Private Sharing of IOCs and Sightings

Tim van de Kamp
University of Twente
t.r.vandekamp@utwente.nl

Andreas Peter
University of Twente
a.peter@utwente.nl

Maarten H. Everts
TNO & University of Twente
maarten.everts@tno.nl

Willem Jonker
University of Twente
w.jonker@utwente.nl

ABSTRACT

Information sharing helps to better protect computer systems against digital threats and known attacks. However, since security information is usually considered sensitive, parties are hesitant to share all their information through public channels. Instead, they only exchange this information with parties with whom they already established trust relationships.

We propose the use of two complementary techniques to allow parties to share information without the need to immediately reveal private information. We consider a cryptographic approach to hide the details of an indicator of compromise so that it can be shared with other parties. These other parties are still able to detect intrusions with these cryptographic indicators. Additionally, we apply another cryptographic construction to let parties report back their number of sightings to a central party. This central party can aggregate the messages from the various parties to learn the total number of sightings for each indicator, without learning the number of sightings from each individual party.

An evaluation of our open-source proof-of-concept implementations shows that both techniques incur only little overhead, making the techniques prime candidates for practice.

Keywords

cryptography; evaluation; private information sharing

1. INTRODUCTION

Over the recent years, many information sharing platforms and services (e.g., Critical Stack's Intel, Microsoft's Interflow, AlienVault's Open Threat Exchange) and standards (e.g., STIX, TAXII, CybOX) have been developed to improve protection against digital attacks. However, it is broadly recognized that security information sharing can also have negative consequences (e.g., reputation damage, legal problems, or revealing a detection method to an attacker), yet not much is done in practice to reduce these risks related to information sharing. Whenever these risks are addressed, the solution usually resorts to trust relationships. The traffic light protocol (TLP), for example, requires all parties to respect the protocol and not to share the information with unauthorized parties. Such a solution does not help if a trusted party is being compromised and does not protect against accidental leakage.

Using existing cryptographic primitives, we can do better. While applying cryptography is by no means a magical solution that solves all challenges in the secure sharing of sensitive information, we can protect sensitive information better than if these techniques were not used at all. However, applying cryptography will have a performance cost.

In this short paper we analyze the practicality and efficiency of two cryptographic solutions for private information sharing. Through proof-of-concept implementations of two tailored cryptographic solutions, we show that the incurred performance overhead can be very little, making them prime candidates for practice.

We consider the private sharing of indicators of compromise (IOCs) and the private reporting of sightings. Both are applied in the source-subscriber information sharing model that is schematically depicted in Figure 1. Here, a single *source*, e.g., a computer emergency response team (CERT) or an anti-virus company, has sensitive information about threats and attacks. This source wants to share its security information with various *subscribers* so that they can monitor their own systems to learn whether they are possibly under attack. One way to share this security information is using IOCs. These IOCs consist of a *rule*, formally a propositional formula where the propositional variables are defined over *features* or *observables*, e.g., an internet protocol (IP) address or the hash of a malicious file. Examples of simple IOCs include `destIP = 198.51.100.43`, which can be interpreted as "if one of your machines connects to this IP address, it is potentially compromised, e.g., part of some botnet," and `fileHash = bbd758d9b26404d9b28957af865d1234`. A subscriber can analyze its systems by extracting the features from its network or computer systems and use these features to evaluate whether the rule *matches* or not. By privately sharing IOCs, we want a party to be able to check if its systems are potentially compromised, while keeping the IOC's content hidden as long as it is not matched.

Often the source is interested in how effective its IOCs are and in statistics as how often every IOC is matched. When the subscriber finds a match for a specific IOC, it can report a *sighting* back to the source. If the subscriber investigated the underlying cause of the match, it can additionally indicate whether the sighting is a false or a true positive. The

WISCS '16, October 24, 2016, Vienna, Austria.

ISBN 978-1-4503-4565-1/16/10...$15.00

DOI: http://dx.doi.org/10.1145/2994539.2994544

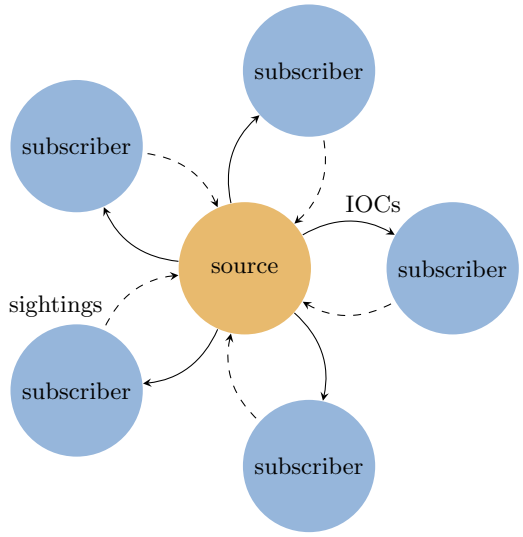

Figure 1: Overview of communication between a source and its subscribers.

source can use this information to improve its IOCs and use the number of true positives to further improve its situational awareness. To privately report a sighting to the source we want to require the source to aggregate the number of sightings from multiple subscribers and only be able to learn the aggregated value. This way the number of sightings for each individual subscriber will remain hidden.

2. RELATED WORK

One of the first works on privacy-preserving data sharing considers the sharing of alerts [8]. Their approach to this problem is to sanitize the sensitive data associated with the alerts before sharing the alert. Gross et al. [7] choose another approach by applying hashes and Bloom filters on the alert data. With this technique they are able to do simple correlations on the alerts, however they do not evaluate a proof-of-concept implementation. Simple correlations of alerts between different parties are also possible using private set intersection (PSI). Freudiger et al. [6] extensively evaluate the practicability of using PSI for predictive IP address blacklisting, but show no results regarding the required computational time of their algorithms.

A simple and efficient technique for interactive searching over other party's security data is proposed by Allman et al. [1]. In their work, they consider an authority that wants to privately query other entities for specific network patterns without revealing what it is looking for. The other entities only want to share their network data with the authority if their data exactly matches the query. They solve the problem by using a hash function with many collisions to create an obfuscated search query. If the query hash matches the hash of the entity's network traffic, the entity sends the matched network record encrypted to the authority. The authority is only able to decrypt the entire record only if the record matches the original search query. The authors did not evaluate the performance of their construction or create an implementation.

The private evaluation of network-based IOCs is considered by Sherry et al. [10]. As their solution is aimed at supporting deep packet inspection (DPI) at middleboxes and requires the sender to use their protocol, it is not designed to protect against attackers not conforming to the protocol.

For the computation of statistics over privacy-sensitive data, interactive secure multi-party computation (MPC) between all parties is most often considered [5, 2, 4]. We are, however, interested in a solution that only requires little communication between the subscribers and the source and no communication between the subscribers themselves.

3. PRIVATE INFORMATION SHARING

We propose two complementing cryptographic techniques for private information sharing. The techniques can be applied separately from each other as they solve two independent problems.

First, we discuss a technique to hide part of an IOC's signature as long as the IOC does not match the subscriber's data. We only consider 'simple' IOCs that consist of formulas in disjunctive normal form (DNF), not containing negations, and where the propositional variables can be evaluated using an equality match. An example of such an IOC is `(destIP = 198.51.100.43 ∧ destPort = 80) ∨ (destIP = 198.51.100.43 ∧ destPort = 443)`. A large portion of commonly used IOCs can be written in this form, AlienVault's Open Threat Exchange[1], for example, only contains these simple IOCs.

Later, we explain how sightings can be privately reported using another cryptographic technique that is proposed by Shi et al. [11].

3.1 Private IOC Sharing

In the current practice of IOC sharing, a file describing one or multiple IOCs is sent to the subscribers. The subscribers locally evaluate the IOCs on their systems. We observe that in this scenario it is intrinsically impossible to fully hide the IOC while still allowing a subscriber to evaluate the rule. Since a subscriber needs to locally evaluate the IOC on its own data, it can always try to evaluate the IOC on false data to see whether it matches—no matter what (cryptographic) protection mechanism is applied. Once it finds a match, for example by conducting a search on the most likely feature values, it automatically learns what the IOC describes.

Although it is theoretically impossible to hide the IOC's content in this scenario, we can make it hard for an attacker to learn the content, while an honest subscriber is still able to efficiently match its data with the IOC. We take the approach of obfuscating the IOC's content, i.e., the feature values, but not the IOC's structure or which features need to be considered to match the IOC. To hide the IOC's content, we could try to simply hash the feature value using a cryptographic hash function H, resulting in IOCs such as `destIP =` H`(198.51.100.43) ∧ destPort =` H`(80)`. For a subscriber to match its observables to an IOC, it would now only need to compute the hash of its observables, the rest of the matching algorithm remains the same.

However, this approach provides little protection. Since every individual feature variable can only take a limited set of values, an attacker could easily precompute the hashes of feature values for all possible features and store them in a large lookup table. Now, for an attacker to learn the content of such an obfuscated rule, the attacker just looks up

[1]See https://otx.alienvault.com/.

the hashes in the table and immediately learns the original plaintext values. The problem is that the search space for the attacker is relatively small. A better approach would be to hash not every feature and its value individually, but to hash the concatenation of the features and its values when they appear as a conjunction in the rule, e.g., resulting in `destIP‖destPort = H(198.51.100.43‖80)`. Since we assumed that every IOC can be written in DNF without containing negations, we can split every rule on the disjunctions and obtain subrules containing only conjunctions. For every IOC containing at least one conjunction, the search space now significantly increases.

We can improve the construction even further, by using a non-secret salt, chosen at random for each IOC, in the computation of the hash. This prevents an attacker from precomputing one giant lookup table for single feature values or the combinations of different feature values. Note that one can make it even harder for an attacker to create such a lookup table by increasing the running time of the hashing—or matching algorithm in general—or by increasing the storage requirements for such a lookup table. However, this will also directly negatively impact an honest subscriber from evaluating its observables and therefore we do not consider this a viable solution.

Finally, with a simple extension to our above described scheme, we can selectively share the description of the IOC or a course of action (COA) when the subscriber matches the IOC. A COA specifies measures that help to prevent the attack from succeeding or mitigate the impact of the attack. Such a COA might be specific to an IOC and reveal information about the IOC if shared in the clear. So, instead of hashing the feature values of an observable, we use the feature values in a key derivation function (KDF) to derive a symmetric key and use this key to encrypt the COA. This assures that only parties with the right IOC values, and thus the correct derived key, can decrypt the encrypted COA and learn its contents.

It is important to realize that the cryptographic IOC need to be signed or sent over a secure channel that provides integrity protection and authenticity. Otherwise an attacker might be able to change the cryptographic IOC in transit and trick a subscriber into using a wrong IOC or COA.

Checking for Substrings.

To extend the expressiveness of the cryptographic IOCs and do simple substring searching, we can use a sliding window. For example, Snort[2] signatures can contain rules as `content=abc ∧ offset=4 ∧ depth=5`, meaning that the rule is matched if the payload contains the string 'abc' starting from the fourth, fifth, or sixth byte. If we allow the offset, depth, and content *length* to be revealed in the cryptographic IOC, it is possible to rewrite the rule in DNF so that it only contains simple equality matches. For example, the rule mentioned above can be rewritten to `content[4-6] = abc ∨ content[5-7] = abc ∨ content[6-8] = abc`.

Traitor Tracing.

To impede users from sharing cryptographic IOCs with other parties, the source can issue different cryptographic IOCs to each subscriber and store the cryptographic IOCs together with the information to whom it was issued. Now, if the source discovers an illegitimately shared cryptographic IOC, it can easily look up to which subscriber it gave the leaked key.

Instead of storing *all* issued IOCs, it is more efficient to include a unique subscriber identifier as a feature in every IOC's rule, e.g., `subscriberID = 1082 ∧ destPort = 80`, and only store the plaintext IOCs together with their salts. Upon discovery of a leaked cryptographic IOC, the source now matches the leaked IOC with the feature values of the plaintext IOCs while enumerating over the different subscriber identifiers. Once it found a match for a specific identifier, it knows that the subscriber with this identifier leaked the cryptographic IOC.

3.2 Private Reporting of Sightings

To privately report back the number of sightings to the source, we apply a construction for privacy-preserving aggregation by Shi et al. [11]. We informally explain their construction applied to our setting of the reporting of sightings. However, we do not add noise to the data as proposed in the original work. The construction consists of three algorithms: Setup, Encrypt, and AggregateDecrypt. After running Setup, each subscriber can encrypt their number of sightings of an IOC over a specific timespan, e.g., by indicating the number of the week. The source collects the encrypted messages for the same IOC identifier and timespan from every subscriber. Using AggregateDecrypt it learns the total number of sightings for the IOC over that timespan.

In Setup the source sets the public parameters by choosing a group $\mathbb{G}$ of prime order p, a generator $g \in \mathbb{G}$, and a cryptographic hash function $H\colon \{0,1\}^* \to \mathbb{G}$. Furthermore, each subscriber i picks its own random encryption key $\mathrm{SK}_i \in \mathbb{Z}_p$. The source starts an MPC protocol with the subscribers to compute the sum of all encryption keys without learning the individual keys, e.g., using [5]. Note that we only need to use MPC once and only during the setup. The source sets its aggregation key $\mathrm{AK} = -\sum_i \mathrm{SK}_i$.

After the setup, the subscribers can encrypt their number of sightings n of an IOC with identifier ID over the timespan t as

$$\mathrm{CT}_{i,\mathrm{ID},t} \leftarrow g^n H(\mathrm{ID} \,\|\, t)^{\mathrm{SK}_i}.$$

The source can learn the aggregated value of the number of sightings by collecting the ciphertexts $\mathrm{CT}_{i,\mathrm{ID},t}$ for all subscribers i and computing the discrete logarithm of

$$V \leftarrow H(\mathrm{ID} \,\|\, t)^{\mathrm{AK}} \prod_i \mathrm{CT}_{i,\mathrm{ID},t}.$$

The correctness of the scheme follows from

$$\begin{aligned} V &= H(\mathrm{ID} \,\|\, t)^{\mathrm{AK}} \cdot g^{\sum_i n} H(\mathrm{ID} \,\|\, t)^{\sum_i \mathrm{SK}_i} \\ &= H(\mathrm{ID} \,\|\, t)^{\mathrm{AK}+\sum_i \mathrm{SK}_i} \cdot g^{\sum_i n} = g^{\sum_i n}. \end{aligned}$$

We stress that it is an important security requirement that *all* subscribers participate in the protocol. If the source would be able to compute the sum of a subset of all subscribers, too much information about the individual values would be leaked. For example, if the source is able to compute the aggregate of all subscribers ($\sum_i n_i$) and of all but one subscribers ($\sum_{i \neq j} n_i$), it can compute the difference to learn the individual value (n_j) as well.

4. EVALUATION

To evaluate feasibility, we have implemented our proposed approach to private IOC sharing and reporting of sightings.

[2]See https://snort.org/.

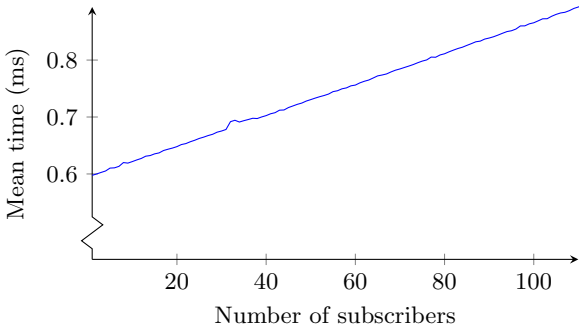

Figure 2: Mean time to compute AggregateDecrypt on curve P-256 for different number of subscribers.

4.1 Private IOC Sharing

We analyzed the efficiency of our approach by writing a simple Python script[3] that calls the network analysis framework Bro [9]. We use the HMAC-based key derivation function (HKDF) with SHA-256 to generate a symmetric key from the feature values. To slow down an attacker—and slowing down the matching algorithm for legitimate subscriber as well—one could use PBKDF2 with many iterations instead. To encrypt the IOC's COA, we use AES in randomized counter mode [3]. The matching algorithm is performed by decrypting the first block of ciphertext and checking whether this equals to a block of null bytes.

We ran our script to match data from the DARPA 1998 tcpdump to a plaintext and a cryptographic IOC. Bro outputs 10,723 records to be analyzed for this dataset. Matching all these records against a single cryptographic or plaintext rule incurs an average overhead of 0.40 seconds, or about 37 µs per analyzed record in an unoptimized implementation on an Intel Core i5 machine.

However, it is possible to optimize the implementation by keeping a cache of earlier computed combinations of observables at the subscriber's systems. The performance gain of using such a cache heavily depends on the combination of features and the subscriber's data. The more features involved, the more likely it is that we have a unique combination of feature values, making the cache less useful. However, for common features the performance significantly improves, e.g., caching the rule for the destination IP address and destination port number pair made matching the cryptographic rule essentially as efficient as matching the plaintext analogue.

4.2 Private Reporting of Sightings

Our proof-of-concept implementation[4] for the private reporting of sightings is written in Python using Charm[5]. The implementation can encrypt values in less than 0.6 milliseconds on an Intel Core i5 machine. To speed up the AggregateDecrypt algorithm, we use the baby-step, giant-step algorithm using hash tables for the computation of the discrete logarithm. This results in an efficient implementation that can also compute the sum in under a millisecond. In Figure 2 the mean running time of the aggregation algorithm is shown, evaluated for NIST's curve P-256.

[3]See https://github.com/CRIPTIM/private-IOC-sharing.
[4]See https://github.com/CRIPTIM/private-sightings.
[5]See http://charm-crypto.com/.

5. CONCLUSIONS AND FUTURE WORK

We propose the use of cryptographic techniques in the context of information sharing to limit the possibility of information misuse. While achieving full privacy guarantees seems intrinsically impossible in some settings, we can make it harder for an attacker to determine the private data while still allowing an honest party to use the data. By creating proof-of-concept implementations using simple, efficient cryptographic primitives, we show that using these cryptographic techniques incur little overhead.

Future work includes extending Bro or other intrusion detection systems (IDSs) to operate on cryptographic IOCs and evaluate them on live traffic data.

6. ACKNOWLEDGMENTS

This work was supported by the Netherlands Organisation for Scientific Research (NWO) in the context of the CRIPTIM project.

7. REFERENCES

[1] M. Allman, E. Blanton, V. Paxson, and S. Shenker. Fighting coordinated attackers with cross-organizational information sharing. In *HotNets*, pages 121–126, 2006.

[2] B. Applebaum, H. Ringberg, M. J. Freedman, M. Caesar, and J. Rexford. Collaborative, privacy-preserving data aggregation at scale. In M. J. Atallah and N. J. Hopper, editors, *PETS*, pages 56–74. Springer, 2010.

[3] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *FOCS*, pages 394–403. IEEE, Oct. 1997.

[4] M. Burkhart, M. Strasser, D. Many, and X. Dimitropoulos. SEPIA: Privacy-preserving aggregation of multi-domain network events and statistics. In *USENIX Security*, 2010.

[5] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. Tools for privacy preserving distributed data mining. *SIGKDD Explorations Newsletter*, 4(2):28–34, Dec. 2002.

[6] J. Freudiger, E. De Cristofaro, and A. Brito. Privacy-friendly collaboration for cyber threat mitigation. Tech. rep. Nov. 2014. arXiv:1403.2123 [cs.CR].

[7] P. Gross, J. Parekh, and G. Kaiser. Secure "selecticast" for collaborative intrusion detection systems. In *DEBS*, pages 50–55, 2004.

[8] P. Lincoln, P. Porras, and V. Shmatikov. Privacy-preserving sharing and correction of security alerts. In M. Blaze, editor, *USENIX Security*, pages 239–254, 2004.

[9] V. Paxson. Bro: A system for detecting network intruders in real-time. In A. D. Rubin, editor, *USENIX Security*, 1998.

[10] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy. BlindBox: Deep packet inspection over encrypted traffic. In *SIGCOMM*, pages 213–226. ACM, 2015.

[11] E. Shi, T. H. Chan, E. G. Rieffel, R. Chow, and D. Song. Privacy-preserving aggregation of time-series data. In *NDSS*. The Internet Society, 2011.