Compare Before You Buy
Privacy-Preserving Selection of Threat Intelligence Providers

Vos, Jelle; Erkin, Zekeriya; Dörr, Christian

DOI
10.1109/WIFS53200.2021.9648381

Publication date
2021

Document Version
Final published version

Published in
2021 IEEE International Workshop on Information Forensics and Security, WIFS 2021

Citation (APA)
Vos, J., Erkin, Z., & Dörr, C. (2021). Compare Before You Buy: Privacy-Preserving Selection of Threat Intelligence Providers. In *2021 IEEE International Workshop on Information Forensics and Security, WIFS 2021: Proceedings* (pp. 44-49). Article 9648381 (2021 IEEE International Workshop on Information Forensics and Security, WIFS 2021). IEEE. https://doi.org/10.1109/WIFS53200.2021.9648381

Compare Before You Buy: Privacy-Preserving Selection of Threat Intelligence Providers

Jelle Vos
Cyber Security Group
Delft University of Technology
Delft, Netherlands
J.V.Vos@tudelft.nl

Zekeriya Erkin
Cyber Security Group
Delft University of Technology
Delft, Netherlands
Z.Erkin@tudelft.nl

Christian Doerr
Cyber Threat Intelligence Lab
Hasso Plattner Institute, University of Potsdam
Potsdam, Germany
Christian.Doerr@hpi.de

2021 IEEE International Workshop on Information Forensics and Security (WIFS) | 978-1-6654-1717-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/WIFS53200.2021.9648381

***Abstract*—In their pursuit to maximize their return on investment, cybercriminals will likely reuse as much as possible between their campaigns. Not only will the same phishing mail be sent to tens of thousands of targets, but reuse of the tools and infrastructure across attempts will lower their costs of doing business. This reuse, however, creates an effective angle for mitigation, as defenders can recognize domain names, attachments, tools, or systems used in a previous compromisation attempt, significantly increasing the cost to the adversary as it would become necessary to recreate the attack infrastructure each time.**

However, generating such cyber threat intelligence (CTI) is resource-intensive, so organizations often turn to CTI providers that commercially sell feeds with such indicators. As providers have different sources and methods to obtain their data, the coverage and relevance of feeds will vary for each of them. To cover the multitude of threats one organization faces, they are best served by obtaining feeds from multiple providers. However, these feeds may overlap, causing an organization to pay for indicators they already obtained through another provider.

This paper presents a privacy-preserving protocol that allows an organization to query the databases of multiple data providers to obtain an estimate of their total coverage without revealing the data they store. In this way, a customer can make a more informed decision on their choice of CTI providers. We implement this protocol in Rust to validate its performance experimentally: When performed between three CTI providers who collectively have 20,000 unique indicators, our protocol takes less than 6 seconds to execute. The code for our experiments is freely available.

***Index Terms*—private set union, mpsu-ca, indicator of compromise, threat intelligence**

I. Introduction

In their pursuit to maximize their return on investment, cybercriminals will likely reuse as much as possible between their campaigns. Crafting one phishing email takes effort, but sending it out with small customization such as the salutation is trivial. Assembling a malicious attachment and setting up the server infrastructure to act as a command & control system requires time and is costly, but shared across many phishing emails this cost becomes marginal. Unless the adversary is aiming to compromise a high value target, where the cost of accomplishing breach is irrelevant, we can thus assume specific artefacts to reappear across compromization attempts.

This modus operandi – the reuse of tools, techniques and practices across individual intrusion attempts – is also an effective leverage a defender can apply for protection. By identifying elements that were used in the context of one intrusion attempt, such as e-mail addresses, server IPs, fingerprints of network traffic, components included in the malware binary, and filtering for all of these so-called *indicators of compromise* (IoC), future intrusion attempts are more effectively stopped by recognizing one of the previously detected indicators. As a result, the adversary must reinvent the wheel and replace the bulk of the machinery for each attempt, which greatly increases the cost of doing business to the offender.

While each organization could develop a corpus of such information, it is more effective to include indicators observed by third parties, with the rationale that adversaries would reuse parts of the machinery not only across individual intrusion attempts against a particular organization, but across several operating in the same sector or country. Although recognized as desirable [1], the process of information sharing and collaborative threat intelligence generation is hampered by privacy laws and the risk that data shared for comparison purposes may be abused by a malicious participant in the group. For this reason, organizations today typically rely on external cyber threat intelligence providers, which analyse adversaries, their behaviour and tools, and provide a feed against payment [2].

The recent attention on cyber threat intelligence (CTI) has also brought a surge in companies offering such feeds, and studies such as [3] have called into question the quality and relevance of the information provided by many intelligence providers. As also the resources of CTI providers are limited, not only may their threat intelligence show biases towards certain sectors, adversaries or geographical areas, the intelligence between multiple providers may overlap significantly [4]. Hence, defenders want to select multiple providers to cover the multitude of threats they face, while making sure their data is sufficiently distinct. This in turn raises a security and privacy dilemma. Certainly, the CTI provider is unlikely to open up its intelligence database for evaluation purposes, as it is selling the data within it. Similarly, an organization that keeps a local list of indicators may refrain from sharing these with the CTI provider. The dilemma may result in defenders simply buying some feeds, rather than those that contribute most in terms of both relevance and coverage [5].

We address this with a new privacy-preserving protocol that securely assesses the total coverage of datasets across multiple databases without revealing the data itself. The protocol lets a customer query the database of several CTI providers to obtain an estimate of how many *unique* indicators they possess collectively, with the possibility of including their local indicators in the query. This functionality is that of a multi-party private set union-cardinality (MPSU-CA) protocol, which considers n parties who have a set of at most k elements, and computes the cardinality of their union without revealing those input sets. By querying multiple combinations of providers, the customer can make an informed decision about which providers to buy from. For example, choosing the combination that provides the most unique indicators for the lowest price.

For many years, MPSU-CA protocols have been researched. However, to our knowledge, all protocols require at least three stages [6], [7]. Here we define a stage to be a part of the protocol where each party must have been online in order to advance to the next. In our approximate MPSU-CA protocol, a CTI provider only has to take two actions, of which one action only has to be performed every time their set changes. While the result is not exact, the estimate can be made arbitrarily accurate at the cost of computation and bandwidth.

Our protocol is based on the homomorphic ElGamal cryptosystem, which we instantiate in such a way that all parties – the queried CTI providers and the customer – are necessary to decrypt. At its core, the protocol uses a shuffle-decrypt protocol that lets parties simultaneously shuffle and decrypt. During this stage every party has to be online only once. To our knowledge, this is a novel protocol that may also be of independent interest. As an optimization, our protocol enjoys smaller key sizes, smaller ciphertexts and faster computation by using elliptic curves rather than integers. In short, our contributions are as follows:

- A provably secure, efficient MPSU-CA protocol, enabling customers to make an optimal selection of CTI providers
- The protocol requires few interactions since CTI providers only have to encrypt their set once and parameters can be fine-tuned for speed or accuracy
- An open-source proof-of-concept implementation

II. Related work

There already exist works proposing a multi-party private set union-cardinality. However, these works require the parties to participate for multiple stages of communication. By such a stage, we refer to a part of the protocol where a party must have been online to proceed. We discuss four works in this domain and state the number of stages they require.

One of the first multi-party private set union protocols was given by Frikken [6], as well as an extension to the union-cardinality. The idea is to encode the sets of elements as the roots of polynomials. After multiplying the polynomials together, the roots of the resulting polynomial are the elements in the original sets. By encrypting the coefficients of the polynomials using a homomorphic threshold cryptosystem, no party can distill the information contained within the polynomials, while allowing for polynomial multiplication and evaluation. Each party takes a turn in evaluating the encrypted polynomial that is passed around, in a way that finally returns a set of numbers corresponding to the total number of elements. Elements that result in 0 are duplicates, while unique elements result in a random number. Then, one can count the amount of random numbers to reveal the union-cardinality. The protocol takes 6 stages and relies on the Paillier cryptosystem.

Burkhart et al. [8] propose SEPIA, which is a library for securely aggregating multi-domain network data using Shamir's secret sharing. They provide a protocol for the distinct-count problem, which is identical to a set union-cardinality. They encode each party's set as a bitset, which is a vector of zeroes and ones: If element 1 is in the set, they set the first bit to 1. If element 2 is in the set, they do the same for the second bit, and so forth. They invert the bitsets, compute the AND operation between them, and return the number of possible elements minus the total sum of the aggregated inverted bitset. The AND and sum operations are executed using secret sharing. Due to the AND operation, the number of rounds of interaction in the protocol scales logarithmically with the number of parties n. By using bitsets, the protocol also scales with the number of possible elements, instead of with a pre-determined number such as when using Bloom filters.

Blanton & Aguiar [9] propose a general framework for private set and multiset operations, which relies on the possibility for elements to be sorted. Their work is based on secret sharing and makes of several sub-protocols, including oblivious sorting and secure comparisons. The number of stages required to run this protocol depends on the choice of these sub-protocols.

Debnath et al. [7] propose an *intersection*-cardinality protocol using (n, n)-exponential ElGamal, so the parties can generate a public key without a trusted dealer. The protocol has every party encode their sets as a Bloom filter, then aggregate these to get the Bloom filter corresponding to the intersection. The leader selects those encrypted bins corresponding to the elements in its set and aggregates them. The parties then take turns shuffling the ciphertexts, after which they collaboratively decrypt. The leader can simply count the number of bins that were 0. This protocol leaks the size of the leader's set because it only submits those bins corresponding to the leader's elements to shuffle and decrypt. The protocol requires 3 stages of communication.

III. Preliminaries

Now we explain Bloom filters, the ElGamal encryption scheme over elliptic curves and our security assumptions. Our notation is given in the following table:

Sets & Bloom filters		EC-ElGamal		Other	
k	Max. set size	$\mathbb{G}$	Cyclic group	n	Number of parties
m	Number of bins	q	Order of $\mathbb{G}$	$\mathcal{P}_i$	Party i
h	Number of hashes	sk_i	Secret key of $\mathcal{P}_i$	$\overset{c}{\equiv}$	Comp. indistinguishable
F	Filled bins	pk_i	Public key of $\mathcal{P}_i$	π_i	Permutation function of $\mathcal{P}_i$
N	Inserted elements	$\langle \alpha, \beta \rangle$	ElGamal ciphertext	κ	Comp. security parameter
$\hat{X}$	Bloom filter of X	y	Randomness	λ	Stat. security parameter
p	Selectivity ratio	$\mathcal{O}$	Identity element	$\in_{\mathrm{R}}$	Random element

A. Bloom filters

A Bloom filter contains m bins, and each element is assigned to one or more of those bins using $h \geq 1$ hash functions. All bins of the Bloom filter start at 0, but when it is chosen by a hash function it is set to 1 and we say that it is 'filled'. Bloom filters with the same m and hash functions can then be combined using `AND` and `OR` operations to perform set intersections and unions, respectively. Due to this property, both approximate private set operations [10] and multi-party private set operations [11], [12] based on Bloom filters have already been proposed.

Assuming h perfect and independent hash functions that perfectly distribute elements over the m bins, we express the expected number of false positives when performing k membership queries. We provide two lemmas:

Lemma 1. *The probability that a given bin in the Bloom filter is still* 0 *after* N *inserted elements is:*

$$\Pr[\textit{bin is } 0] = \left(1 - \frac{1}{m}\right)^{hN} \approx \exp\left(-\frac{hN}{m}\right)$$

What follows, is the Bloom filter's false positive rate ε_N:

Lemma 2. *The probability* ε_N *that a random element is included in the set encoded by a Bloom filter is:*

$$\varepsilon_N \approx \left(1 - \exp\left(-\frac{hN}{m}\right)\right)^h$$

Since Bloom filters are approximations, it is only possible to estimate the number of elements that one encodes, rather than exactly. It suffices to count the number of filled bins F and to know the Bloom filter's parameters. Note that the distribution of the number of filled bins given the number of inserted elements N is binomial, following the probability from Lemma 2:

$$\Pr[\mathbf{F} = F \mid \mathbf{N} = N] \sim \mathcal{B}\left(m, \left(1 - \exp\left(-\frac{hN}{m}\right)\right)\right) \quad (1)$$

Theorem 3. *The expected number of elements encoded by a Bloom filter with* F *filled bins is given by:*

$$\mathrm{E}[N \mid F] \approx -\frac{m}{h} \ln\left(1 - \frac{F}{m}\right) \quad (2)$$

Proof. The probability for a bin to be 1 is expected to be equal to the ratio of filled bins $\Pr[\text{bin is } 1] = \frac{F}{m}$. It follows that:

$$\frac{F}{m} \approx 1 - \exp\left(-\frac{hN}{m}\right) \quad (3)$$

$$N \approx \log_{\exp\left(-\frac{h}{m}\right)}\left(1 - \frac{F}{m}\right) \quad (4)$$

$$\approx \frac{\ln\left(1 - \frac{F}{m}\right)}{\ln\left(\exp\left(-\frac{h}{m}\right)\right)} \quad (5)$$

$$\approx -\frac{m}{h} \ln\left(1 - \frac{F}{m}\right) \qquad \square$$

For use later, we define the cardinality function for Bloom filters as $\textsc{cardinality}(F) \approx E[N|F]$. Unfortunately it is hard to give guarantees about this estimation because the reversed conditional probability of Equation 1 is not a binomial. In fact, the distribution is asymmetric and it gets wider for a larger number of filled bins F. It is proportional to:

$$\Pr[\mathbf{N} = N \mid \mathbf{F} = F] \propto \Pr[\text{bin is } 1]^F \Pr[\text{bin is } 0]^{m-F} \quad (6)$$

Since estimation of the cardinality of the cardinality only involves the number of filled bins, one can shuffle the bins of a Bloom filter to obscure the elements encoded within it but retain the ability to estimate its cardinality. This forms the basis of our MPSU-CA protocol.

B. ElGamal over elliptic curves

The ElGamal cryptosystem allows the use of any group $\mathbb{G}$ in which the DDH assumption holds [13], here noted additively:

Definition 1 (Decisional Diffie-Hellman)**.** The DDH assumption states that given ag and bg for some random $a, b \in \mathbb{Z}_{|\mathbb{G}|}$, abg is computationally indistinguishable from some $r \in_{\mathrm{R}} \mathbb{G}$.

For some elliptic curve groups, DDH is assumed to hold: in this work, we use the Montgomery curve Curve25519 [14]. This curve has a co-factor of 8, which means that the prime order subgroup that we actually use in cryptographic applications is one eight of the size of the total group. However, when implementing protocols, it is possible to introduce bugs related to these co-factors. For this reason, we use a highly-optimized abstraction that allows us to use a curve with a co-factor to realize a prime-order group [15], [16], so there is no co-factor. Additionally, this technique allows for faster equality checks.[1]

C. Threat model & security assumptions

We consider adversaries in the semi-honest model, so the parties perform the protocol according to its specifications. They do try to learn as much as possible. We allow for $n - 1$ parties to collude. No secure channels are required, but we assume the channels are lossless. The security of the protocol is implied by the DDH assumption.

IV. Shuffle-decrypt protocol

When shuffling and decryption are performed separately using a threshold cryptosystem, the shuffling requires each party to sequentially permute the ciphertexts and re-randomize them, then the decryption requires each party to partially decrypt in parallel. If a party is unavailable in any of those two stages, the protocol cannot progress. By shuffling and decrypting in one pass, we require parties to be online for only one stage and we save the parties from sending unnecessary messages. This comes at the cost of sending along at most n more curve points for each ciphertext to be decrypted.

For our protocol we adapt the ElGamal cryptosystem to behave like an (n, n)-cryptosystem. The intuition behind the protocol is to let each party in turn shuffle the ciphertexts and 'peel back' one layer of encryption. At the same time, such a party re-randomizes the ciphertexts to ensure that

[1] https://ristretto.group/details/equality.html

the ciphertexts they received are indistinguishable from those they send to the next party. More specifically, this protocol is an aggregate-shuffle-decrypt protocol, since the first step homomorphically aggregates the ciphertexts.

A. Protocol description

To ease interpretation we use additive notation and a special notation for vector indexing: for a vector x, $x[i]$ denotes the ith element of x. Additionally, q is the order of the cyclic group $\mathbb{G}$ of the ElGamal cryptosystem generated by g. Finally, each party $\mathcal{P}_i$ has a secure permutation function $\pi_i(\_)$ that takes a vector like x and returns a permuted vector of the same length. We assume that each party has a secret key $sk_i \in_{\mathrm{R}} \mathbb{Z}_q$ and a corresponding public key $pk_i \leftarrow sk_i g$ known to all parties.The leader is $\mathcal{P}_1$, without loss of generality.

Aggregate-shuffle-decrypt protocol

1) Each party $\mathcal{P}_i$ for $i = 1, \dots, n$ sends m ciphertexts of messages M_i to the leader $\mathcal{P}_1$, encrypted with their own public key pk_i using randomness $y_i[j] \in_{\mathrm{R}} \mathbb{Z}_q$, and for $j = 1, \dots, m$:

$$\langle \alpha_i[j], \beta_i[j] \rangle = \langle y_i[j]g, M_i[j] + y_i[j]pk_i \rangle$$

2) Leader $\mathcal{P}_1$ constructs a vector of tuples $\alpha[j] = \langle \alpha_1[j], \dots, \alpha_n[j] \rangle$ and $\beta[j] = \sum_{i=1}^{n} \beta_i[j]$, and sends them to party $\mathcal{P}_n$.
3) In turn, each party $\mathcal{P}_i$ for $i = n, \dots, 2$ shuffles and partially decrypts the ciphertexts, sending the results to the next party $\mathcal{P}_{i-1}$:
   - Party $\mathcal{P}_i$ uses permutation function π_i to permute $\alpha \leftarrow \pi_i(\alpha)$ and $\beta \leftarrow \pi_i(\beta)$.
   - Party $\mathcal{P}_i$ removes its corresponding entry from each tuple in α and randomizes it so that $\alpha[j]$ becomes:

     $$\left\langle \alpha_1[j] + y'_{i,1}[j]g, \dots, \alpha_{i-1}[j] + y'_{i,i-1}[j]g \right\rangle,$$

     for $j = 1, \dots, m$, using randomness $y'_i[i'] \in_{\mathrm{R}} \mathbb{Z}_q$.
   - Party $\mathcal{P}_i$ partially decrypts each $\beta[j]$ and uses the same randomness $y'_i[i']$ to compute:

     $$\beta[j] \leftarrow \beta[j] - sk_i \alpha_i[j] + \sum_{i'=1}^{i-1} y'_{i,i'}[j] pk_{i'}$$

4) Leader $\mathcal{P}_1$ decrypts $M = \beta[j] - sk_1 \alpha_1[j]$ for $j = 1, \dots, m$.

B. Correctness

Let us denote the composition of permutations as $\circ$, and the final permutation as $\pi = \pi_1 \circ \dots \circ \pi_n$. To prevent complicated notation, we say that when these permutation functions are supplied with an index, that index is mapped to the permuted index. Then, the protocol is considered correct when it holds that $M[\pi(j)] = M_1[j] + \dots + M_n[j]$ for all $j = 1, \dots, m$. Since the permutation functions shuffle both α and β according to the same permutation, their elements are not shuffled relative to each other. For this reason we provide a correctness proof where all permutation functions are the identity function, to ease notation.

Theorem 4. *When all permutation functions are the identity, it holds that $M[j] = \sum_{i=1}^{n} M_i[j]$ for every $j = 1, \dots, m$.*

Proof. After step 4, we can express $a_i[j]$ as:

$$\alpha_i[j] = y_i[j]g + \sum_{i'=i+1}^{n} y'_{i',i}[j]g \tag{7}$$

Combining steps 3 and 4, and substituting $\beta[j] = \sum_{i=1}^{n} \beta_i[j]$:

$$\begin{aligned}
M[j] =& \sum_{i=1}^{n} \beta_i[j] - \sum_{i=1}^{n} sk_i \alpha_i[j] + \sum_{i=1}^{n} \sum_{i'=1}^{i-1} y'_{i,i'}[j] pk_{i'} && (8)\\
=& \sum_{i=1}^{n} M_i[j] + \sum_{i=1}^{n} y_i[j] pk_i - \sum_{i=1}^{n} y_i[j] pk_i - \\
& \sum_{i=1}^{n} \sum_{i'=i+1}^{n} y'_{i',i}[j] pk_i + \sum_{i=1}^{n} \sum_{i'=1}^{i-1} y'_{i,i'}[j] pk_{i'} && (9)\\
=& \sum_{i=1}^{n} M_i[j] - \sum_{i=1}^{n} \sum_{i'=i+1}^{n} y'_{i',i}[j] pk_i + \\
& \sum_{i=1}^{n} \sum_{i'=i+1}^{n} y'_{i',i}[j] pk_i && (10)\\
=& \sum_{i=1}^{n} M_i[j] && \square
\end{aligned}$$

C. Security

To underline the security of the aggregate-shuffle-decrypt protocol, we prove that even with $n - 1$ colluding parties, at least for one honest party its outputs are computationally indistinguishable from randomness. As a result, none of the other parties can learn how the inputs were shuffled. For an argument about the confidentiality of the ciphertexts we refer the reader to the paper by ElGamal [13]. In our proof, we denote the inputs α and β for an honest party as α^{in} and β^{in}, and its outputs as α^{out} and β^{out}.

Theorem 5. *For an honest and non-colluding party $\mathcal{P}_i$, for each index $j = 1, \dots, m$ it holds that $\alpha^{out}[j] \overset{c}{\equiv} r$ and $\beta^{out}[j] \overset{c}{\equiv} r'$ where $r, r' \in_R \mathbb{Z}_q$.*

Proof. We express the outputs of party $\mathcal{P}_i$ in terms of its inputs:

$$\alpha^{\text{out}}[j] \leftarrow \alpha^{\text{in}}[j] + y'_{i,i'}[j]g \tag{11}$$

$$\beta^{\text{out}}[j] \leftarrow \beta^{\text{in}}[j] - sk_i \alpha_i[j] + \dots \tag{12}$$

Since $y'_{i,i'}[j] \in_{\mathrm{R}} \mathbb{Z}_q$ and is only known to this party, $y'_{i,i'}[j]g$ is statistically indistinguishable from randomness. Thereby, $\alpha^{\text{out}}[j] \overset{c}{\equiv} r$. Moreover, since $sk_i \in \mathbb{Z}_q$ and is only known to this party, for any other party $sk_i \alpha_i$ is computationally indistinguishable from randomness by the DDH assumption. So, $\beta^{\text{out}}[j] \overset{c}{\equiv} r'$. □

D. Efficiency

We denote the number of ciphertexts by m. In the first step, each party performs m ElGamal encryptions, which takes $O(m\kappa)$ operations. Then, in step 2, the leader sums mn curve points, which takes $mn\kappa$ operations. In step 3, each party performs a permutation, which we deem to be negligible in terms of its runtime. Additionally, assistant $\mathcal{P}_i$ randomizes $m(i-1)$ curve points, which takes at most $O(mn\kappa)$, and partially decrypts, which also takes at most $O(mn\kappa)$ operations. The same goes for the leader in step 4. So, each party performs $O(mn\kappa)$ operations asymptotically. Additionally, each party sends at most $O(mn\lambda)$ bits during steps 2 & 3.

V. MPSU-CA PROTOCOL

As explained in Section III-A, to compute the union of multiple Bloom filters, we combine them using an `OR` operation: we let parties encode a 0 as the identity and 1 as randomness, so that when aggregated, the result is only 0 when all inputs were 0. The identity is $\mathcal{O}$, denoting the point at infinity of an elliptic curve group. Here, leader $\mathcal{P}_1$ is the querying customer. If the customer has a local set of indicators, they can join the protocol, otherwise they can submit the empty set.

MPSU-CA protocol

1) All parties $\mathcal{P}_i$ for $i = 1, \ldots, n$ encode their set X_i as Bloom filter $\hat{X}_i$.
2) All parties $\mathcal{P}_i$ for $i = 1, \ldots, n$ take part in the **aggregate-shuffle-decrypt** protocol with the bins of $\hat{X}_i$, where:
$$M_i[j] = \begin{cases} \mathcal{O} \in \mathbb{G}_T & \text{if } \hat{X}_i[j] = 0 \\ r \in_{\mathrm{R}} \mathbb{G}_T & \text{if } \hat{X}_i[j] = 1 \end{cases}$$
3) The leader $\mathcal{P}_1$ extracts M, counts the points for which it holds that $M[j] \neq \mathcal{O}$ as F, and returns $z \leftarrow$CARDINALITY(F).

As an optimization, rather than encrypting a randomly generated element, a party chooses two random elements for alpha and beta. Also note that an intersection-cardinality protocol follows similarly by inverting the bins of the Bloom filter and checking for $M[j] = \mathcal{O}$.

A. Selectivity

The probability distribution of the cardinality estimates for a Bloom filter becomes wide when the number of filled bins is large, as discussed in Section III-A. To prevent poor estimates for highly-filled Bloom filters, one could increase the size of the Bloom filter m abundantly, but this would increase computation and communication. As an alternative, we introduce 'selectivity', inspired by sketches. Instead of encoding every element from a party's set in the Bloom filter, we only insert an element with a specific probability p, called the selectivity ratio. This is not a coin toss, but it is deterministically decided by a public hash function. A simple way to achieve $p = 50\%$ is to only insert elements of which the hash starts with 0. In the case where $p = 100\%$, it functions like a regular Bloom filter. We accommodate for selectivity in the final estimate. Again, we use $\mathrm{E}[N]$ as our estimate, which can be derived in the same way as in Theorem 3:

Lemma 6. *The expected number of elements encoded by a Bloom filter with F filled bins and a selectivity ratio p is given by:*
$$\mathrm{E}[N] \approx -\frac{pm}{h} \ln\left(1 - \frac{F}{m}\right)$$

B. Efficiency

The computation of this protocol is based on the aggregate-shuffle-decrypt protocol, since we deem the operations performed in step 1 and 3 to be negligible. Each party computes $O(mn\kappa)$ operations, and sends $O(mn\lambda)$ bits. So, the total complexities are $O(mn^2\kappa)$ and $O(mn^2\lambda)$, respectively.

C. Use in practice

Since the shuffle-decrypt protocol requires a party to create ciphertexts using their own public key, a company has to perform encryption only once for one set of IoCs and one set of Bloom filter parameters, after which the customer can initiate comparisons between any providers. If they agree with the query, they take part in the shuffle-decrypt protocol, which only requires one action. They can also disagree, for example because they do not agree who is querying, the number of involved parties is too low, or the Bloom filter is too large, which would require more compute. We assume that providers otherwise act faithfully, motivated by a financial incentive.

VI. RESULTS

We evaluate the performance of an implementation[2] of our approximate MPSU-CA protocol in Rust on sets of IP addresses in the context of buying cyber threat intelligence. We stress that the protocol is not restricted to IP addresses.

A. Setup

To evaluate the performance of our MPSU-CA protocol, we perform 20 repetitions of experiments using different parameters. We measure the time it takes to encrypt, the time it takes to perform the shuffle-decrypt protocol and we keep track of the estimated cardinalities. We executed all experiments on a 64-bit Unix machine with an Intel i7-1065G7 processor with 8 cores at 1.30GHz. We run all parties sequentially on a single core. This is not detrimental to performance, though, since the shuffle-decrypt must be performed sequentially anyways. It does affect the encryption stage. We analyze two scenarios:

- **Scenario A:** Three parties with $10,000$ elements each, whose union contains $20,000$ elements, see Table I.
- **Scenario B:** Five parties with $20,000$ elements each, whose union contains $50,000$ elements, see Table II.

For each of the scenarios we evaluate the performance using a Bloom filter with $m = 10,000$ bins, and a higher accuracy

[2] The code is available at: https://github.com/jellevos/mpsu-ca

TABLE I
RESULTS FOR SCENARIO A AVERAGED OVER 20 EXPERIMENTS

	Lower accuracy			**Higher accuracy**		
	$p=1$	$p=\frac{1}{2}$	$p=\frac{1}{4}$	$p=1$	$p=\frac{1}{2}$	$p=\frac{1}{4}$
Bins m	10,000	5,000	2,500	50,000	25,000	12,500
Encryption [s]	1.4	0.7	0.4	12.2	6.0	3.0
Shuffle-decrypt [s]	4.4	2.3	1.1	21.8	11.0	5.4
Estimate mean	19,986	19,925	20,069	19,995	20,011	20,061
Standard deviation	±251	±386	±418	±62	±136	±285

TABLE II
RESULTS FOR SCENARIO B AVERAGED OVER 20 EXPERIMENTS

	Lower accuracy			**Higher accuracy**		
	$p=1$	$p=\frac{1}{2}$	$p=\frac{1}{4}$	$p=1$	$p=\frac{1}{2}$	$p=\frac{1}{4}$
Bins m	10,000	5,000	2,500	50,000	25,000	12,500
Encryption [s]	1.5	0.8	0.4	17.5	8.7	4.4
Share-decrypt [s]	12.0	6.1	3.0	60.0	30.4	15.0
Estimate mean	49,539	49,510	50,771	50,066	50,081	50,065
Standard deviation	±1,284	±1,364	±2,381	±165	±307	±490

Bloom filter with $m = 50,000$ bins. For each combination of parameters we also vary the selectivity ratio p between 100%, 50% and 25%, and adjust the number of bins accordingly. We set the number of hash functions $h = 1$ to make the probability distributions for higher cardinalities as narrow as possible. In all experiments, the setup and aggregation stages took at most 100 milliseconds together, so we omit them from the tables.

B. Scenario A

See Table I. In the higher accuracy case with a larger Bloom filter, run time increases but the standard deviation decreases. Also, run time of the shuffle-decrypt stage scales roughly linearly with the number of bins m. As a sign of validation, the estimated mean is indeed centered around 20,000.

C. Scenario B

See Table II. While run time for $p = 100\%$ in the higher accuracy case takes more than a minute, we see that the selectivity ratio $p = 25\%$ with $m = 12{,}500$ brings it down to less than 20 seconds, while keeping a standard deviation of only 490, which is significantly better than the case where $m = 10{,}000$ and $p = 100\%$: by being more selective, less bins are filled so the probability distribution stays more narrow.

VII. Conclusion

We propose an MPSU-CA protocol that lets organizations compare the lists of indicators sold by CTI providers to find a combination of providers that offer a set with relevant indicators, without too much overlap. As such, they can better protect themselves against cyber criminals. Our protocol takes only two stages, or just one stage if they publish their encrypted set in advance, as compared to previous work which requires at least three. Even for a total of 100,000 elements, an organization gets an accurate estimate of the unique number of indicators within 20 seconds.

References

[1] "Enisa threat landscape report 2017," European Network Information and Security Agency, Tech. Rep., 2017.

[2] K. Oosthoek and C. Doerr, "Cyber threat intelligence: A product without a process?" *International Journal of Intelligence and CounterIntelligence*, vol. 34, no. 2, pp. 300–315, 2021. [Online]. Available: https://doi.org/10.1080/08850607.2020.1780062

[3] "2015 data breach investigations report," Verizon, Tech. Rep., 2015.

[4] H. Griffioen, T. M. Booij, and C. Doerr, "Quality evaluation of cyber threat intelligence feeds," in *Applied Cryptography and Network Security - 18th International Conference, ACNS 2020, Rome, Italy, October 19-22, 2020, Proceedings, Part II*, ser. Lecture Notes in Computer Science, M. Conti, J. Zhou, E. Casalicchio, and A. Spognardi, Eds., vol. 12147. Springer, 2020, pp. 277–296. [Online]. Available: https://doi.org/10.1007/978-3-030-57878-7_14

[5] X. Bouwman, H. Griffioen, J. Egbers, C. Doerr, B. Klievink, and M. van Eeten, "A different cup of TI? the added value of commercial threat intelligence," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, pp. 433–450. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/bouwman

[6] K. B. Frikken, "Privacy-preserving set union," in *Applied Cryptography and Network Security, 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007, Proceedings*, ser. Lecture Notes in Computer Science, J. Katz and M. Yung, Eds., vol. 4521. Springer, 2007, pp. 237–252. [Online]. Available: https://doi.org/10.1007/978-3-540-72738-5_16

[7] S. K. Debnath, P. Stanica, N. Kundu, and T. Choudhury, "Secure and efficient multiparty private set intersection cardinality," *Adv. Math. Commun.*, vol. 15, no. 2, pp. 365–386, 2021. [Online]. Available: https://doi.org/10.3934/amc.2020071

[8] M. Burkhart, M. Strasser, D. Many, and X. A. Dimitropoulos, "SEPIA: privacy-preserving aggregation of multi-domain network events and statistics," in *19th USENIX Security Symposium, Washington, DC, USA, August 11-13, 2010, Proceedings*. USENIX Association, 2010, pp. 223–240. [Online]. Available: http://www.usenix.org/events/sec10/tech/full_papers/Burkhart.pdf

[9] M. Blanton and E. Aguiar, "Private and oblivious set and multiset operations," *Int. J. Inf. Sec.*, vol. 15, no. 5, pp. 493–518, 2016. [Online]. Available: https://doi.org/10.1007/s10207-015-0301-1

[10] F. Kerschbaum, "Outsourced private set intersection using homomorphic encryption," in *7th ACM Symposium on Information, Compuer and Communications Security, ASIACCS '12, Seoul, Korea, May 2-4, 2012*, H. Y. Youm and Y. Won, Eds. ACM, 2012, pp. 85–86.

[11] R. Inbar, E. Omri, and B. Pinkas, "Efficient scalable multiparty private set-intersection via garbled bloom filters," in *Security and Cryptography for Networks*, D. Catalano and R. De Prisco, Eds. Springer International Publishing, 2018, vol. 11035, pp. 235–252.

[12] A. Miyaji and K. Shishido, "Efficient and quasi-accurate multiparty private set union," *IEEE International Conference on Smart Computing*, p. 6, 2018.

[13] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, ser. Lecture Notes in Computer Science, G. R. Blakley and D. Chaum, Eds., vol. 196. Springer, 1984, pp. 10–18. [Online]. Available: https://doi.org/10.1007/3-540-39568-7_2

[14] D. J. Bernstein, "Curve25519: New diffie-hellman speed records," in *Public Key Cryptography - PKC 2006, 9th International Conference on Theory and Practice of Public-Key Cryptography, New York, NY, USA, April 24-26, 2006, Proceedings*, ser. Lecture Notes in Computer Science, M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, Eds., vol. 3958. Springer, 2006, pp. 207–228. [Online]. Available: https://doi.org/10.1007/11745853_14

[15] M. Hamburg, "Decaf: Eliminating cofactors through point compression," in *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, ser. Lecture Notes in Computer Science, R. Gennaro and M. Robshaw, Eds., vol. 9215. Springer, 2015, pp. 705–723. [Online]. Available: https://doi.org/10.1007/978-3-662-47989-6_34

[16] H. d. Valence, I. Lovecruft, and T. Arcieri, "The ristretto group." [Online]. Available: https://ristretto.group/why_ristretto.html