On Collaborative Predictive Blacklisting*

Luca Melis, Apostolos Pyrgelis, Emiliano De Cristofaro
University College London

Abstract

Collaborative predictive blacklisting (CPB) allows to forecast future attack sources based on logs and alerts contributed by multiple organizations. Unfortunately, however, research on CPB has only focused on increasing the number of predicted attacks but has not considered the impact on false positives and false negatives. Moreover, sharing alerts is often hindered by confidentiality, trust, and liability issues, which motivates the need for privacy-preserving approaches to the problem. In this paper, we present a measurement study of state-of-the-art CPB techniques, aiming to shed light on the actual impact of collaboration. To this end, we reproduce and measure two systems: a non privacy-friendly one that uses a trusted coordinating party with access to all alerts [14] and a peer-to-peer one using privacy-preserving data sharing [9]. We show that, while collaboration boosts the number of predicted attacks, it also yields high false positives, ultimately leading to poor accuracy. This motivates us to present a hybrid approach, using a semi-trusted central entity, aiming to increase utility from collaboration while, at the same time, limiting information disclosure and false positives. This leads to a better trade-off of true and false positive rates, while at the same time addressing privacy concerns.

1 Introduction

Filtering connections from/to malicious hosts is often used to reduce network attacks and their impact. Due to the impossibility of performing expensive computations in real-time on each connection, filtering is usually done via simple look-ups, using periodically updated lists of suspicious hosts, i.e., *blacklists*. These can be created locally and/or by obtaining the most prolific attack sources from alert repositories such as DShield.org or DeepSight [1].

In [11], Katti et al. study the prevalence of "correlated" attacks, i.e., mounted by the same sources against different networks. They find them to be very common, and highly targeted, suggesting that real-time collaboration between victims could improve malicious IP detection time. Zhang et al. [16] are the first to introduce the concept of *collaborative predictive blacklisting* (CPB): different organizations send their logs to a central authority that, in turn, provides them with customized blacklists based on relevance ranking. In follow-up work, Soldo et al. [14] improve on [16] by replacing ranking with an implicit recommender system. Overall, collaborative approaches to threat mitigation are increasingly advocated, with more and more efforts to promote information sharing, including those proposed by CERT [4], RedSky Alliance [13], Facebook's ThreatExchange [3], or the White House [15].

In this work, we focus on two open problems that remain largely unaddressed w.r.t. the impact of collaboration on (1) false positives/negatives, and (2) privacy. Prior work on CPB [14, 16] only focuses on measuring "hit counts", i.e., the number of true positives, but fails to account for incorrect predictions—i.e., false positive/negatives. Moreover, real-world deployment of collaborative blacklisting is hindered by confidentiality issues, as well as trust, liability, and competitiveness concerns as sharing alerts could harm an organization's reputation or disclose sensitive information about customers and business practices [2]. To the best of our knowledge, the peer-to-peer model proposed by Freudiger et al. [9] is the only privacy-friendly approach to the problem: organizations interact in a pairwise manner, aiming to privately estimate the benefits of collaboration, and then share data with "good" partners. However, as discussed later in this paper, it is not clear how to deploy their decentralized techniques in practice.

First, we reproduce, measure, and compare the centralized (non-private) system by Soldo et al. [14] vs the peer-to-peer privacy-friendly one by Freudiger et al. [9], using alerts obtained from DShield.org, involving 70 organizations which report an average of 4,000 daily events over a 15-day time window. We finding that the former [14] achieves high hit counts (almost doubling correct predictions compared to no collaboration), but its F1 accuracy is ultimately poor (14%) due to high false positives. Whereas, the latter [9] allows for better control over incorrect predictions, thus resulting in a better F1 score overall (29%), but actually only slightly improves the hit counts over no collaboration since its peer-to-peer approach limits the amount of data that gets shared.

Our measurements lead to the intuition that, if one needs to control false positives, a controlled data sharing approach might kill two birds with one stone: (1) help organizations find a better trade-off between prediction improvement and increase in false positives, and (2) do so while actually minimizing exposure of possibly confidential data. Therefore, we introduce and analyze a novel hybrid model, relying on a semi-trusted authority, or STA, which acts as a coordinating entity to facilitate clustering without having access to the raw data. The STA clusters contributors based on the similarity of their logs (without seeing these logs), and helps organizations in the same cluster to share relevant data. Toward this goal, we perform a set of measurements to shed light on (i) how to cluster

*A preliminary version of this paper appears in ACM SIGCOMM's Computer Communication Review. This is the full version.

Contributor ID	Source ID	Source Port	Target Port	Timestamp
...D982918	104.217.230.059	6000	1433	2015-06-06 11:49:32

Table 1: Example of an entry in the DShield logs.

organizations, (ii) what should be shared among them, and (iii) how to measure the effect of collaboration on accuracy.

We experiment with a few clustering algorithms using the number of common attacks as a measure of similarity, which can be computed in a privacy-preserving way, and experiment with privacy-friendly within-clusters sharing strategies, namely, only disclosing the details of common/correlated attacks. Overall, we show that our new hybrid model outperforms [9] in terms of hit counts (4x), while achieving better accuracy than [14] (2x).

2 Preliminaries

2.1 Datasets

We gather a dataset of blacklisted IP addresses from DShield.org, a collaborative firewall log correlation system to which various organizations volunteer daily alerts. Each entry in the logs includes a pseudonymized *Contributor ID* (the target), *source IP* address (the attacker), *source* and *target port* number, and a *timestamp*. An example of an entry log is illustrated in Table 1.

With DShield's permission, we collect logs using a web crawler, from February to September 2015, gathering, on average, 10 million logs from 120,000 organizations every day. We exclude entries for invalid or non-routable IP addresses, and discard port numbers, then, for each IP address, we extract its /24 subnet and use /24 addresses for all experiments, following experimental choices made in prior work [9, 14, 16]. This does not necessarily mean that predictive blacklisting algorithms will blacklist entire /24 subnets, since blacklisting an address does not imply blocking all its traffic, but rather subject it to further scrutiny, e.g., enforcing rate limiting or only allowing outgoing packets. Nonetheless, recall that our main goal here is to compare the impact of different collaboration approaches on prediction.

We select a 15-day period, May 17–31, 2015 and restrict our evaluations to a reasonably-sized sample of regularly contributing organizations. We select the top-100 contributors, based on the number of unique IPs reported, that also report logs every day during the 15 days and notice that most contributors (around 60) submit less than 100K logs, while fewer (around 20) submit between 100K and 500K, and only a few organizations contribute large amounts of logs (above 1M). Then, we pick 70 organizations, for each time window, leaving out the top-10 and the bottom-20 contributors. We do so, like in previous work [9, 14], to minimize bias. More specifically, the top contributors contribute a huge number of IPs (order of magnitudes more than other contributors) which might be irrelevant to most organizations, whereas, the bottom ones only report very few logs, thus adding little or nothing to the collaboration. Our final sample dataset includes 30 million attacks, contributed by 118 different organizations over 15 days, each reporting a daily average of 600 suspicious (unique) IPs and 4,000 attack events. This constitutes our "ground truth": if an IP appears in the blacklist for an organization, it is considered to be malicious for that organization.

We use this dataset both as *training* and *testing* sets – more precisely, we consider a sliding window of 5 days for training and 1 day for testing, as done in previous work [9, 14].

Note that we have also repeated our experiments on two more sets of DShield logs, using another 15-day periods (over Feb-Dec 2015), but have not found any significant difference in the results.

Notation. We use notation $\mathcal{O} = \{O_i\}_{i=1}^{n}$ to denote a group of n organizations, where each O_i holds a dataset D_i of alerts, i.e., suspicious IP addresses along with the related timestamp. We aim to predict IP addresses generating attacks to each O_i in the next day, using, as the training set, both its local dataset D_i, as well the set D_i', with suspicious IP addresses obtained by collaborating with other organizations. As discussed above, we consider $n = 70$ organizations using alerts collected from DShield.

2.2 EWMA Time Series Prediction

We use Exponentially Weighted Moving Average (EWMA) to perform prediction. Given a signal over time $r(t)$, we indicate with $\tilde{r}(t+1)$ the predicted value of $r(t+1)$, given past observations $r(t')$ at time $t' \leq t$. The predicted signal is computed as:

$$\tilde{r}(t+1) = \sum_{t'=1}^{t} \alpha \cdot (1-\alpha)^{t-t'} \cdot r(t') \qquad (1)$$

where $\alpha \in (0,1)$ is a smoothing coefficient, $t' = 1, \ldots, t$ denotes the training window, and $t+1$ is the time slot to be predicted. For small values of α, EWMA aggregates past information uniformly across the training window, while, with a large α, the prediction algorithm focuses more on events taking place in the recent past.

2.3 Metrics

Throughout our evaluations, we use the following metrics to evaluate the performance of the predictions.

True and False Positives. For each time window and for each organization, we count *True Positives* (TP) as well as *False Positives* (FP). A TP occurs when the prediction algorithm includes an IP address in an organization's predictive blacklist that does appear in its testing set, and a FP – when it does not.

False Negatives. For each time window/organization, we generate predictive *whitelists*, i.e., sets of IPs that are not likely to attack an organization the next day, and count a *False Negative* (FN) when a whitelisted IP address instead appears in the testing set.

TP Improvement and FP/FN Increase. We also measure the average improvement/increase in TP, FP, and FN when compared to a baseline local approach, i.e., when no collaboration occurs between organizations and each of them makes its predictions based only on its local dataset. The improvement in TP is calculated as: $TP_{impr} = (TP_c - TP)/TP$ where TP_c is the number of true positives after collaboration

and TP without. Similarly, the increase in FP and FN is denoted, resp., as $FP_{incr} = (FP_c - FP)/FP$ and $FN_{incr} = (FN_c - FN)/FN$.

Precision, Recall and F1-Score. We calculate the True Positive Rate (TPR), aka *recall*, False Positive Rate (FPR), as well as Positive Predictive Value (PPV), aka *precision*, defined as: TPR=TP/(TP+FN), FPR=FP/(FP + TN), PPV=TP/(TP + FP), and derive the F1 measure, i.e.,

$$\text{F1} = 2 \cdot \frac{\text{PPV} \cdot \text{TPR}}{\text{PPV} + \text{TPR}}$$

Remarks on FP: The absence of an IP from our testing set can occur *either* when the IP is not considered suspicious *or* if it does not generate requests. While we cannot actually distinguish between the two cases, in the latter a FP is actually less "severe" than in the former, thus our FP count may be a bit more conservative. However, our main goal is really to measure and compare with each other the impact of *different* collaboration strategies on predictions so we use this method without loss of generality.

3 Evaluating CPB Techniques

3.1 Soldo et al. [14]

We first evaluate Soldo et al [14]'s CPB approach based on implicit recommendation. We do so aiming to: (1) evaluate false positives and false negatives, which were not taken into consideration in [14], and (2) compare against privacy-friendly approaches, presented later. Essentially, Soldo et al.'s work builds on [16], which bases on a relevance ranking scheme similar to PageRank, measuring the correlation of an attacker to a contributor relying on their history as well as the attacker's recent log production patterns. Soldo et al. significantly improve on this, by using an implicit recommendation system to discover similar victims as well as groups of correlated victims and attackers. The presence of attacks performed by the same source around the same time leads to stronger victim similarity, and a neighborhood model (k-NN) is applied to cluster similar victims. Cross Association (CA) co-clustering [5] is then used to discover groups of correlated attackers and victims, and prediction within the cluster is done via the EWMA time series algorithm (TS) to capture attacks' temporal trends. In other words, the prediction score for each organization is a weighted ensemble of three methods (TS, k-NN and CA). We have re-implemented their system in Python, using Chakrabarti's CA implementation [5].

We start by measuring the basic predictor which only relies on a local EWMA time series algorithm (TS), using $\alpha = 0.9$ as it yields the best results, then, apply the co-clustering techniques (TS-CA), and, finally, implement their full scheme by combining k-NN to cluster victims based on their similarity with CA and TS (TS-CA-k-NN). Fig. 1 illustrates the improvement/increase in TP, FP, FN (compared to the TS baseline) as well as TPR, PPV, and F1, with various k values (ranging from 1 to 35) used by the k-NN algorithm to discover similar organizations. Obviously, the k-NN parameter k does not affect TS-CA and TS.

Fig. 1(a) shows that, with TS-CA-k-NN, TP_{impr} increases significantly with k, almost doubling the "hit count" compared to the TS baseline, whereas, TS-CA improves less (0.67). On the other hand, however, there is FP_{incr} too, 5- to 50-fold, as clusters become bigger (Fig. 1(b)), and naturally, this stark increase in FP leads to low precision, as shown in Fig. 1(e). FNs also always increase compared to TS (Fig. 1(c)), specifically, they double with TS-CA and increase between 0.55 and 0.99 (less for larger k values) compared to TS. FN_{incr} also affects TPR (Fig. 1(d)), with an increase between 0.58 and 0.66. The TP_{impr} does not correspond to a comparable increase in TPR, due to the poor FN performance, as shown by the fact that TS-CA-k-NN reaches 0.99 in TP_{impr} but only at most 0.66 TPR compared to 0.59 with the baseline TS. Overall, Soldo et al.'s techniques achieve poor F1 measures, at most 0.16 and 0.14, with TS-CA and TS-CA-k-NN, actually lower than a simple local time-series prediction (0.26).

3.2 Freudiger et al. [9]

Next, we evaluate the privacy-friendly peer-to-peer approach to CPB by Freudiger et al. [9]. Organizations interact pairwise, aiming to privately estimate the benefits of collaboration, and then share data with entities that are likely to yield the most benefits. They also use DShield data and perform prediction using EWMA. They find that: (1) the number of common attacks is the best predictor of benefits, which can be estimated privately, using Private Set Intersection Cardinality (PSI-CA) [6]; and (2) sharing only the intersection of attacks – which can be done privately using Private Set Intersection (PSI) [7] – is almost as beneficial as sharing everything. Their goal is really to assess benefit estimation/sharing strategies, rather than to focus on deployment. They assume a network of 100 organizations, select the "top 50" among all possible 4950 pairs (in terms of estimated benefits), and only experiment on those. Naturally, without a coordinating entity, it is impossible to rank the pairs, so they suggest that one should collaborate with either organizations when estimated benefits are above a threshold, although it is not stated how to set this threshold; or with the top x organizations with the biggest estimated benefits, but do not experiment with or discuss how x impacts overhead or true/false positives. We replicate both approaches: **(A)** with the top 1% to 5% of global pairs, and **(B)** having each organization pick 1 to 35 most similar organizations.

Fig. 2 shows the improvement/increase in TP, FP, FN (compared to a baseline with no sharing) as well as TPR, PPV and F1 with increasing percentage of global pairs (A). We omit plots for approach (B) since they are worse across the board, although we discuss them next. Looking at TP_{impr}, (A) yields 13% increase when 3% of global pairs are selected whereas for (B), i.e. picking local pairs, TP_{impr} increases along with the number of local pairs selected. (A) has a rather small FP_{incr} (13% increase when the 75 top pairs are selected) compared to (B) which is affected by the number of pairs that each organizations picks for collaboration. When an organization collaborates with 5 others a 25% FP_{incr} is observed on average while when it collaborates with 30 others FP_{incr} reaches 80%.

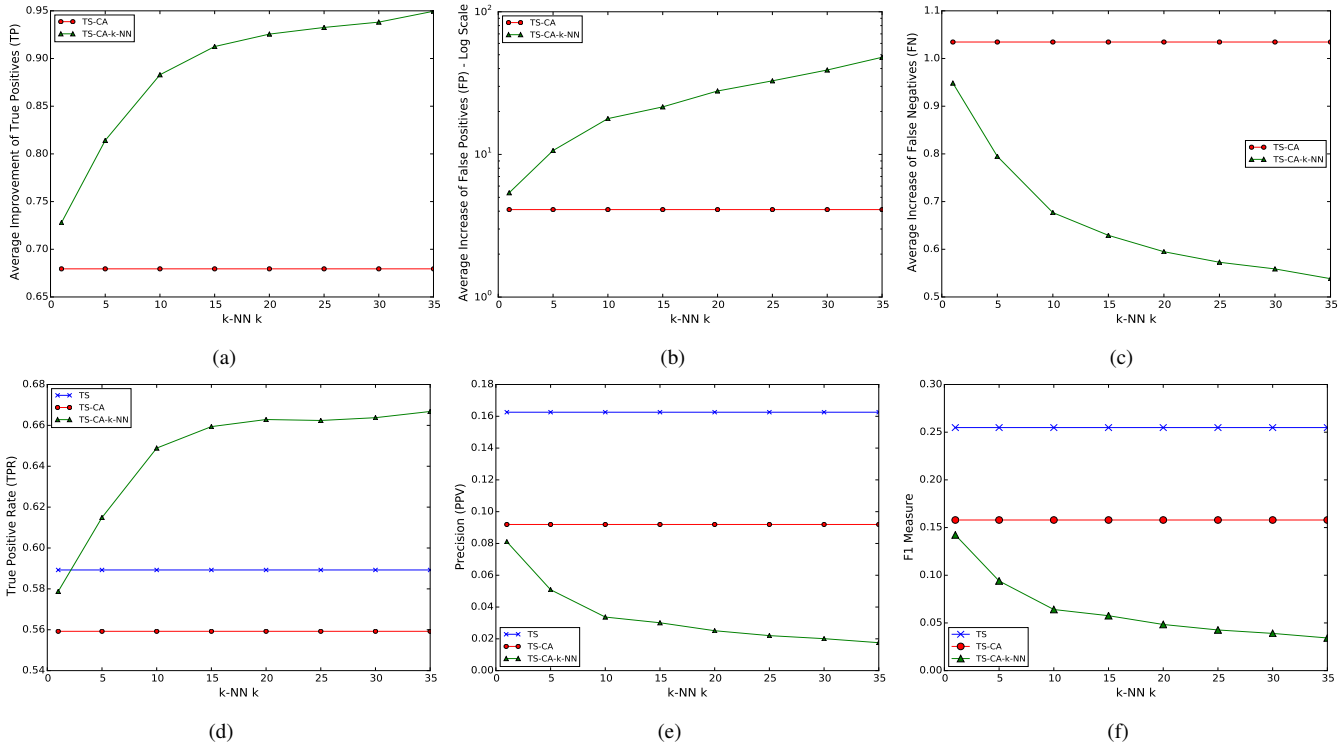


Figure 1: Soldo et al. [14]: (a) TP improvement, (b) FP increase (y-axis in log scale), (c) FN increase, (d) TPR, (e) Precision, (f) F1 measure.

Moreover, we find both approaches achieve a decrease in false negatives with the second approach achieving bigger decreases as the number of collaborators increases.

Overall, both approaches improve precision and recall of the system, yielding higher F1 scores compared to a local approach. Although the increase in TP is not as high as with the non-private approach of [14], a more balanced increase of false positives and a decrease of the false negatives seems possible. However, the system is limited in the amount of new information organizations learn (e.g., only events about IPs they have already seen is shared) as well as scalability, since both the computation of the metrics and the actual data sharing are conducted pair-wise (if there are n collaborating entities, the complexity of the data sharing would be $O(n^2)$).

4 A Novel Hybrid Approach

Centralized state-of-the-art CPB techniques [14] have only focused on improving "hit counts," but, as shown above, they generate very high false positive rates. In practice, organizations might not adopt such solutions if they generate a large number of false alarms. Naturally, one could design better centralized approaches that yield better accuracy, e.g., by learning to discard the data that yield false positives. However, our intuition is that in this case a privacy-preserving approach might be best suited as it can (i) help organizations find a better trade-off between prediction improvement and increase in false positives, and (ii) do so while actually minimizing exposure of possibly confidential data.

Overview. To this end, we introduce a novel hybrid system which relies on a semi-trusted authority, or STA, acting as a coordinating entity to facilitate clustering without having access to the raw data. In other words, the STA clusters contributors based on the similarity of their logs (without accessing these logs), and helps organizations in the same cluster to share relevant logs.

The system involves four steps. (1) First, organizations interact in a pairwise manner to privately compute a similarity measure of their logs, based on the number of common attacks (similar to [9]). Then, (2) the STA collects the similarity measures from each organization and performs clustering using one of three possible algorithms, i.e., Agglomerative Clustering, k-means, or k-NN.[1] Next, (3) the STA reports to each organization the identifiers of other organizations in the same cluster (if any), so that they collaboratively, yet privately, share logs to boost the accuracy of their prediction, by either sharing common attacks (*intersection*), correlated attacks (*IP2IP*), or both. For comparison, we also consider baseline approaches, i.e., sharing nothing (*local*) or sharing everything (*global*). Finally, (4) each organization performs EWMA prediction (again, with $\alpha = 0.9$, as done in our evaluation of [14]). based on their logs, plus those from entities in the same cluster. This approach is *hybrid* in that, while involving a central authority, data sharing is privacy-friendly: in (1) the number of common attacks can be computed using PSI-CA [6], while in (3) sharing of common attacks can occur using PSI [7] and of correlated attacks using [12].

Settings. We once again use datasets and settings from Sec-

[1] Note that, to ease presentation, we do not plot results using Agglomerative Clustering because it yields the worse results.

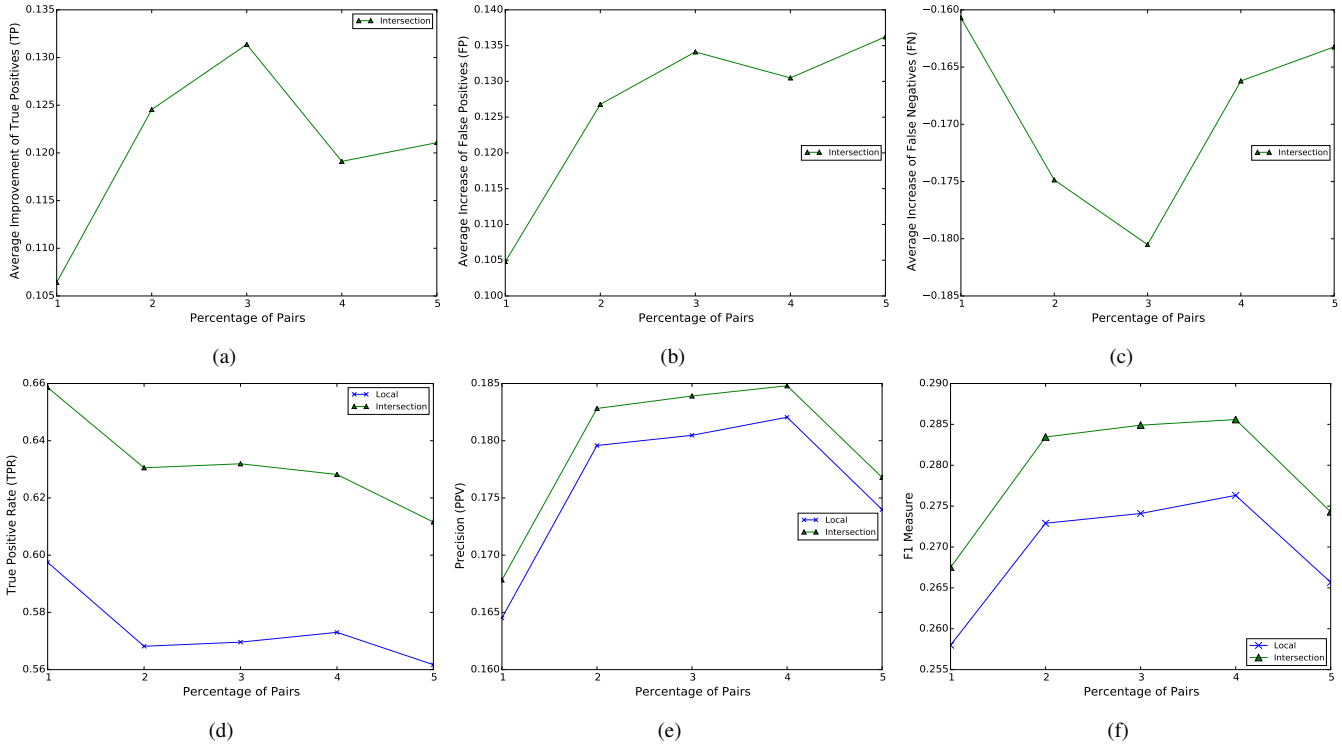


Figure 2: Freudiger et al. [9] (a) TP improvement, (b) FP increase, (c) FN increase, (d) TPR, (e) Precision and (f) F1, with increasing percentage of global pairs.

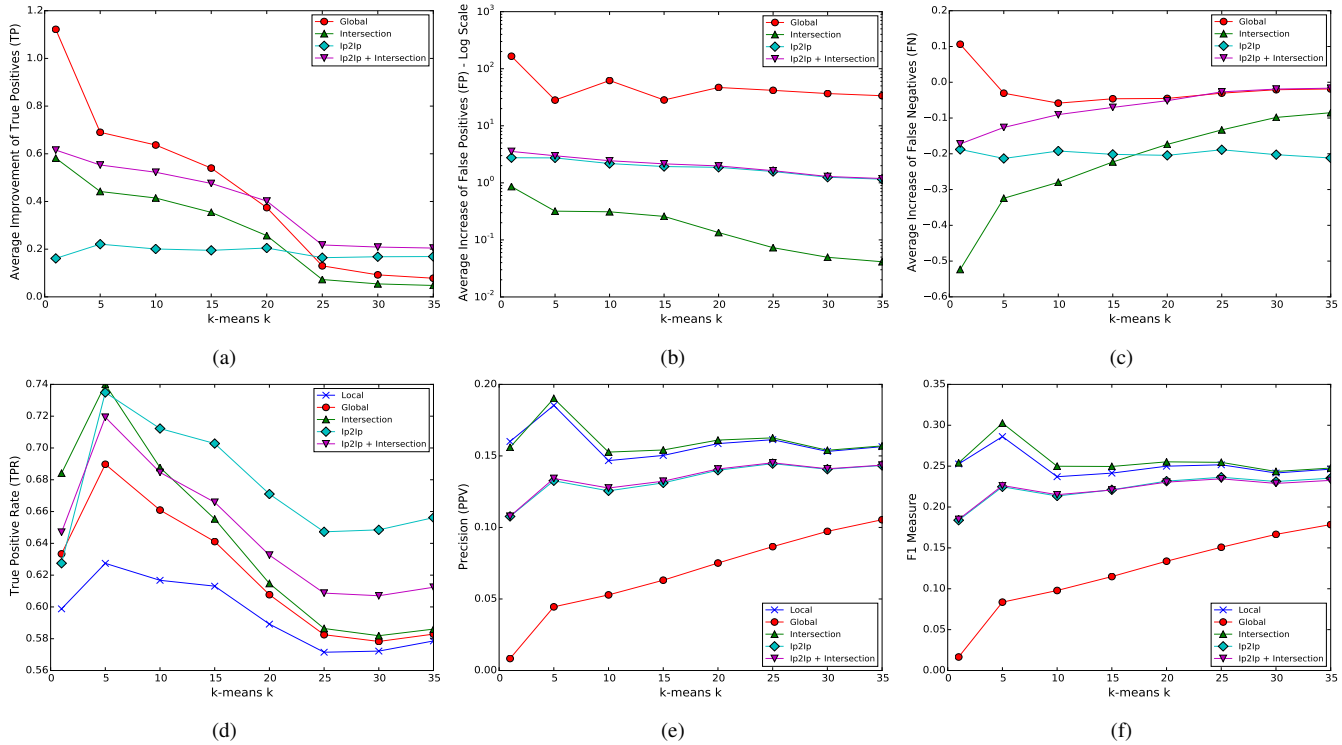


Figure 3: k-means: (a) TP improvement, (b) FP increase (y-axis in log scale), (c) FN increase, (d) TPR, (e) Precision, (f) F1 measure.

tion 2. Also, for the IP2IP method, we only consider the top-1000 attackers (i.e., the top-1000 *heavy hitters*) in each cluster, for each 5-day training-set window, rather than looking for correlations over all the /24 IP space. We fix the k value for the k-NN based recommendation to 50, as it provides the best results in our experiments.

k-means. Next, we use k-means for clustering and decide to restrict to *stronger correlations*, by only taking into account organizations closer to the cluster's centroid, and excluding the rest of them as outliers. We set a distance threshold and choose the value that yields the best result, i.e., the cluster distance value below which 40% of the organizations can be found.

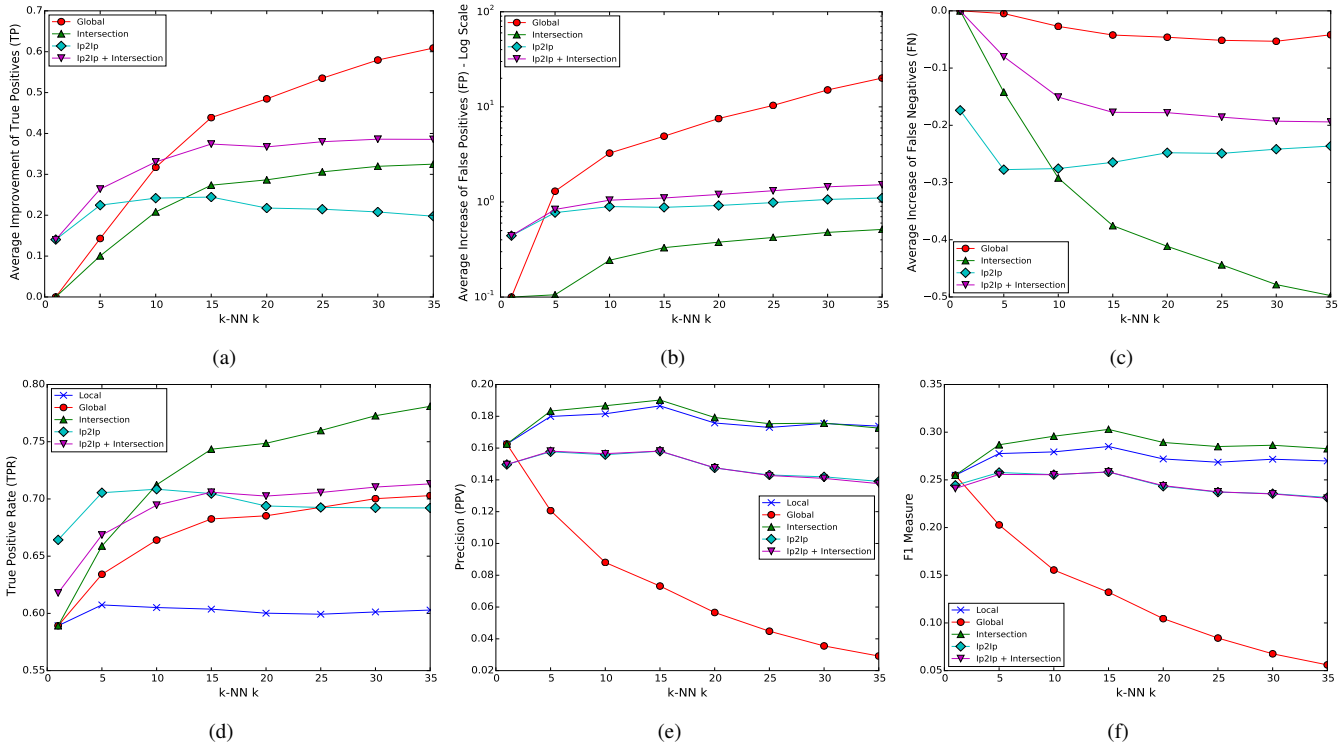


Figure 4: k-NN: (a) TP improvement, (b) FP increase (y-axis in log scale), (c) FN increase, (d) TPR, (e) Precision, (f) F1 measure.

Fig. 3(a)–3(c) plot the average improvement in TP and increase in FP and FN. TP_{impr} is almost constant with IP2IP (0.2) independent of the cluster sizes, while with the other methods it decreases faster due to the distance thresholds, ranging from 1.1 with global for $k = 1$ to 0.1 of intersection for $k = 35$. IP2IP shows steady FN_{incr} values compared to other methods (-0.2, i.e., a 20% decrease) which leads to a better performance in TPR, as shown in Fig. 3(d), for $k \geq 10$ (up to 0.71). Furthermore, intersection yields the best performance in FN_{incr} (-0.52), with $k = 1$. Fig. 3(f) shows the best F1 measure (0.30) is reached with $k = 5$, due to a peak both in PPV and TPR. IP2IP performs slightly worse (0.23) than local (0.28) while poor F1 values for global, with $k = 35$, (0.18) are due to its bad PPV (0.10) – see Fig. 3(e).

k-NN. Recall that k indicates the number of nearest neighbors that each entity considers as its most similar ones. Thus, organizations can end up in more than one neighborhood. Since the algorithm builds a neighborhood for each organization, not all clusters have the same *strength*, so we only consider *strong* clusters in terms of their members similarity and as done with k-means, after tuning the parameters, we set a distance threshold as the 40th percentile to leave possible outliers out of the clusters. From Fig. 4(a), we observe that IP2IP+intersection yields the second best performance in TP_{impr} (0.38, with $k = 35$), while global peaks at 0.60. In terms of FP_{incr}, IP2IP doubles it (for $k = 35$), while intersection achieves the lowest value with 0.51 (again, for $k = 35$). As with previous clustering algorithms, we notice that intersection yields the best decrease in FN, i.e., -0.5 with $k = 35$. Intersection also achieves the highest TPR (up to 0.77) with larger cluster sizes (i.e., for $k \geq 10$), while its combination with the IP2IP reduces it (0.71) – see Fig. 4(d). Fig. 4(e) shows that intersection has the best PPV (0.19 for $k = 15$), similar to local (0.18), while IP2IP performs worse (0.16) due to higher FP_{incr} (almost doubling the FP for $k = 35$). Finally, from Fig. 4(f), note that intersection yields the highest F1 (0.30 for $k = 15$).

Summary of results. We summarize the best results for each clustering algorithm, in terms of best F1, recall, precision, and TP_{impr} in Tables 2–5. We note that intersection is that sharing mechanism that maximizes all metrics, except for TP_{impr}, which is instead maximized with IP2IP+intersection. Both k-means and k-NN peak at 0.30 in F1 including, respectively, 280 and 240 collaborators over all time windows. Agglomerative clustering involves all 700 contributors and achieves $F1 = 0.27$. k-NN with $k = 35$ yields the best results for TPR (0.77), while both k-NN with $k = 35$ and k-means with $k = 5$ achieve 0.19 in PPV. In terms of TP_{impr}, k-means reaches a maximum of 0.61 with $k = 1$ and clusters of size 28 on average, selecting 270 collaborators overall. Slightly lower improvements are achieved with other clustering algorithms, but with more collaborators benefiting from sharing, as well as fewer FP.

Data sharing always helps organizations forecast attacks, compared to performing predictions locally. Predicting based on all data from collaborators yields the highest improvement in TP_{impr} – especially for bigger clusters – but with a dramatic increase in FP_{incr}. When organizations share correlated attacks (IP2IP), we observe a steady TP_{impr}, while sharing common attacks (intersection) outperforms the former when bigger clusters are formed. However, intersection introduces lower FP_{incr}, ultimately leading to better precision and F1 measures. IP2IP+intersection always outperforms the two sep-

Setting		Max F1 [Sharing Intersection]							
Clustering	k	Avg Size	#Coll.	TPR	PPV	TP_{impr}	FP_{incr}	FN_{incr}	$F1$
Agglom.	15	4.6	700	0.72	0.16	0.38 ± 3.51	0.71 ± 4.49	-0.42	0.27
k-means	5	5.8	280	0.73	0.19	0.44 ± 4.21	0.31 ± 1.47	-0.32	0.30
k-NN	15	6	240	0.74	0.19	0.27 ± 0.20	0.33 ± 0.28	-0.37	0.30

Table 2: Best Cases of our Experiments for F1.

Setting		Max TPR [Sharing Intersection]							
Clustering	k	Avg Size	#Coll.	TPR	PPV	TP_{impr}	FP_{incr}	FN_{incr}	$F1$
Agglom.	1	70	700	0.76	0.15	0.50 ± 3.95	1.12 ± 6.98	-0.53	0.25
k-means	5	5.8	280	0.73	0.19	0.44 ± 4.21	0.31 ± 1.47	-0.32	0.30
k-NN	35	14	320	0.77	0.17	0.32 ± 0.21	0.51 ± 0.50	-0.49	0.28

Table 3: Best Cases of our Experiments for TPR.

Setting		Max PPV [Sharing Intersection]							
Clustering	k	Avg Size	#Coll.	TPR	PPV	TP_{impr}	FP_{incr}	FN_{incr}	$F1$
Agglom.	25	2.8	700	0.69	0.16	0.33 ± 3.29	0.47 ± 2.23	-0.35	0.26
k-means	5	5.8	280	0.73	0.19	0.44 ± 4.21	0.31 ± 1.47	-0.32	0.30
k-NN	15	6	240	0.74	0.19	0.27 ± 0.20	0.33 ± 0.28	-0.37	0.30

Table 4: Best Cases of our Experiments for PPV.

Setting		Max TP Improvement [Sharing IP2IP+intersection]							
Clustering	k	Avg Size	#Coll.	TPR	PPV	TP_{impr}	FP_{incr}	FN_{incr}	$F1$
Agglom.	1	70	700	0.67	0.11	0.52 ± 3.95	5.33 ± 16.9	-0.08	0.19
k-means	1	28	270	0.64	0.11	0.61 ± 5.36	3.55 ± 7.17	-0.17	0.18
k-NN	35	14	320	0.71	0.14	0.38 ± 0.25	1.51 ± 1.02	-0.19	0.23

Table 5: Best Cases of our Experiments for TP_{impr}.

arate methods in terms of TP_{impr}, thus, it is the recommended strategy if one only wants to maximize the number of predicted attacks.

Impact of cluster size. With agglomerative clustering, each organization is assigned to exactly one cluster and thus participates in/benefits from collaboration. We observe higher TPR for bigger clusters and, generally, a stable improvement in TP is achieved on average. Similar results are obtained with k-means when all organizations are assigned to clusters. However, when we set a distance threshold, creating more consistent clusters, we observe fluctuations in TPR: as clusters get smaller much faster (in relation to k value), IP2IP starts outperforming intersection. This indicates that correlated attacks can improve knowledge of organizations and enhance their local predictions, especially in smaller clusters. With k-NN, a different behavior is observed: for smaller clusters, IP2IP achieves higher TPR (up to 0.7 for $k = 5$) but, as clusters get bigger, intersection yields the best results (up to 0.77 for $k = 35$). Overall, collaborating in big clusters leads to high TP_{impr} but at the same time it introduces significant FP_{incr}.

Increase/Improvement in TP/FP/FN. We observe that, for all clustering algorithms, maximizing TP_{impr} always leads to higher FP_{incr}, from 1.51 of k-NN up to 5.33 of Agglomerative. The settings that maximize the F1 measure, TPR, and PPV, (when sharing intersection) also minimize FN_{incr}, e.g. agglomerative with $k = 1$ achieves -0.53 FN_{incr}. In general, we observe that (privacy-friendly) collaboration does yield a remarkable increase in TP but also in FP, which results in a limited improvement in F1 score compared to predicting using local logs only.

Overall, our measurements allow us to quantify how different collaboration strategies affect prediction in terms of increasing true positives, false positive, and false negatives, and in general precision, recall, and F1. Ultimately, the main goal is to find settings that improve TP while keeping the increase in FP as low as possible. In this context, the best approach is sharing common and correlated attacks (IP2IP+intersection) with k-NN (see Table 5).

Hybrid approach vs state of the art [9, 14]. When comparing the hybrid approach to Soldo et al. [14], we observe that [14] achieves higher maximum TP_{impr} (0.99 vs 0.61 with k-means, $k = 1$). However, our privacy-preserving techniques outperform [14] in terms of recall (TPR) (e.g., with k-NN we reach 0.77 compared to their 0.66, i.e. up to 18% increase) as well as precision (0.19 with k-means, $k = 5$ vs 0.08, i.e. up to 15% increase) and F1 measure (0.30 with k-NN, $k = 15$ vs 0.14). Finally, comparing the hybrid approach to [9]the former yields better results in terms of TP_{impr} (0.61 for k-means, $k = 1$ vs 0.13 for top 3% of global pairs) and TPR (0.77 for k-NN, $k = 35$ vs 0.66 for top 1% of global pairs), but similar F1 score (0.30 vs 0.28), due to the latter's smaller increase in FP.

Overall, we conclude that a controlled data sharing approach, compared to a centralized one, helps organizations find a better trade-off between prediction improvement and increase in false positives, while minimizing exposure of possibly confidential data.

5 Implementing At Scale

As discussed above, our hybrid system involves four steps: (1) secure computation of pairwise similarity, (2) clustering, (3) secure data sharing within the clusters, and (4) time-series prediction. To assess its scalability, we need to evaluate computation and communication complexities incurred by each step. Naturally, steps (1) and (3) dominate complexities as they require running a number of cryptographic operations (involving public-key crypto) that depends on the number of organizations involved. In fact, clustering incurs a negligible overhead: on commodity hardware, to perform clustering with 1,000 organizations, it takes $6.1ms$ for k-means, $81ms$ for agglomerative and $5.2ms$ for k-NN ($k = 2$).Also, time-series EWMA prediction requires $4.6\mu s$ per IP, so it takes $4.6ms$ for 1,000 IPs. As we compute pairwise similarity based on the amount of common attacks between two organizations, and support its secure computation via PSI-CA [6], step (1) requires a number of protocol runs *quadratic* in the number of organizations. In our experiments, it takes $1.98s$ and 2.12MB bandwidth for one protocol execution, using 2048-bit moduli, with sets of size 4,000 (the average number of attacks observed by each organization). As for (3), i.e., secure within-cluster sharing of events related to common attacks (intersection), we rely on PSI-DT [7], and it takes $1.24s$ and 2.18MB for a single execution with the same settings. Therefore, complexities may become prohibitive when more organizations are involved or more alerts are used.

Aiming to improve scalability, we also implement a variant supporting secure computation of pairwise similarity as well as secure log sharing *without* a quadratic number of public-key operations/quadratic communication overhead. Recall that we rely on a semi-trusted authority, STA, for clustering and coordination, which is assumed to follow protocol specifications and not to collude with other organizations, thus, we can actually use it to also help with secure computations. Inspired by

Algorithm 1 ENCRYPTION [All Organizations]

for each $O_i \in \mathcal{O}$ **do**
  $S_i \leftarrow \emptyset$, $E_i \leftarrow \emptyset$, $K_i \leftarrow \emptyset$
  for each $(d_j, time_j) \in D_i$ **do**
    for $cnt := 1$ **to** $COUNT(d_j)$ **do**
      $S_i \leftarrow S_i \cup \text{PRP}_k(d_j||cnt)$
      $k_j \leftarrow H(d_j||cnt)$
      $E_i \leftarrow E_i \cup \text{Enc}_{k_j}(d_j, time_j)$
      $K_i \leftarrow K_i \cup k_j$
  Send S_i, E_i to STA and store K_i

Algorithm 2 O2O COMPUTATION [STA]

for each $O_i \in \mathcal{O}$ **do**
  for each $O_j \neq O_i$ **do**
    $\text{O2O}[i,j] \leftarrow |S_i \cap S_j|$
    $\text{Buff}[i,j] \leftarrow \{(\ell, E_{j_\ell}), \forall \ell \in \text{INDEX}(S_i \cap S_j)\}$
Perform Clustering on $\text{O2O}[\cdot,\cdot]$
Send relevant $\text{Buff}[\cdot,\cdot]$ entries to organizations in the same cluster

Algorithm 3 LOG SHARING [Organizations in C^*]

for each $O_i \in C^*$ **do**
  $S'_i \leftarrow \emptyset$
  for each $O_j \neq O_i \in C^*$ **do**
    for each $(\ell, E_{j_\ell}) \in \text{Buff}[i,j]$ **do**
      $S'_i = S'_i \cup \text{Dec}_{k_\ell}(E_\ell)$

Kamara et al.'s server-aided PSI [10], we extend our framework by replacing public-key cryptography operations with pseudo-random permutations (PRP), which we instantiate using AES. Specifically, we minimize interactions among pairs of organizations so that the complexity incurred by each of them is constant, while only imposing a minimal, linear communication overhead on STA.

Our extension involves four phases: (i) *setup* where, as in [10], one organization generates a random key κ and sends it to the other organizations, (ii) *encryption* (see Algorithm 1), where each organization O_i evaluates the PRP on each entry d_j in their sets and encrypts the associated timestamp $time_j$, (iii) *O2O computation* (see Algorithm 2), where STA computes the magnitude of common attacks between each pair of organizations in order to perform clustering, and (iv) *log sharing* (see Algorithm 3), where organizations in the same cluster C^* receive information about common attacks (S'_i-s). Note that building the O2O matrix is actually optimized using hash tables (i.e., *dense_hash_set* and *dense_hash_map* from Sparehash. Also, since sets in our system are multi-sets, we concatenate counters to the IP address, so that the STA cannot tell which and how many IPs appear more than once.

Experimental Evaluation. We benchmark the performance of PSI-CA [7] and PSI-DT [7] using 2048-bit moduli, modifying the OpenSSL/GMP-based C implementation in [8], as well as the PRP-based scheme presented above and inspired by Kamara et al.'s work [10]. Experiments are run using two 2.3GHz Intel Core i5 CPUs with 8GB of RAM connected via a 100Mbps Ethernet link. Figures 5(a) and 5(b) plot computation and communication complexities incurred by an individual organization vis-à-vis the total number of organizations involved in the system, while Fig. 5(c) reports the communication overhead introduced on the STA-side for the PRP scheme. As expected, complexities for PSI-CA/PSI-DT protocols on each organization grow linearly in the number of organizations (hence, quadratic overall). For instance, if 1,000 organizations are involved, it would take about 16 minutes per organization, each transmitting 1GB. Whereas, the PRP-based scheme incurs constant complexities on each organization ($57.6ms$ and 120KB) and a low communication overhead on the STA (about 100MB) for 1,000 organizations (Fig. 5(c)).

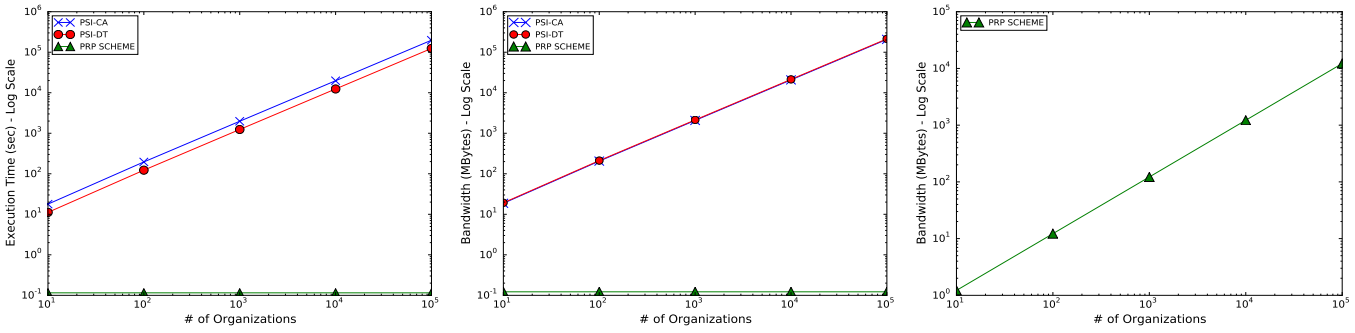
We also evaluate the IP2IP method whereby organizations interact with STA in order to discover cluster-wide correlated attacks. Assuming clusters of 100 organizations and an IP2IP matrix of $(2^{24} \cdot 2^{24})/2$ (recall we consider the whole /24 IP space), we measure a $2.7s$ running time per organization with 41KB of bandwidth as well as a $0.07s$ overhead on the STA with 4.1MB bandwidth. Using the private Count-Min sketch based implementation by Melis et al. [12], we can compress to a logarithmic factor with a small, bounded loss, and the private aggregation is done over 10,336 elements. Even if clusters are bigger than 100, as detailed in [12], one can still perform private aggregation on multiple subgroups (e.g., of size 100) without endangering organizations' privacy.

Security. Protocols do not leak *any* information about the logs of each organization to the STA, with or without using the server-aided variant. Clustering is performed over similarity measures computed obliviously to STA, and so does within-cluster data sharing. Privacy-preserving computation occurs by using existing secure protocols such as PSI-CA/PSI-DT by De Cristofaro et al. [7, 6]), server-aided PSI by Kamara et al. [10], as well as private recommendation via succinct sketches by Melis et al. [12]. Therefore, we do not provide any additional proofs in the paper as the security of our techniques straightforwardly relies on that of these protocols.

6 Conclusion

This paper presented the result of a measurement study of collaborative predictive blacklisting (CPB). We evaluated a number of metrics on a real-world dataset obtained from DShield.org, aiming to shed light on the effects of collaboration when considering two state-of-the-art approaches, one, non privacy-preserving, relying on trusted central party [14] and another peer-to-peer using privacy-preserving data sharing [9]. We also introduced a third, hybrid approach that aims to combine the best of the two worlds.

Naturally, having access to more logs does not necessarily result in better predictions. In fact, our experiments showed that the techniques proposed by Soldo et al. [14] achieve impressive hit counts (almost doubling the number of correct predictions compared to local predictions) but suffer from poor precision due to high FP. On the other hand, the privacy-friendly decentralized system proposed by Freudiger et al. [9] achieves better F1 scores overall, although with a decreased improvement in TP. Finally, our analysis shows that our hybrid approach outperforms both approaches, balancing out true and



(a) (b) (c)

Figure 5: Computation (a) and communication (b) overhead at each organization for PSI-CA, PSI-DT, and PRP-based scheme, and communication overhead at the STA in PRP scheme (c).

false positives, while maintaining privacy protection.

As part of future work, we plan to conduct a longitudinal measurement to fully grasp the effectiveness of privacy-enhanced CPB in the wild, apply our methods to other datasets, and experiment with more advanced machine learning techniques to improve overall performances.

References

[1] Symantec DeepSight. https://symc.ly/2rXxB1w.

[2] U.S. Anti-Bot Code of Conduct for Internet service providers: Barriers and Metrics Considerations. https://transition.fcc.gov/bureaus/pshs/advisory/csric3/CSRIC_III_WG7_Report_March_%202013.pdf, 2013.

[3] Facebook ThreatExchange. https://threatexchange.fb.com, 2015.

[4] CERT UK. Cyber-security Information Sharing Partnership (CiSP). https://www.cert.gov.uk/cisp/, 2015.

[5] D. Chakrabarti, S. Papadimitriou, D. S. Modha, and C. Faloutsos. Fully automatic cross-associations. In *ACM KDD*, 2004.

[6] E. De Cristofaro, P. Gasti, and G. Tsudik. Fast and Private Computation of Cardinality of Set Intersection and Union. In *CANS*, 2012.

[7] E. De Cristofaro and G. Tsudik. Practical private set intersection protocols with linear complexity. In *Financial Cryptography and Data Security*, 2010.

[8] E. De Cristofaro and G. Tsudik. Experimenting with fast private set intersection. In *TRUST*, 2012.

[9] J. Freudiger, E. De Cristofaro, and A. Brito. Controlled Data Sharing for Collaborative Predictive Blacklisting. In *DIMVA*, 2015.

[10] S. Kamara, P. Mohassel, M. Raykova, and S. Sadeghian. Scaling private set intersection to billion-element sets. In *FC*, 2014.

[11] S. Katti, B. Krishnamurthy, and D. Katabi. Collaborating against common enemies. In *ACM IMC*, 2005.

[12] L. Melis, G. Danezis, and E. De Cristofaro. Efficient Private Statistics with Succinct Sketches. In *NDSS*, 2016.

[13] Red Sky Alliance. http://redskyalliance.org/.

[14] F. Soldo, A. Le, and A. Markopoulou. Predictive blacklisting as an implicit recommendation system. In *INFOCOM*, 2010.

[15] The White House. Executive order promoting private sector cybersecurity information sharing. http://1.usa.gov/1vISfBO, 2015.

[16] J. Zhang, P. A. Porras, and J. Ullrich. Highly predictive blacklisting. In *USENIX*, 2008.

A Reproducing Our Experiments

We provide detailed information for researchers wishing to reproduce the experimental results presented in this paper. Please install Python 2.7 as well as the following Python packages: numpy 1.14.0, scipy 1.0.0, scikit-learn 0.19.1, pandas 0.22.0 and matplotlib 2.0.0. All of the above packages can be installed via "pip".

Code. Source code is available at the following git repository: https://github.com/mex2meou/collsec.git

Dataset. To obtain the DShield dataset that was used in our experiments, use the following download link and extract its contents (i.e., the .pkl files) in the 'data' folder of the cloned repository.

```
1 wget https://www.dropbox.com/s/
      kmiejttl4ceufpp/data.zip
2 cp data.zip collsec/data
3 cd collsec/data
4 unzip data.zip
```

Soldo et al [14]. To replicate the experiments for Soldo et al [14]'s implicit recommendation system, we also need the MATLAB implementation of Chakrabarti et al. [5] for the Cross Associations (CA) co-clustering algorithm. To this end, one should install Octave 4.0.0 as well as the Python package oct2py 3.5.0 (which can also be installed via "pip"). First, to compile the Cross Associations algorithm follow the steps:

```
1 cd collsec/soldo/CA_python
2 octave
3 mex cc_col_nz.c
4 quit
```

Then, to link our Python implementation with the Octave workspace of CA configure accordingly the path in the file 'collsec/soldo/CA_python/ca_utils.py' (line 6). Finally, to run the experiments of Section 3.1:

```
cd collsec/soldo
python soldo.py
```

Note. To configure the parameter k of the k-NN algorithm included in the ensemble method of [14] modify the file 'collsec/soldo/top_neighbors.py' (line 4). Moreover, if experiments for various values of k are executed, modify the file 'collsec/soldo/soldo.py' (line 41) to prevent the CA algorithm from running again.

Controlled Data Sharing. To repeat the experiments for the Controlled Data Sharing system by Freudiger et al. [9], i.e., Section 3.2:

```
%
cd collsec/dimva-global
python dimva-global.py

%
cd collsec/dimva-local
python dimva-local.py
```

Note. To configure the length of the training and testing windows for the system of Freudiger et al. [9], modify the file 'collsec/utils/dimva_util.py'.

Hybrid Approach. To launch the experiments for our proposed hybrid scheme (see Section 4) please execute the following steps:

```
%
%
cd collsec/agglomerative
python agglomerative.py

%
%
cd collsec/kmeans
python kmeans.py

%
%
cd collsec/knn
python knn.py
```

Note. Our implementation by default is configured to utilize a 5-day training window and a 1-day testing one as done in previous work [9, 14]. If one wants to change this setting, please adjust the parameters indicated in the files 'collsec/utils/util.py' and 'collsec/utils/time_series.py'.

Results. The results of all the above scripts are stored in the folder titled 'collsec/results'. To visualize the results and obtain the figures presented in the paper, type the following commands:

```
cd collsec/results

%
python soldo_plots.py

%
python dimva_global_plots.py

%
python dimva_local_plots.py

%
python hybrid_plots.py
```