Fuzzy PSI for Cybersecurity

Peter Grochal[1], Martin Stanek[1]

[1]Department of Computer Science, Faculty of Mathematics, Physics and Informatics, Comenius University

Abstract

Private Set Intersection (PSI) and its fuzzy variant (FPSI) are two-party cryptographic protocols with diverse practical applications. The participants compute the intersection of their private sets, while revealing no information besides the intersection. Recent proposals focus on optimizing the communication and computational complexity of these protocols. For FPSI, the notion of "equality" is typically defined in terms of a small Hamming distance or some p-norm distance. We focus on the application of FPSI in cybersecurity and discuss metrics that might be useful in this context for various datasets, such as passwords, IP addresses, domain names, and so forth. We analyze how these metrics can be used or modified to fit into existing FPSI protocols, and we evaluate a key property of efficient FPSI protocols, the overlap of data points, for real-world datasets.

Keywords

Private Set Intersection, Fuzzy PSI, Indicators of Compromise, Passwords

1. Introduction

Various forms of secure multiparty computation protocols are increasingly used in practice. Private set intersection (PSI) is a theoretically and practically important two-party protocol, where the participants compute an intersection of their private sets without revealing any information about elements outside of the intersection (except the size of their sets). One participant, the receiver, usually learns the intersection, while the second participant, called the sender, learns nothing in the PSI. However, there are various variants, such as one where both parties learn the outcome, or where the result is a set of labels attached to the elements in the intersection, etc. Constructions of PSI need to reflect an adversary model – semi-honest or malicious. For efficiency, it is also important to consider the relative and absolute sizes of the input private sets; some PSI protocols are more suited for balanced scenarios, while others are designed for situations where one set is significantly smaller than the other. For more details, see, for example, the survey [1].

A naive recipe for constructing PSI is straightforward: use a secure two-party equality test for each pair of elements. However, a quadratic number of comparisons is prohibitive for applications with large sets. There are many proposals for various variants of PSI that address computational and communication complexity, employing various cryptographic assumptions and primitives, such as homomorphic encryption, oblivious transfer, Bloom filters, Cuckoo hashing, oblivious pseudo-random functions, vector oblivious linear evaluation, oblivious key-value stores, etc.

Fuzzy PSI (FPSI) is a natural extension of PSI, where the intersection consists of those elements that are "close enough". Fuzzy testing can capture the similarity of attributes more effectively than a strict equality, e.g., geospatial proximity, approximate string matching for genomic data, or similarity of facial data. FPSI was introduced in [2]. The ideas that led to very efficient PSI protocols are generally not directly transferable to FPSI. New constructions are proposed to overcome the complexity of the naive approach (in this case, comparing each pair of elements using a secure two-party fuzzy similarity test). In FPSI protocols, the similarity is usually defined in terms of a small Hamming distance or some p-norm distance [3, 4, 5, 6].

Most linear or almost linear FPSI proposals require some form of disjointness assumption on elements in one set [7, 8]. Such an assumption ensures that the elements are not too close to one another in a

ITAT'26: Information Technologies – Applications and Theory, September 25–29, 2026, Biele Karpaty, Slovakia

peter.grochal@fmph.uniba.sk (P. Grochal); martin.stanek@fmph.uniba.sk (M. Stanek)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-09

CEUR-WS.org/Vol-4323/paper37.pdf

fuzzy sense, or at least that only a small number of elements are mutually close. A question whether real-world datasets and natural fuzzy distances have this property was studied in [8].

Cybersecurity is an area where PSI and FPSI protocols are useful. Researching or responding to security incidents brings simultaneous requirements for (1) data confidentiality, and (2) the need for cooperation in sharing IoC (Indicators of Compromise) and other data with peers.

Our contribution. We extend the discussion in [8] with datasets relevant to cybersecurity – datasets containing passwords, or various indicators of compromise (IoC). We propose natural fuzzy metrics for particular datasets, such as passwords, IP addresses, domain names, etc. When useful, the datasets are enriched with new attributes to better mirror real-world analyst workflows. We analyze how these metrics can be used or modified to fit into existing FPSI protocols, i.e., expressed in terms of Hamming or p-norm distances. Finally, we evaluate data point overlap across real-world datasets, quantifying a key property that directly dictates the performance constraints and computational complexity of existing efficient FPSI constructions.

Remark. We provide examples of potentially useful metrics. Specific applications will require modifications to these metrics to reduce the number of false positives or false negatives in the resulting intersection.

2. Preliminaries and datasets

2.1. FPSI and Labeled-FPSI

In this work, we use both standard FPSI and its variant, Labeled-FPSI (LFPSI). Informally, FPSI allows two parties to privately compute the intersection of their sets under a relaxed notion of equality. Labeled-FPSI extends this idea by returning labels of the items in the intersection instead of the items. This allows one to use more information in the protocol than one would want to share with the other party. More information might be used for testing close elements, while ensuring that the other party learns only what it needs (the label). The formal definition of the functionality provided by the protocols FPSI and LFPSI follows.

Parameters: Universe $\mathbb{U}$, Sender S, Receiver R; Set size n, m; Dimension d. Distance function $dist(\cdot,\cdot)$, Distance threshold δ; Leakage function leakage$(\cdot,\cdot)$; Label length σ.
Functionality:
- Get an input $P_S \in \mathbb{U}^{d\times n}$ from S.
  For LFPSI, wait another input $\text{Label}_S \in \{0,1\}^{\sigma\times n}$ from S.
- Get an input $P_R \in \mathbb{U}^{d\times m}$ from R.
- Return leakage(P_S, P_R) to R.

Leakage Functions: leakage(Q, W) is defined as:
- **FPSI:** $\text{leakage}(Q,W) = \{q_j \mid \exists i \in [n],\ dist(q_j, w_i) \le \delta\}$.
- **LFPSI:** $\text{leakage}(Q,W) = \{\text{label}_j \mid \exists i \in [n],\ dist(q_j, w_i) \le \delta\}$, where label_j is the label associated with q_j.

Figure 1: Definition of FPSI and Labeled-FPSI (adapted from [7])

In the original FPSI setting, efficiency is achieved only under strong structural conditions such as disjoint balls, disjoint projections, or related sparsity assumptions. Later, the density-based extension of Piske et al. [9] replaces these requirements with a bound on local density, namely the number of points per grid cell (C). Their approach consequently requires running C instances of the original protocols. More recently, the formulation of C-weak disjointness [8] weakened the stronger assumptions used in earlier FPSI work, resulting in FPSI protocols with complexity directly determined by the parameter C – the additional cost induced by density.

Weak disjointness.

For intuition, we first describe the setting in terms of *balls*: for each element x held by P_{balls} (e.g., Sender), we associate a ball of radius δ centered at x, under distance $dist(\cdot,\cdot)$. The other party, P_{points} (e.g., Receiver), holds points, and the task is to determine which points lie inside which balls. This is simply another way of viewing proximity under the distance $dist(\cdot,\cdot)$ and threshold δ: a point y lies in the ball centered at x if and only if $d(x,y) \leq \delta$. Throughout the paper, we use this geometric language for intuition (and because it is used in [8]), but we also use the classical formal definition of disjointness.

Definition 1 (Weak disjointness [8]). We say that a set of balls is *C-weakly disjoint* if each element in the set overlaps with at most C other balls.

When each element participates in at most C overlaps, C captures the maximum overlap degree, or informally, the density of the set. This notion relaxes the stronger disjointness assumptions used in prior work, as the authors of [8] transform a generic FPSI protocol with strong disjointness assumption with complexity $O(2^d \cdot \#\text{balls} + \#\text{points})$, to a protocol with the relaxed assumption with complexity increase depending on C: $O(2^d \cdot \#\text{balls} + C \cdot \#\text{points})$.

In this work, we focus on the parameter C in real datasets and compute it experimentally to assess how realistic the weak-disjointness assumption is in practice. This allows us to test whether the constraints on C required by the protocol are compatible with real-world data.

2.2. Distances

Data points (elements) are represented as multi-dimensional vectors of objects from universe $\mathbb{U}$ (integers, real numbers, strings, etc.). Let $x \in \mathbb{U}^d$ and $y \in \mathbb{U}^d$ be data points. A Hamming distance is the number of dimensions in which two vectors differ: $dist_H(x,y) = \sum_{i=1}^{d}(x_i \neq y_i)$. For real-valued vectors x, y, and real $p \geq 1$, the p-norm distance (or L_p distance) is defined as $dist_p(x,y) = \left(\sum_{i=1}^{d} |x_i - y_i|\right)^{1/p}$. For $p = \infty$ we have $dist_\infty(x,y) = \max_{i\in\{1,\ldots,d\}} |x_i - y_i|$.

FPSI protocols are designed for Hamming or L_p distances. Therefore, we sometimes rely on little "tricks" to embed our ideal of what vectors should be considered close.

Increasing the weight of selected attributes. To increase the weight of an attribute (i.e., a vector coordinate) in the overall Hamming distance, we may replicate that attribute multiple times in the vector representation. Consequently, equality on the attribute contributes 0 to the overall distance, whereas inequality contributes in proportion to its multiplicity. With an appropriate choice of threshold, this mechanism can be used to enforce equality on other attributes for vectors that are sufficiently close. A concrete application of this approach to URLs is presented in Section 3.5.

Processing of nominal data. Some proximity definitions can benefit from nominal attributes, e.g., registrar for a domain name. Since FPSI protocols rely on Hamming distance or L_p distance, we need to adjust our data. The case of Hamming distance is easy: since any inequality produces 1 and equality produces 0, no special adjustment is needed. However, we need L_1 distance for domain name vectors, see Section 3.3. In this case, a straightforward integer encoding of (string) nominal data influences the resulting norm unpredictably. We propose to use an n-bit hash function $H : \{0,1\}^* \to \{0,\ldots,2^n-1\}$, and replace the attribute with its hash value. Starting with a set of M distinct nominal elements, a *near match* occurs when two elements with hash values (integers) differ by less than τ. Using the random oracle assumption for H, the probability of a near match can be expressed in the following way [10]:

$$1 - \frac{(M-1)!}{2^{n(M-1)}}\binom{2^n - M(\tau-1) - 1}{M-1}.$$

More concretely, for up to 10000 registrars, the probability of near match with $\tau = 2^{20}$ is approximately 2^{-81} for a 128-bit hash function, and approximately 2^{-209} for a 256-bit hash function. The hash function can be instantiated with a suitable SHAKE instance [11] in practice.

2.3. Datasets and enrichment sources

The focus of the paper is on realistic, real-world data. We use multiple data sources for some data to evaluate how different focus of a particular dataset impacts FPSI. The following datasets are used in our experiments:

- Leaked password lists from SecLists [12] (e.g., Rockyou, Ashley Madison).
- IoCs from the MISP [13] feed of Luxembourg's national CERT/CSIRT (CIRCL[1]), downloaded from the repository [14].
- Malicious IP addresses from AbuseIPDB[2], downloaded from the repository [15].
- Malicious IP addresses aggregated by IPsum threat intelligence feed [16].
- Aggregated list of malicious domains [17], we call this RM dataset.
- Malicious URLs from URLhaus [18].
- Known Command-and-Control (C2) URLs from ViriBack [19].

Some datasets are enriched to provide additional context that allows for the definition of meaningful record proximity. For IP address enrichment, we combine registry-based and provider-published sources. Regional Internet Registries (RIRs) expose registration data through WHOIS and RDAP protocols. Since these protocols are rate-limited, the combined delegated statistics file [20] provided by the NRO (Number Resource Organization) is more suitable for bulk processing. This source aggregates the daily delegation ledgers of all five RIRs and IANA in a standard exchange format and enables local querying without rate limits. Additionally, to test if an IP is a CDN or web-hosting provider, we use the CDN IP Registry [21]. It contains a database of provider-published IPv4/IPv6 ranges and ASN allocations for CDN operators such as Cloudflare, Akamai, and Fastly.

3. Cybersecurity data

3.1. Passwords

A common application is finding a password in a dataset of leaked passwords. Popular site *Have I Been Pwned*[3] implements this service. A more secure way is to perform a private membership test, when the dataset owner does not learn the tested password. We can generalize this scenario further – consider two parties, R and S, each having their own set of passwords P_R and P_S, respectively. Using PSI protocol R learns $P_R \cap P_S$, while S learns nothing. However, the exact match is often not sufficient for a password test; close enough passwords should be considered equal as well, leading us to FPSI.

For the closeness of passwords, we propose to use some set of common password modification rules, often applied in password-cracking attacks. This approach is more realistic than using character-based Hamming distance up to some threshold value [8], where simple and natural modifications like prepending or deleting the first character make the password and its modification distant.

We use Hashcat's `best66.rule` file[4] (previously known as best64.rule), containing 66 rules, and create potentially 66 variants of a password (including the original one).

Remark. An interesting observation can be made by calculating the percentage of unique passwords obtained by using `best66.rule` with respect to the theoretical maximum of 66 times the dataset size. We used Rockyou dataset, and the results were 78.11% for the first $n = 10000$ passwords, and 66.71% for the first $n = 100000$ passwords.

Let R be a set of rules. For passwords u and v, $u \to_R v$ denotes that v can be obtained from u by some rule from R. Since the rules are directional, we have two natural possibilities for how to define when two passwords are close.

[1] https://www.circl.lu/
[2] https://www.abuseipdb.com/
[3] https://haveibeenpwned.com/
[4] https://github.com/hashcat/hashcat/blob/master/rules/best66.rule

Definition 2. Let R be a set of rules. Passwords u, v are

- *asymmetrically close*, if $u \to_R v$ (this relation is not symmetric in general);
- *close with common root*, if $\exists w : \ w \to_R v \ \wedge \ w \to_R u$

In the FPSI protocol, these definitions reflect different use cases. For asymmetrically close passwords, receiver R asks what passwords in P_R can be derived from the sender's P_S ("Do I have a password derivable from your leak?"). For passwords that share a common root, we are asking whether there is a potential attacker that could derive both passwords from some common password ("Can our common attacker succeed?").

3.1.1. Converting FPSI to PSI

FPSI with asymmetric closeness can be converted to simple PSI:

1. S updates their set to $P'_S = \{v \mid \exists u \in P_S : u \to_R v\}$
2. R and S run PSI with P_R and P'_S

The size of P'_S can be upper-bounded $|P'_S| \leq |R| \cdot |P_S|$, although our experiment with the Rockyou dataset and best66.rule suggests $|P'_S|$ will be substantially smaller. Still, this FPSI to PSI conversion is practically efficient only for a small set of rules.

Common root closeness is hard – we don't care what the common root password is, we just want to test the existence of such a password. The idea is to transform both P_R and P_S (using inversion operations) to obtain *root* candidates and then run PSI on these sets. The inversion is easy for some operations, for example:

- `:` (identity) – inversion is the identity itself
- `^c` (prepend character c) – inversion is to delete c, if the password starts with c (otherwise, there is no root candidate)
- `$c` (append character c) – inversion is to delete c, if the password ends with c

However, inverting some other operations leads to a large number of candidates. For example, `]` and `DN` operations delete the last and Nth character, respectively. An inversion is the set of all possible passwords with an inserted character at the given position. The size of the set, therefore, depends on the alphabet size. Moreover, a rule that is a sequence of multiple operations is inverted as a reversed sequence of inversions. This might cause a huge size explosion, especially for rules invoking multiple deletions.

We conclude that a better way for common root closeness FPSI is to use a custom Boolean circuit, a secure two-party sub-protocol for testing this relation, and evaluate each pair of passwords using this sub-protocol.

3.2. IP addresses

IP addresses are common IoC artifacts in threat intelligence and malware analysis. However, raw IPs lack context for effective attribution and clustering. IP enrichment addresses this by augmenting IP addresses with additional features. While our implementation relies on freely available databases like RIR (see Section 2.3), applications may employ other commercial intelligence feeds. Let us examine specific registry fields and their utility for analyzing the closeness of IP addresses.

- `netblock` – Part of what netblock the IP address is (e.g., `158.195.0.0 - 158.195.255.255`).
- `allocated` – Registration date of allocation (e.g., `2001-09-26T10:21:39Z`).
- `opaque-id` – The owner's unique identifier can be used as a substitute for the owner's identity, which is not easily obtainable.

- `country` – Two-letter country code corresponding to the legal or administrative jurisdiction. It does not require every IP address in the block to be physically located in, or to route traffic within that specific country.
- `registry` – Which registry the IP is registered in (APNIC, ARIN, RIPE, LACNIC, AFRINIC).

Additionally, the following binary flag is useful when reasoning about IP addresses:

- `isCDN` – Indicate whether the IP address is used as CDN, web-hosting, or other hosting service. Unlike the information listed above, this information is not stored by RIRs, but fortunately for us, there are free, open-source projects mapping CDN ranges [21].

3.2.1. When are two IP addresses close

In our experiments, we use the fields `ip`, `isCDN`, and `netblock` or `opaque-id` to define close IP addresses.

Definition 3. We consider two IP addresses *close*, if and only if (1) they are the same, or (2) they share the same `netblock` and neither of them is part of a CDN or hosting service.

This means that for IP addresses falling within declared hosting or CDN ranges, exact IP equality is required. This distinction reflects the observation that a compromised host or malicious service rarely implies compromise of other IP addresses within the same provider range, since the underlying issue is often specific to the hosted software rather than the infrastructure itself. More formally, the distance of IP addresses can be expressed using vectors and their Hamming distance.

Definition 4. For an IP address x, with `netblock` n, we define its corresponding vector as vector $v_x = (x, n)$ if x is not part of a CDN or hosting service, and $v_x = (x, x)$ otherwise. The distance of two IP addresses x and y is defined as a Hamming distance of their vectors $dist_H(v_x, v_y)$.

It can be easily seen that two IP addresses are close, if and only if their distance is at most 1. Transformation of IP addresses into vectors would be a preprocessing step in an FPSI protocol. The FPSI with Hamming distance can then be executed on the corresponding vectors. The resulting intersection will reflect Definition 3.

An alternative is to replace `netblock` with `opaque-id`, and this choice depends on the level of false positives we are willing to accept. It is particularly useful for narrow netblocks owned by the same entity. The `country` attribute may also provide limited additional value; close IPs can be constrained to share the same country code by extending the feature vector with `country` encoded as an integer and still requiring Hamming distance at most 1. By contrast, other attributes are either too coarse (`registry`), or too stale to be reliably informative for current suspicious activity (`allocated`).

Remark. Alternatively, we can run a Labeled-PSI protocol. The first element of the vector from Definition 4 will be used as the label, and the second element as the value. This might be preferable if the available Labeled-PSI protocol is more efficient than the FPSI protocol.

3.3. Domain names

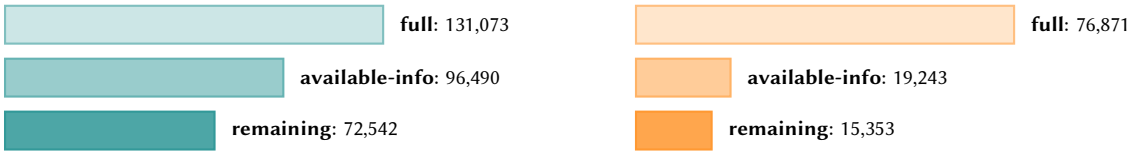
Raw domain data, like IP addresses, does not by itself provide sufficient contextual information. Accordingly, our approach follows the previous section and relies on freely available data sources, primarily RDAP/WHOIS records and DNS data.

A practical limitation is that not all information is available at the level of a full host name. In many cases, relevant registry data can only be obtained for the registrable domain. For example, `subdomain.domain.tld` does not typically provide additional registry-level information beyond what is already available for `domain.tld`. To handle this consistently, we therefore query information for the *registrable domain*. To achieve this, we use the Public Suffix List[5], since the registrable domain is not always the second-level domain (e.g., `*.co.uk`). Let us examine selected fields and their utility for comparative analysis:

[5] https://publicsuffix.org/list/public_suffix_list.dat

- `registered` – The domain registration date (e.g., `2001-09-26T10:21:39Z`), since threat actors often register multiple domains in a short time-frame.
- `registrant` – Usually empty or unavailable, and therefore of limited practical value.
- `registrar` – The organization through which the domain was registered[6] (e.g., `Cloudflare, Inc.`).

An objective limitation is that the information may be unavailable – due to the domain being taken down, or because it is a subdomain of a private domain like `duckdns.org` or `weebly.com`, which allows its users to create subdomains, but does not provide logs of registration dates. A way to get this information (for sites using https) would be through certificate-transparency logs, since the logs contain the not-before part of the oldest certificate for that subdomain. This option is impractical for our experiment, since public service provided by `crt.sh` is slow and rate-limited. We therefore have to remove domains, for which we cannot get WHOIS data, and then domains that were registered as a subdomain registry. The graph below shows the number of entries removed.



(a) RM dataset (b) CIRCL dataset

Figure 2: Shrinking datasets after removing domains without available information, and then subdomain registries.

3.3.1. When are two domain names close

Definition 5. We consider two domain names *close*, if and only if (1) they are the same, or (2) their registrable domain is not a subdomain registry and it was `registered` by the same `registrar` in a short time τ (for example, within a day or a week), or (3) they both resolve to the same IP address.

The rationale is that both conditions are weak but useful indicators of common control. Domains registered by the same registrar within a short time window may reflect coordinated bulk registration, a pattern often seen in malicious campaigns. Likewise, domains resolving to the same IP can suggest reused infrastructure. Excluding subdomain registries reduces false positives.

Access to proprietary intelligence feeds or commercial APIs could allow for getting the registration date for the full domain. As a result, the proposed methodology could be applied to a wider dataset – including subdomain registries, which are favoured by malicious actors, but had to be excluded from our dataset to prevent false positives.

Remark. Threat actors often reuse TLS certificates, use specific TLS configurations (JA3/JA4 fingerprints), and set up their own malicious nameservers. All this information can be used to enrich domain names and improve the definition of close domain names.

Definition 6. For a domain name x, `registered` at date and time t via registrar r, we define its corresponding vector as $v_x = (t, H(r))$, where H is a hash function returning an integer, modelled as a random oracle. The distance of two domain names x and y is defined as an L_1 distance of their vectors $dist_1(v_x, v_y)$.

A natural criterion for temporal proximity is an upper bound on the time difference. Since the Hamming distance does not capture this notion adequately, we instead employ the L_1 distance. However, the equality tests on the `registrar` must still be enforced, which we handle using the proposed hashing-based reduction, see Section 2. We therefore run Labeled-FPSI with L_1 distance on the corresponding vectors with threshold τ, and domain names as labels.

[6]Different registry types (especially ccTLDs) may give different identifiers for the same registrar, whereas gTLDs provide the full registrar name.

The choice of constructing vectors and using L_1 distance, however, does not result in the intended notion of closeness defined in Definition 5. Generally, the complication is condition (3) in Definition 5, where testing a non-empty intersection of resolved IP addresses, and how to transform this predicate into an L_p distance. We conclude that not every generic closeness relation can be efficiently reduced to an L_p or Hamming distance with reasonable overhead.

Remark. Another problem with condition (3) in Definition 5 was that DNS resolving returned an empty set for 53% out of 72,523 remaining domain names in the MR dataset, and 41% out of 15,326 remaining domain names in the CIRCL dataset. We observed many IP addresses that belonged to Cloudflare, and many IP addresses that were known to be malicious.

3.4. Hash values

A common IoC is the cryptographic hash of a malicious file; under the random-oracle assumption, such values are uniformly random and independent. Consequently, this dataset lacks a meaningful distance notion, and therefore, fuzzy matching is not appropriate, and only standard PSI can be used.

3.5. URLs

URLs are inferior as primary IoCs compared to domains or IPs because their path and query components are trivially mutable – attackers change landing pages, obfuscate paths, or rotate URLs on the same infrastructure with minimal operational cost – whereas domain/IP pairs and C2 infrastructure fingerprints persist across campaigns and offer greater detection longevity. Therefore, full URLs are more valuable for immediate triage and takedown actions rather than sustained threat intelligence. Therefore, we propose a closeness useful for threat-hunting with tolerance for false positives, aiming to detect behavior similar to the already-detected attacker's activity.

URLs contain a lot of information already, and we therefore use various parts of the URL for the definition of closeness, without additional enrichment.

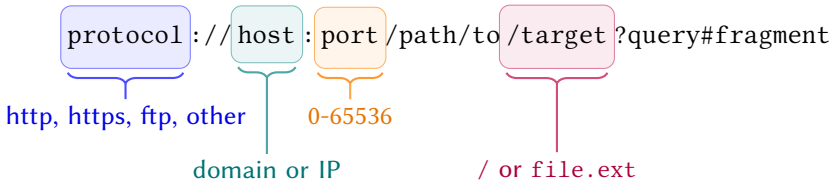


Figure 3: Parts of URL

The `type` of URL `target`, is either none or the type of file extension, one of a fixed set: office-document, image, audio-video, archive, generic-data, web-script, executable, shared library, system-file and other. We chose these because they imply a different tactic of the threat actor.

3.5.1. When we consider two URLs close

Definition 7. We consider two URLs *close*, if and only if they refer to the same `host`, or they share the `protocol`, `port`, and `type`.

Shared hostname/IP address suggests potential common infrastructure or administrative control, indicating that it is managed by the same actor. In contrast, matching attack characteristics like the combination of the port and type of payload, captures the similarity in technical implementation, which often reflects reused malicious infrastructure, automated deployment, or standardized attack frameworks, even when the hosting domains differ.

Definition 8. For a URL x which connects to `host` h over `protocol` p and `port` number n with target `type` t, we define its corresponding vector as $v_x = (H(h), H(h), H(h), p, n, t)$. The distance of two URLs x and y is defined as the Hamming distance of their vectors $dist_H(v_x, v_y)$.

It can be seen that two URLs are close, if and only if their distance is at most 3. The Labeled-FPSI with Hamming distance can then be executed on the corresponding vectors to URLs precomputed in the preprocessing step, and URLs as labels. The result will reflect Definition 7.

Remark. Hashing is necessary because we require fields with a bounded size (not an arbitrary-length string like hostname/domain).

The closeness from Definition 7 may be influenced by the dataset or by the specific application in which FPSI is intended to be used. The datasets analyzed in our study exhibited the following characteristics:

- URLhaus dataset contained 6,234 occurrences (8.04%) of `raw.githubusercontent.com`, followed by 2,841 (2.60%) and 1967 (2.54%) occurrences of two known malicious domains, and then 1,690 occurrences (2.18%) of `github.com`.
- CIRCL MISP dataset contained 2,017 occurrences (13.05%) of `drive.google.com` followed by 94 occurrences (0.61%) of `github.com`.
- Viriback contained no such amounts legitimate hosting sites, most likely because they focus on Command-and-Control (C2) infrastructure.

This suggests that the metric may need to be adapted for the peculiarities of well-known storage and content-hosting providers. For example, in the case of GitHub, the first path segment (the repository owner) may need to be treated as part of the host. This would result in at most 72 identical repositories, compared to 2, 841 entries of `raw.githubusercontent.com`. Similar adjustments may be appropriate for other providers, such as `drive.google.com`, which may appear in other datasets.

3.6. File paths

File paths are not frequently relied upon as primary IoCs, because they are trivially mutable – attackers can rename, relocate, or randomize paths with negligible operational cost. Consequently, file path references are most often found in tightly scoped case reports that document a single incident's environment rather than in generalized IoC feeds.

We use the following information: `filename`, `extension`, type of the file will be similar to the type of URL, one of none, office-document, image, audio-video, archive, generic-data, web-script, executable, shared library, system-file and other.

3.6.1. When we consider two file paths close

Definition 9. We consider two file paths *close*, if and only if (1) they share the `extension` and either the `filename` or the beginning of the path (of length 2), or (2) they share the extension `type` and both the `filename` and the beginning of the path.

The rationale is that the type of the file, its extension, is often related to the "procedure" part of TTP (especially on Windows). In a practical sense, it determines whether the attacker exploits a specific audio-video software, launches an executable file, performs DLL injection, etc. This technique is harder to swap than simply renaming the file or changing its location on the file system.

Definition 10. For a file path x with `extension` e, `type` t, and `filename` f with its first two directories d. We define its corresponding vector as vector $v_x = (H(f), H(e), t, H(p))$. The distance of two URLs x and y is defined as the Hamming distance of their vectors $dist_H(v_x, v_y)$.

Since having the same `extension` implies having the same `type`, we can see that two file paths are close, if and only if their distance is at most 1. The result of Labeled-FPSI with Hamming distance, executed on corresponding vectors of the file paths, will reflect Definition 9.

Remark. An interesting note, in the CIRCL dataset, the second most frequent extension type was `other`, with around 500 unique custom extensions. The most frequent type was `executable`, out of which 47.87% of records (1912) were `.pyo`, a file format used before Python 3.5 for `.pyc` files that were created

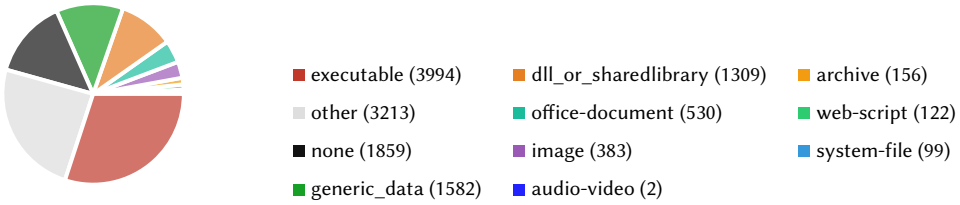


Figure 4: Dataset extension types

with optimizations. We point this out to illustrate that it is not simple to make special cases for known extensions that we expect to see in bulk. Rather unexpected extensions may appear during a campaign, for example, because a certain vulnerability is actively exploited. Therefore, it is very likely that at some time or another, the number of close file paths explodes to numbers comparable to our dataset.

4. Experiments

Based on measures of closeness defined in the previous sections, we ran a simple experiment on a number of different datasets. Each experiment counts the overlap, i.e., how many *close* elements are there for each element, and then presents the minimum, average, and maximum number of close elements. The minimum was 0 for all experiments and datasets – expected and confirmed value. Throughout this section, we denote the size of a database by n.

Passwords

The datasets used in this evaluation consist of leaked password collections obtained from SecLists [12], since we aim to simulate a realistic setting where two parties check if their passwords are close. For this purpose, we mainly used full leaks, except for the Rockyou dataset, which we limited to the first 1 million entries. The Ashley Madison dataset was also included to allow direct comparison with the paper [8]. We evaluate asymmetrically close passwords, see Definition 2. Table 1 summarizes the corresponding results.

Since the applied rule set is `best66.rule`, there is a strict upper bound on the number of close passwords determined by the number of rules. However, this bound was not reached for any of the datasets. Compared with the study [8], the expected overlap is smaller, which is a positive result for the construction of FPSI. At the same time, the degree of overlap is strongly influenced by both the size of the rule set and the size of the dataset. For comparison, in [8] a Hamming distance of at most 3 implies that from a password of length k over an alphabet of size a, one can generate approximately $a^{O(k^3)}$ close passwords.

Table 1
Number of *close* passwords to a password for each dataset

dataset	n	MIN	AVG	MAX
fortinet-2021	78044	0	0.448	22
phpbb	184384	0	1.464	33
000webhost	720300	0	0.5059	29
Rockyou	1000000	0	5.5001	53
Ashley Madison	375824	0	2.555	35

The results in Table 1 suggest that larger datasets tend to produce greater overlap (note that the Ashley Madison dataset, unlike other datasets, contains only passwords whose MD5 hash was cracked). To verify this observation, we conducted an additional experiment using different prefixes of the Rockyou

dataset. The results are presented in Table 2. In general, both the average and the maximum overlap increase as n grows, up to a certain threshold. Although this is not a surprising result, it is an important factor to take under consideration when constructing FPSI protocols.

Table 2
Changing statistics depending on n (Rockyou)

n	MIN	AVG	MAX
1000	0	0.178	9
3000	0	0.081	10
5000	0	0.050	10
10000	0	1.337	25
25000	0	1.789	29
50000	0	2.232	35
100000	0	2.864	41
1000000	0	5.5001	53
10000000	0	4.8783	58

IP addresses

Various datasets were analyzed using Definition 3, and its variant where `netblock` was replaced with `opaque-id` attribute. The second variant produced significantly larger overlap. The results are presented in Table 3. In general, a larger dataset implies a greater amount of close IP addresses on average. Even more dense entries were observed in the IPsum dataset.

Table 3
Number of *close* IP addresses to an IP address from dataset, using `netblock` (left) or `opaque-id` (right)

dataset	n	MIN	AVG	MAX
abuse3	55721	0	155.464	828
abuse30	134577	0	176.356	1131
circl	37346	0	58.955	805
IPsum	16487	0	78.500	492

dataset	n	MIN	AVG	MAX
abuse3	55721	0	612.106	2292
abuse30	134577	0	1044.943	5118
CIRCL	37346	0	289.809	2335
IPsum	16487	0	319.869	1295

Domain names

The results for close domain names in Table 4 are based on Definition 5, where we set the threshold $\tau = 1$ day (86400 seconds). Similar to other experiments, the results for CIRCL and RM datasets show that maximum and average values increase with larger dataset.

Table 4
Experimental number of *close* domains to a domain from dataset

dataset	n	MIN	AVG	MAX
CIRCL	15326	0	22.740	422
RM	72523	0	71.026	748

URLs

Table 5 shows the results of our experiment on URL datasets using Definition 7. The overlaps are rather large, mainly due to the broad notion of closeness in the definition. Because more than 10% of

the URLhaus dataset contains `raw.githubusercontent.com`, we ran the experiment for the revised definition too, to reduce false positives by treating the first path segment (the repository owner) as part of the host. While this had little effect on the average, it reduced the worst-case C by about 37%. As noted in Section 3.5, however, extending this treatment to all file-hosting sites would require substantial effort, and we would likely still see spikes of C from time to time due to threat actors using non-standard data-hosting.

Table 5
Number of *close* URLs to an URL from dataset (left), and with special treatment for `github` (right)

dataset	n	MIN	AVG	MAX
CIRCL	15453	0	2849.651	5030
Urlhaus	77568	0	3778.696	16374
Viriback	13007	0	3810.610	6456

dataset	n	MIN	AVG	MAX
CIRCL	15453	0	2849.586	5030
Urlhaus	77568	0	3702.024	10237
Viriback	13007	0	3810.610	6456

File paths

As mentioned in Section 3.6, there are unexpected features of a particular dataset that can skew the distribution of close data points. The results for available CIRCL dataset and Definition 9 are presented in Table 6.

Table 6
Experimental number of *close* filepaths within the dataset

dataset	n	MIN	AVG	MAX
CIRCL	13249	0	465.647	1911

We can see that in the experiments shown above, the multiplicative factor C is larger than 2^d (the longest of our vectors was 6). The only exceptions are results for passwords. Hence, FPSI protocols with complexity $O(2^d \cdot \#\text{balls} + C \cdot \#\text{points})$, provided that the number of sender and receiver points is comparable, will have their complexity dominated by the term $C \cdot \#\text{points}$.

5. Conclusion

This work shows that the choice of metric is not merely a technical detail, but a defining factor in how well FPSI can serve cybersecurity use cases. Some metrics are naturally broader and produce larger expected intersection sizes than others. However, this should not be viewed as a weakness of the proposed metrics, even though larger intersections often contain more false positives. In practice, the acceptable level of false positives depends on the task: routine triage demands very few false positives, while active hunting for a specific attacker may justify a much broader net in exchange for a true positive, even at the cost of substantial noise.

Our experiments suggest that FPSI protocols with complexity $O(2^d \cdot \#\text{balls} + C \cdot \#\text{points})$ are particularly promising for passwords, where the bounded definition of closeness makes the approach practical. At the same time, this study highlights the difficulty of defining a useful fixed metric that consistently yields only a small number of close candidates – especially because sometimes what we want may inherently contain any number of close candidates. In adversarial settings, attackers do not behave uniformly; they exploit whatever currently works, whether that is a specific file type, a compromised IP range, or a hosting provider that is slow to respond. As a result, real-world data can produce sharp spikes in one category or another, and consequently trigger a steep blowup in the number of elements close to an element (C), and therefore a blowup in complexity.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] D. Morales, I. Agudo, J. Lopez, Private set intersection: A systematic literature review, Computer Science Review 49 (2023). doi:10.1016/j.cosrev.2023.100567.

[2] M. J. Freedman, K. Nissim, B. Pinkas, Efficient Private Matching and Set Intersection, in: Advances in Cryptology – EUROCRYPT 2004, Springer Berlin Heidelberg, 2004, pp. 1–19.

[3] A. van Baarsen, S. Pu, Fuzzy Private Set Intersection with Large Hyperballs, in: Advances in Cryptology – EUROCRYPT 2024: 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26–30, 2024, Proceedings, Part V, Springer-Verlag, 2024, pp. 340–369. doi:10.1007/978-3-031-58740-5_12.

[4] G. Garimella, M. Rosulek, J. Singh, Structure-Aware Private Set Intersection, with Applications to Fuzzy Matching, Springer-Verlag, Berlin, Heidelberg, 2022, pp. 323–352. doi:10.1007/978-3-031-15802-5_12.

[5] E. Uzun, S. P. Chung, V. Kolesnikov, A. Boldyreva, W. Lee, Fuzzy Labeled Private Set Intersection with Applications to Private Real-Time Biometric Search, in: 30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021, USENIX Association, 2021, pp. 911–928. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/uzun.

[6] A. Chakraborti, G. Fanti, M. K. Reiter, Distance-Aware Private Set Intersection, in: 32nd USENIX Security Symposium (USENIX Security 23), USENIX Association, 2023, pp. 319–336. URL: https://www.usenix.org/conference/usenixsecurity23/presentation/chakraborti-intersection.

[7] Y. Gao, L. Qi, X. Liu, Y. Luo, L. Wang, Efficient Fuzzy Private Set Intersection from Fuzzy Mapping, in: Advances in Cryptology – ASIACRYPT 2024, Springer Nature Singapore, 2025, pp. 36–68.

[8] S. Singh, Y. Jia, A. Kate, Fuzzy Private Set Intersection for Real-World Datasets, Cryptology ePrint Archive, Paper 2026/441, 2026. URL: https://eprint.iacr.org/2026/441.

[9] L. Piske, J. Singh, N. Trieu, V. Kolesnikov, V. Zikas, Distance-Aware OT with Application to Fuzzy PSI, in: Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security, CCS '25, Association for Computing Machinery, New York, NY, USA, 2025, pp. 4679–4691. doi:10.1145/3719027.3744857.

[10] M. Abramson, W. O. J. Moser, More Birthday Surprises, The American Mathematical Monthly 77 (1970) 856–858. doi:10.1080/00029890.1970.11992600.

[11] National Institute of Standards and Technology, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, Technical Report Federal Information Processing Standards Publication (FIPS) 202, 2015. doi:10.6028/NIST.FIPS.202.

[12] D. Miessler, SecLists: The security tester's companion, 2026. URL: https://github.com/danielmiessler/SecLists, accessed: 2026-06-26.

[13] C. Wagner, A. Dulaunoy, G. Wagener, A. Iklody, MISP: The Design and Implementation of a Collaborative Threat Intelligence Sharing Platform, in: Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security, 2016, pp. 105–113. doi:10.1145/2994539.2994542.

[14] A. Dulaunoy, misp-circl-feed: MISP open-source intelligence feed from CIRCL, 2026. URL: https://codeberg.org/adulau/misp-circl-feed, accessed: 2026-06-27.

[15] J. Borestad, blocklist-abuseipdb: Aggregated AbuseIPDB blocklists, https://raw.githubusercontent.com/borestad/blocklist-abuseipdb/refs/heads/main/abuseipdb-s100-30d.ipv4, 2026. Accessed: 2026-06-27.

[16] M. Stampar, IPsum: Daily feed of bad IPs (with blacklist hit scores), https://raw.githubusercontent.com/stamparm/ipsum/refs/heads/master/levels/3.txt, 2026. Accessed: 2026-06-27.

[17] R. Marcoux, malicious-domains: Aggregation of lists of malicious domains (phishing), 2026. URL: https://github.com/romainmarcoux/malicious-domains, accessed: 2026-06-24.
[18] abuse.ch, URLhaus: Malware URL exchange, 2026. URL: https://urlhaus.abuse.ch/, accessed: 2026-06-26.
[19] ViriBack, Malware C2 Tracker, 2026. URL: https://tracker.viriback.com/, accessed: 2026-06-26.
[20] The Number Resource Organization, NRO Extended Allocation and Assignment Reports, 2026. URL: https://ftp.ripe.net/pub/stats/ripencc/nro-stats/latest/nro-delegated-stats, accessed: 2026-06-26.
[21] M. Jabin, cdn-ip-database, 2026. URL: https://github.com/mansourjabin/cdn-ip-database, accessed: 2026-06-26.