid \exists t \in T : (a, t) \in DT^A \land (t, d) \in DT^T\}\]

**return** \( (n, DT) \)

**return** \( m \)

---

**Figure 9:** PSI-DT with Data Deduplication.