de Fuentes, J.M., González-Manzano L., Tapiador J., Peris-Lopez, P. (2017). PRACIS: Privacy-preserving and aggregatable cybersecurity information sharing. Computers & Security, vol. 69, pp. 127-141,
Available in https://doi.org/10.1016/j.cose.2016.12.011

PRACIS: Privacy-preserving and Aggregatable Cybersecurity Information Sharing

José M. de Fuentes, Lorena González-Manzano, Juan Tapiador, and Pedro Peris-Lopez

Department of Computer Science. Universidad Carlos III de Madrid
Avda. Universidad, 30, 28911 Leganés, Madrid, Spain
{jfuentes, lgmanzan, jestevez, pperis}@inf.uc3m.es

Abstract. Cooperative cyberdefense has been recognized as an essential strategy to fight against cyberattacks. Cybersecurity Information Sharing (CIS), especially about threats and incidents, is a key aspect in this regard. CIS provides members with an improved situational awareness to prepare for and respond to future cyberthreats. Privacy preservation is critical in this context, since organizations can be reluctant to share information otherwise. This is particularly critical when CIS is facilitated through an untrusted infrastructure provided by a third party (e.g., the cloud). Despite this, current data formats and protocols for CIS do not guarantee any form of privacy preservation to participants. In this paper we introduce PRACIS, a scheme for CIS networks that guarantees private data forwarding and aggregation. PRACIS leverages the well-known Structured Threat Information Expression (STIX) standard data format. Remarkably, PRACIS can be seamlessly integrated with existing STIX-based message brokering middleware such as publish-subscribe architectures. PRACIS achieves these goals by combining standard format-preserving and homomorphic encryption primitives. We discuss experimental results obtained with a prototype implementation developed for a subset of STIX. Results show that entities may create up to 689 incidents per minute, far beyond the estimated average of 81. Moreover, aggregation of 10^4 incidents can be carried out in just 2.1 seconds, and the transmission overhead is just 13.5 kbps. Overall, these results suggest that the costs incurred by PRACIS are easily affordable in real-world scenarios.

Keywords: cybersecurity information sharing; cyberthreat management; format preserving encryption; homomorphic encryption; cooperative cyberdefense

1 Introduction

The number and sophistication of cybersecurity incidents has increased substantially in the last years. According to a 2016 report by PricewaterhouseCoopers, around 59 million security incidents were identified in 2015 alone, which constitutes a 38% increase with respect to the previous year [51]. Moreover, these incidents are also growing in complexity, as it is the case of large-scale coordinated attacks [64].

In order to defend against this evolving type of threats, novel detection and protection mechanisms are being developed. Among them, Intrusion Detection Systems

(IDSs) have received extensive research attention [23]. One critical issue is that isolated IDSs are not always effective against coordinated attacks, since their traces may be spread across different domains [64]. Thus, collaboration between entities (the so called *cooperative cyberdefense*) is essential for properly detecting these attacks [28].

To enable cooperative cyberdefense, mechanisms for timely sharing actionable cybersecurity information (e.g. vulnerabilities, detection signatures, or indicators of compromise) are paramount [28], [15]. Over the past decade, MITRE Corporation and others have developed numerous languages, data formats, and standards to codify cybersecurity information [45]. Moreover, other approaches aim to facilitate automatic sharing, e.g., through protocols such as Trusted Automated eXchange of Indicator Information (TAXII) [46]. A recent survey by the European Union Agency for Network and Information Security (ENISA) provides an overview of existing standards and tools in this area [19].

Cooperative cyberdefense and, in particular, Cybersecurity Information Sharing (CIS), has been nurtured and encouraged by governments worldwide through a number of recent legal initiatives [57]. For example, in the US the CIS Act has recently proposed the creation of a government-managed structure to gather and distribute information about cybersecurity threats [60]. A similar effort, known as the CIS Partnership, is a joint industry-government initiative launched in 2013 in the UK to share cybersecurity information [10]. Additionally, in 2015 the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) published an international standard to provide guidance in the sensitive information exchange. This standard also serves for the implementation of information security management within information sharing communities [31].

CIS has also received much attention from the research community under different perspectives. Research interest has range from pure technical questions (such as the infrastructure needed for this exchange) to socio-economical issues. For instance, Gal-Or and Ghose [22] identified clear economical benefits for information sharing as a result of better security prevention and increased reputation. Despite this, organizations are not inclined to share cybersecurity intelligence (including ongoing or past cyberincidents) neither with governments nor with other partners or competitors. Reasons include the lack of trust in the sharing infrastructure, particularly if it is run by a potential competitor or adversary, and the way sharing is carried out. For example, in networks in which the government is involved, companies prefer to remain anonymous in case of incidents that uncover infringement of rules, i.e., leakage of unprotected personal data. To address this issue, previous works have proposed data sharing schemes based on encrypting exchanged data [2, 38, 42], working with aggregated messages [35, 58], or guaranteeing some form of sender anonymity [16, 39]. While interesting, the main drawback of these schemes is that they are not compatible with currently deployed CIS infrastructures, including widely adopted standards to exchange cybersecurity information.

Contributions. In this paper we introduce PRACIS (PRivacy-preserving and Aggregatable Cybersecurity Information Sharing), a scheme that provides privacy-preserving data forwarding and aggregation in a data sharing network. PRACIS is intended for semi-trusted (i.e., honest-but-curious) message-oriented middlewares, which are one

of the most common architectures for CIS. Thus, PRACIS guarantees that the sharing infrastructure (i.e., the message broker) can perform its store-and-forward functions without learning anything relevant about the messages in the process. In particular, the broker cannot link the identity of the reporting entity and the type of reported incident. We achieve this by adapting existing format-preserving encryption techniques to cybersecurity information messages, particularly the Structured Threat Information Expression (STIX) standard data format [4], thus allowing the message broker to privately forward messages. Additionally, our scheme also leverages homomorphic encryption to provide some network members with simple statistics about reported information (e.g., global or per-type counts and averages) in a privacy-preserving way. All in all, the main novelty PRACIS does not rely on innovative cryptographic primitives, but in the combination of existing ones to achieve the pursued goals in already-existing STIX-based CIS infrastructures.

We have developed a proof-of-concept implementation of PRACIS to assess the overhead imposed by the encryption. The average time to build a STIX message in which three fields are sensitive, is 87.4 ms while its decryption and verification requires 105 ms. Both figures are very low and appropriate for information sharing in nearly real time. In terms of data size, our scheme introduces non-negligible yet affordable overhead in each message field. Depending on factors such as the field length, the frequency of message delivery, or the number of members in the sharing network, such overhead ranges between 2 and less than 1300 bytes. In summary, in this work we make the following contributions:

- We introduce the idea of using privacy-preserving data forwarding and aggregation for CIS networks. We argue that this is an essential service for this communities to succeed, as otherwise partners might not trust the infrastructure nor provide other members with their data.
- We describe PRACIS, a scheme that applies format-preserving and homomorphic encryption techniques to the STIX data format to achieve these goals. Our experimental results suggest that PRACIS introduces an affordable overhead in today's applications.
- We make freely available a prototype implementation of PRACIS to foster further research in this area.

Organization. The rest of this paper is organized as follows. Section 2 provides some background concepts on CIS and the cryptographic techniques used in this work. In Section 3 we introduce the system and adversarial models, along with the protocol goals. Section 4 describes our proposal in detail, and Section 5 provides an evaluation based on experimental results. An overview of related work in this area is provided in Section 6, and Section 7 concludes the paper.

2 Background

We next provide some necessary background on the main concepts used in our work: the STIX format, homomorphic cryptography and format-preserving encryption.

Fig. 1: An example of a STIX message [4].

2.1 The STIX format

In the last years, much effort has been devoted to develop common formats for the exchange and processing of actionable security information, such as vulnerabilities, detection signatures, or indicators of compromise, among others. As of today, the number of such formats is relatively high and some of them have become *de facto* standards. The interested reader can find a comprehensive survey in a 2014 report by ENISA [19]. One of the most widely used of such formats is the Structured Threat Information Expression (STIX) [4], which was designed to specify, characterize, and communicate cyber threat intelligence information. STIX is an XML-like structured language in which issues related to threats (e.g., malware descriptions or indicators of compromise) can be expressed. For the interest of this proposal, Figure 1 shows an example of a STIX incident. Thus, aspects such as the affected assets, the nature of the threat, start and recovery times, or the perceived risk can be formalized.

2.2 Format-preserving encryption

Format-preserving encryption (FPE) is a cryptographic technique in which the output of the encryption operation (i.e., the ciphertext) has the same *format* than the input (i.e., the plaintext) [5]. The idea has been traditionally motivated by the problems associated with integrating encrypted data into some legacy applications that expect data items in a particular format. One prominent example is social security or credit card numbers, which after encryption may no longer have the required length and may include alphanumeric or special characters. In general, the notion of "format" in FPE can be extended to almost any structured data item, but typically only finite domains are supported.

One of the simplest examples of FPE is an n-bit block cipher, which is an FPE on the set $\{0, \ldots, 2^{n-1}\}$. In a more general setting, Black and Rogaway [8] provided the first provable-security approach to construct a block cipher with an arbitrary domain $\mathcal{X}$, though their solution focused on $\mathcal{X} = \mathbb{Z}_n$, i.e. the integers $\{0, 1, \ldots, n-1\}$. Bellare *et al.* [5] later provided a more general construction called the *rank-then-encipher* approach. This assumes that the format space $\mathcal{X}$ is a collection of a finite number of domains called *slices*, i.e., $\mathcal{X} = \cup_N \mathcal{X}_N$. The points in each slice can be arbitrarily numbered, say $\mathcal{X}_N = \{X_0, \ldots, X_{n-1}\}$, with $|\mathcal{X}_N| = n$. To encrypt X_i, the rank-then-encipher strategy first finds the index i of X in the enumeration of $\mathcal{X}_N$; then encrypts i to j using an integer-to-integer cipher; and finally returns X_j as the encryption of X. This construction is based on an integer FPE cipher $\mathsf{Enc} : \mathcal{K} \times \mathbb{Z}_n \to \mathbb{Z}_n$ that can encrypt on $\mathbb{Z}_n$ for an arbitrary n, and a ranking function rank that maps each (N, X), with $X \in \mathcal{X}_N$, to an element of $\mathbb{Z}_n$, where $\mathsf{rank}(N, \cdot) : \mathcal{X}_N \to \mathbb{Z}_n$ is a bijection for all N. In [5] it is shown how to build ranking functions for domains that are regular languages.

2.3 Data aggregation using homomorphic cryptography

A homomorphic encryption scheme is a cryptographic primitive that produces ciphertexts with two properties: (i) only parties knowing the key can retrieve the original plaintext; and (ii) it is possible to perform operations (e.g., addition [3], multiplications [34], or counts [37]) over ciphertexts that result in a desired transformation of the original plaintexts. One representative example of a cipher with homomorphic properties is the Paillier cryptosystem [48]. This scheme is a public-key cryptosystem whose encryption function is additively homomorphic. Thus, the product of two ciphertexts decrypts to the sum of their associated plaintexts, i.e.,

$$D(E(m_1)E(m_2) \mod n^2) = (m_1 + m_2) \mod n,$$

where $E(m_i)$ is the encryption of plaintext m_i, $D(\cdot)$ is the decryption function, and n is the Paillier modulus. This property can be used to compute some basic statistics of a set of plaintexts, such as the sum and the mean value, using only the ciphertexts. In fact, the prototype implementation in this paper makes use of this cryptosystem. However, it must be noted that any other mechanism providing homomorphic addition (e.g., Benaloh's [7]) could be applied as well.

3 System Model and Goals

In this section, we first discuss the main elements of the CIS network model, including its entities, information and privacy models (Section 3.1). We next describe the adversarial model (Section 3.2) and the goals of our proposal (Section 3.3). The notation used throughout the paper is presented in Table 1.

Table 1: Notation

3.1 Cyber security information sharing network

Information model. Nodes in an information sharing network produce messages that encode relevant intelligence about a particular cyber threat. In this work, our focus is on messages reporting cyber incidents. One remarkable aspect is that PRACIS is format-agnostic. This means that it could be adapted to any incident format, though in the following we adopt the STIX format [4]. Each message, which will be an incident I_t, includes at least the following data items: an identifier; a definition of when, where and what happened; the identity of the victim; the identity of the entity reporting the incident; an assessment of the incident's impact; and the confidence of the publisher on the reported information. As discussed later, extending the protocol to support other STIX objects is trivial.

Fig. 2: Overview of PRACIS

Network model. We assume a distributed information sharing network implemented through an standard message-oriented middleware such as a publish-subscribe architecture, see Figure 2. This is one of the most used alternatives for CIS (the other being a peer-to-peer network among partners) and is fully compatible with standard sharing protocols such as TAXII [46]. Publishers D_i post messages to a message broker AF, which normally implements a store-and-forward function to deliver messages to subscribers IS_i. For simplicity, subscriptions are topic-based, so nodes subscribe to messages tagged as belonging to a particular topic. The delivering mechanism can implement additional restrictions based, for example, on security attributes of the message (e.g., a classification level) and the subscriber (e.g., its clearance). We assume a distinguished subscriber StS called the *statistics subscriber*. It is an entity interested in receiving from the message broker statistics (e.g., averages or counts) of the incidents registered in a given time span. The realization of any of these entities could be done either in a standalone machine or in a distributed fashion (e.g., cloud computing infrastructure). However, this network model is technology-agnostic and valid regardless of the actual implementation.

Information privacy model. The said data items in an incident I_t can be divided into *sensitive* and *non-sensitive* elements. Sensitive values are the type of asset, the effect and the confidence. We believe that these three attributes have to be hidden from the adversary since they leak information from the type of incident. For example, a database asset with a data loss effect with high confidence reveals a breach into the corporate storage systems. Although there are other attributes that convey private information, the described scenario does not make these fields sensitive. For example, the name of the victim is privacy-critical in a general setting, but in the considered scenario the adversary (i.e., AF) already knows this value beforehand. This could be the case if, for example, a different communication port is used in AF for each entity's incidents. Thus, the remaining data items are considered non-sensitive herein. Of course, this definition could be easily adapted for other CIS scenarios.

3.2 Adversarial model

Most nodes (i.e., publishers D_i and subscribers IS_i, including the statistics subscriber StS) in the CIS network are assumed to be honest. This means that they trust each other [18], they do not attempt to attack the system, and that they cooperate by publishing incidents shortly after their occurrence.

Nodes are distrustful of the sharing infrastructure, since this can be deployed in a third-party server and compromised by an attacker. We assume that the message broker AF is considered a semi-trusted (i.e., *honest-but-curious*) entity. It will behave correctly in its storing, aggregation, and forwarding functions. However, AF will eavesdrop on received messages and will be interested in learning the types of incidents occurred in each entity. This is similar to adversarial models adopted in other schemes in which

storage, aggregation, and forwarding is delegated to a semi-trusted entity (see, e.g., [20, 26, 55]).

We assume that nodes connect to the message broker AF using a secure channel such as TLS. This is a reasonable assumption for sharing protocols such as TAXII [46], which is essentially XML over HTTP.

3.3 Goals

Considering the system and adversarial models described above, the goals of our scheme are:

- **Privacy-preserving message forwarding**. Subscribers IS_i must be able to choose the type of incidents they are interested in. The message broker AF must be able to forward them incidents without learning any sensitive data about the incident itself (recall Section 3.1).
- **Privacy-preserving data aggregation**. AF must be able to aggregate (i.e., tally) incidents per type (as requested by StS) without learning the result of the operation. Furthermore, StS must be able to verify the correctness of the result.

4 PRACIS: Privacy-preserving and Aggregatable Sharing

This Section describes each phase of our proposal, namely the setup phase (Section 4.1), how incidents are prepared and delivered (Section 4.2), how they are processed and aggregated (Section 4.3), how they are finally decrypted (Section 4.4), and how aggregated incidents are verified (Section 4.5). A formal description of all algorithms in PRACIS is provided in Appendix A.

4.1 Setup

In the setup phase, all entities are provided with the appropriate cryptographic material to participate in the information sharing network. This phase can take place during one of the face-to-face meeting suggested by ENISA [18] for trust building. Specifically, key exchanges are specified in Algorithm 1 and described next.

- Each publisher D_i shares two keys with all incident subscribers IS_i authorized to receive its incidents. The first one is a Format-Preserving Encryption (FPE) key ($K_{FPE}(D_i)$) to encrypt security incidents. The second one is a symmetric key ($K_{s(D_i)}(t_i)$) per type of incident t_i to allow incident forwarding (Algorithm 1, lines 1-10).
- StS creates a homomorphic key (K_H) and shares it with all D_i. This key allows computing statistics over the shared incidents. StS also creates a random number per D_i, referred to as r_{D_i}. This number enables verifying the incident aggregation. Furthermore, StS specifies the maximum size in bits ($N_{b_{rnd}}$) of the addition of the said random numbers. This value ensures that homomorphic additions do not overflow (lines 11-19).

Fig. 3: Example of STIX incident format to apply PRACIS.

– All D_i, AF and IS_i share a HMAC key (K_{HMAC}) to perform incident integrity checking (lines 20-28).

Once cryptographic materials have been prepared, the last step of the setup phase is incident subscription, in which every IS_i subscribe to incidents of its interest (Algorithm 3). For this purpose, each IS_i sends AF chosen set of incident types t_i symmetrically encrypted using the key $K_{s(D_i)}(t_i)$ already introduced. Thanks to encryption, AF is unaware of the particular incident that IS_i is subscribing to.

4.2 Preparation and delivery of security incidents

Once an incident takes place in a given D_i, it is shared with interested IS_i (Algorithm 4). In order to convey all fields of an incident, a simple STIX incident structure is adopted (Figure 3). It consists of 10 fields, namely, *Title*, *Description*, *Affected_Asset:Type*, *Effect*, *Confidence:Value*, *Initial_Compromise*, *Incident_Discovery*, *Restoration_Achieved*, *Incident_Reported* and *Victim:name*, and 3 ID tags involved in "Incident", "Description" and "Victim" (*Incident:id*, *Description:id* and *Victim:id* respectively). This structure contains the information elements contained in the abstract description of I_t (recall Section 3.1). Instead of storing the actual values in each field, our protocol pre-processes them to: (1) only allow authorized IS_i to access the actual content; and (2) enable incident aggregation so that StS can receive incident statistics. With respect to the first goal, the three sensitive attributes (recall Section 3.1), that is, the *Affected_Asset:Type*, *Effect* and *Confidence:Value* fields are encrypted applying FPE. Except for the cases discussed next, the remaining fields are sent in plaintext[1] as long as they do not convey any valuable information for the considered adversary. The second goal requires the aggregation by AF of values in the *Incident:id* field.

There are two additional fields which are pre-processed, namely *Description:id* and *Victim:id*. The first one is prepared to enable matching the event type to IS_i interests. It must be noted that AF is not aware of the actual type of package, but it should be able to forward it to appropriate IS_i. The idea behind is to store in this field the same value as the one sent by ISs to subscribe (recall Section 4.1). On the other hand, "Victim:id" is used to enable file integrity checking of the remaining contents. For this purpose, it stores the result of HMACing all remaining STIX fields, using K_{HMAC} as key. One important remark is that the election of fields for incident aggregation, subscription and integrity checking could be different than the one proposed herein. Our decision is based on the fact that these fields contain identifiers, which can be freely adopted by D_i. Therefore, these fields are suitable to contain random-like values such as the results of encryption or HMAC operations. However, different decisions could be taken in this regard, such as splitting these values among different fields in a single incident or even among several incidents, but this does not affect our proposal.

[1] They are sent without an explicit encryption by the sender. However, transport mechanisms such as TLS could be applied, thus providing encryption by default.

Preparing Incident:id. The *Incident:id* field is intended to allow the aggregation of incidents by AF. The result of aggregating this piece of information will be sent to StS. Therefore, it must be possible to gather this field from all incidents I_t and operate them to obtain their aggregation, that is, the amount of incidents received per type t in a given time frame.

In PRACIS, aggregation is performed by adding the values of *Incident:id* in all I_{t_i}. For this purpose, it is necessary to prepare a data structure to store a maximum of I_{max} incidents per type t_i. This data structure must allocate a set of bits N_{bt} for each type t_i to represent I_{max}. Thus, considering nt types of incident, this structure needs $N_{b_{total}}$ bits to represent them all (Equation 1).

$$N_{b_{total}} = nt \cdot N_{bt} = nt \cdot \lceil \log_2(I_{max}) \rceil \text{ bits.} \tag{1}$$

Given an incident I_{t_i} of type t_i, the aforementioned data structure is prepared by setting to 1 least significant bit of the set of N_{b_t} bits for this type. The remaining bits are set to zero. In order to conceal the type of incident to third parties, the said data structure is encrypted using the Paillier cryptosystem.

To provide aggregation verification, the random number r_{D_i} is inserted in this structure. The rationale behind is that the aggregation (i.e. addition) of these numbers enables StS to conclude if the aggregation is correct, with a low probability of identifying the particular D_i. The total amount of bits needed for this purpose is given by $N_{b_{AV}}$ (Equation 2). Thus, $N_{b_{AV}}$ is calculated as the number of bits to express I_{max} multiplied by the maximum bit length of random numbers ($N_{b_{rnd}}$) as specified by StS:

$$N_{b_{AV}} = \lceil \log_2(I_{max} \cdot (2^{N_{b_{rnd}}} - 1)) \rceil. \tag{2}$$

Besides, $N_{b_{rnd}}$ should be established considering $N_{b_{rnd}} > \log_2(|D_i|)$ to reduce the possibilities of choosing the same random number for multiple D_i. Considering the previous equations, the total length of the cleartext of *Incident:id* is given by Equation 3:

$$N_{b_{Incident:id}} = N_{b_{total}} + N_{b_{AV}}. \tag{3}$$

Example: setting Incident:id. To illustrate how an incident identifier is prepared, consider an scenario with $|D_i| = 10$ and 7 types of incidents, numbered from 0 to 6. Assume that an incident of *Malicious code* (e.g., $t = 4$) takes place in D_1 whose random number is r_{D_1}=49 (with $N_{b_{rnd}}$=6). AF aggregates up to I_{max}=100 events. Considering these parameters, $N_{b_{total}} = 7 \times \lceil \log_2(100) \rceil$= 49, and $N_{b_{AV}} = \lceil \log_2(100 \times (2^6 - 1)) \rceil$ = 13. Thus, Incident:id is formed by $N_{b_{Incident:id}} = 62$ bits as follows:

$$\begin{aligned} Incident : id = \text{HE}(K_H, &0000000\ 0000000\ 0000000\ 0000001\ 0000000 \\ &0000000\ 0000000\ 0000000110001) \end{aligned} \tag{4}$$

where $\text{HE}(k, x)$ represents homomorphic encryption of x with key k.

4.3 Forwarding and aggregation of incidents

Once incidents are sent by D_i, AF processes each message (Algorithm 2). Firstly, it checks the message integrity by comparing the HMAC of all fields against the value

stored in the *Victim:id* field. If the integrity remains, AF finds ISs subscribed to this type of event. This is done by comparing their subscription packages (sent during setup) against the value of *Description:Id*. If both elements match, I_{t_i} is forwarded to IS_i. On the other hand, after a maximum of I_{max} authentic security incidents have been received by AF, it aggregates them. For this purpose, the values of the packet fields *Incident:id* are combined using homomorphic addition. Such homomorphic addition is dependent upon the cryptosystem at stake – in the Paillier crytosystem it consists of the multiplication of *Incident:id* (recall Section 2.3). The number of incidents received per D_i is appended to the previous result, and this structure is sent to StS. Note that the attachment of the number of incidents is related to the aggregation verification protocol presented in Section 4.5.

Example: aggregating Incident:id values. Using the numerical example presented above, consider now that D_2 and D_3 have reported a t_4 (*Malicious Code*) incident and a t_5 (e.g. *Improper Usage*) incident, respectively. Their random numbers are r_{D_2}=20 and r_{D_3}=21, respectively. The value of each *Incident:id* field, as well as the result of their homomorphic aggregation, is shown in Table 2. This is the information to be sent along with the indication of the amount of incidents reported by D_i. For the sake of clarity, the underlying, non-encrypted values are shown. However, it must be recalled that sent values are the encrypted versions of these data structures.

Table 2: Example of *Incident:id* aggregation.

4.4 Decrypting security incidents

Upon reception of an incident forwarded by AF, each IS_i needs to decrypt their actual content (Algorithm 6). First of all, the message integrity is verified computing its HMAC using K_{HMAC} (lines 1-5). If the result is equivalent to "Victim:id", the encrypted fields are decrypted using format-preserving decryption with $K_{FPE}(D_i)$ being D_i identified in "Victim:Name".

4.5 Aggregation verification

Once the aggregated incident types are received by StS, it decodes the received message contents and verifies its correctness. Regarding the first issue, StS simply needs to identify how many bits are devoted to each incident type and it proceeds accordingly. Thus, going back to the example used above, it is straightforward for StS to identify that there are two *Malicious Code* and one *Improper Usage* incidents, as well as checking that the received addition of random numbers (referred to as r_{sum}) is 90 (that is, 20+21+49).

With respect to data correctness, StS verifies that aggregated data is unpolluted. For this purpose the amount of incidents per D_i attached to aggregated data comes

into play. Algorithm 5 shows the steps that StS needs to carry out. Particularly, StS first gathers the status of the (initially random) counter value r_{D_i} of each D_i (lines 1-2). Afterwards, StS computes the sum based on received amount of incidents per D_j ($|I(D_i)|$) and each chosen r_{D_i} (lines 3-6). This is computed as

$$\sum_{\forall D_i}(\sum_{r_{D_i}}^{D_i+(|I(D_i)|-1)} (r_{D_i})), \tag{5}$$

and if the result matches r_{sum}, the claimed amount of incidents received per entity is correct. StS then accepts the data and updates D_i counters accordingly; otherwise, data is discarded.

Example: Aggregated data verification. To show how aggregated data is verified, let us follow the previous example. Let us consider that D_2 and D_3 report one additional incident each one. Thus, recall that the initial counters were r_{D_1}=49, r_{D_2}=20 and r_{D_3}=21, and that $N_{b_{rnd}} = 13$. When StS receives r_{sum}=133, it verifies the aggregation computing Eq. 6. Given that the result is equal to r_{sum}, the aggregation is correct:

$$\begin{gathered}(\sum_{r_{D_1}}^{r_{D_1}+(1-1)} (r_{D_1}) + \sum_{r_{D_2}}^{r_{D_2}+(2-1)} (r_{D_2}) + \sum_{r_{D_3}}^{r_{D_3}+(2-1)} (r_{D_3})) \\ \bmod (2^{N_{b_{rnd}}} - 1) = (49 + 20 + 21 + 21 + 22) \bmod 8191 = 133\end{gathered} \tag{6}$$

5 Evaluation

We next discuss how our proposal meets the established goals (Section 5.1) and then report experimental results obtained with a prototype implementation (Section 5.2). To facilitate the reproducibility of our results and foster further research in this area, we make our implementation freely available[2].

5.1 Goals

Privacy-preserving message forwarding. Each incident I_t includes its type t in encrypted form within the *Description:id* field. Then, AF is able to match incidents with interested subscribers without gaining any information about the actual type. This allows a given IS_i to receive only the I_t of interest. Incidents are encrypted to prevent AF from identifying the actual attack types suffered by each D_i. Thanks to FPE, three pieces of information from the incident are hidden for the attacker. Particularly, the type of asset involved in the incident, the type of incident and the confidence of the reported information are encrypted using FPE. The remaining fields of the incident are not assumed to give any information for the attacker or, in other cases, are already known (e.g., the reporting time). In any case, if required other fields can be FPEed too.

[2] https://github.com/jmdefuentes/SPCIS.

PRACIS also inserts a HMAC in the *Victim:id* field to enable that IS_i verifies the correct transmission of incidents and to guarantee that they come from valid publishers. Similarly, the trusted key exchange performed in the setup phase prevents the introduction of fake IS_i.

Privacy-preserving aggregation. Homomorphic encryption is used for aggregation. In this way StS knows existing incidents of each type without identifying their sources. One important remark is that StS has some probability of reidentification of incidents. In other words, there is a chance for StS to link which types of incidents every D_i has reported. Table 3 illustrates this issue. StS already knows the marginals of that table, since it knows the total per t_i and per D_i.

The hardness of reidentification is to find out which are the values for the cells in italics in Table 3. Thus, the probability of discovering the distribution of event types among publishers is thus the same of finding the right table that satisfies the said marginals. Indeed, the probability $P_{reident}$ of this event is given by Equation 7 in which $|T|$ is the total amount of tables satisfying the given marginals.

$$P_{reident} = 1/|T|. \tag{7}$$

The value of $|T|$ can be calculated following the method proposed by Gail and Mantel [44]. Intuitively, this amount grows with the size of the table. For example, considering Table 3, the total amount of tables of the same dimensions is 40,500, which leads to $P_{reident} = 2.47e^{-5}$. Even if this probability is significantly low, bigger tables (for example, 4 D_i and 6 t_i) may lead to $|T| = 68e^6$ [13], which cause $P_{reident}$ to be negligible.

Apart from privacy preservation, StS is still able to verify the proper aggregation of incidents. This verification is based on a sum of random numbers included in *Incident:id* (recall Section 4.5). If the sum received by StS matches with the computed one, the aggregation will be considered correct.

Table 3: Example of contingency table of event types and deliverers. Cells in bold are the values known to StS after PRACIS, whereas values in italics are the ones to be discovered in the reidentification attack.

5.2 Performance analysis

We have considered two factors to assess the performance of our scheme. First, we have studied the expected computation time per entity. The goal here is to determine if the time taken is acceptable considering the pace at which incidents might occur in the real world. On the other hand, we analyze the overhead incurred by exchanging messages. As PRACIS modifies some fields of a classical STIX package, it is important to determine whether the introduced information produces an excessive overhead that may result in network issues.

Experimental settings. An implementation of PRACIS has been developed to assess the performance of the approach. It has been implemented in Python 3 using various libraries, in particular `libFTE`[3] to implement FPE; `Charm`[4] to implement the Paillier cryptosystem; and the Python STIX library[5] for managing STIX packets. For FPE, we adopt the ranking-then-encryption algorithm provided by Luchaup *et al.* [41]. AES-256 is applied for symmetric encryption and 1024-bit keys are used for the Paillier cryptosystem. We consider 33 incident types, 5 levels of confidence, 34 affected assets and 14 effects, these being values taken from STIX. More values for each element could be added to our prototype, although the expected impact in performance is negligible as the complexity of involved operations is not affected by this factor.

The results shown in the remaining of this section have been obtained by averaging 50 executions on a single machine 4-core Intel(R) Xeon(R) CPU E5645 2.40GHz with 1GB of RAM. Note that these settings are far from representing a realistic CIS network. For example, AF could be implemented in a distributed infrastructure such as a cloud computing platform. This leads to significantly better performance figures in what comes to computation times. However, these settings may serve as a worst-case scenario in which the CIS network runs with limited resources.

Computation time. We first study the time taken by D_i to prepare a STIX message, the time taken by IS_i to decrypt and verify the integrity of received STIX messages, and the time taken by AF to perform the aggregation of STIX messages. Creating a STIX message involves different tasks: the setup phase to create all applied cryptographic parameters; the homomorphic encryption of the incident id; FPEing the affected asset, the effect and the confidence; the HMAC of all fields for integrity purposes; and the creation of the STIX message. Table 4 presents results of our experiments. According to this, format-preserving encryption and decryption operations are the most computationally demanding. On the other side, homomorphic encryption and HMACs are extremely fast operations. The setup phase needs less than a second in computation time.

To contextualize the implications of these figures, the 2015 Global State of Information Security Survey states than an average of 81 incidents per minute might be detected by an entity [51]. Our experimental results suggest that a party could create up to 689 incidents per minute. Another important finding is the time required to aggregate packets by AF is extremely low, needing around 2.1 seconds to group 10^4 messages. Considering the said average of incidents per minute, PRACIS would be feasible even for big information sharing networks. For example, assuming 100 D_i, such an amount of messages would allow to aggregate incidents produced in an interval of $10^4/(100 \cdot 81) = 1.23$ minutes.

Table 4: Creation, decryption, verification and aggregation of STIX messages (average)

[3] https://libfte.org

[4] http://python-paillier.readthedocs.org/en/latest/alternatives.html

[5] https://github.com/STIXProject/python-stix

In order to study the scalability of the incident creation and aggregation, different scenarios may be devised depending on the amount of incidents per minute and the amount of D_i at stake. Table 5 summarizes the analysis. For the first issue, values from 40 (i.e., half the average) to 1280 (i.e., roughly 16 times the average, which is a reasonable limit in the short term) are considered. Concerning the amount of D_i, the analysis is carried out considering networks that ranges from small size (i.e., 10 D_i) to very big ones (i.e., 800 D_i).

Based on the aforementioned settings, incident creation time does not depend on the amount of D_i since all of them may create incidents in parallel. Thus, it can be seen that PRACIS can cope with up to 640 incidents per minute. With respect to aggregation time, the amount of D_i is critical – aggregation is centralized on AF, which will receive all events. The greater the amount of D_i, the bigger amount of events to aggregate for a given rate of incidents per minute. In particular, PRACIS can work with up to 800 D_i if the rate of incidents is up to 160 per minute. Beyond that limit, two issues may be noticed. On the one hand, the aggregation takes more than one minute, which means that a growing queue would appear. In other words, aggregating the incidents produced in one minute would take more than that minute. This makes the system potentially unusable. On the other hand, the sum of both creation and aggregation is bigger than one minute under some settings (e.g., 640 incidents per minute and 100 D_i). Again, this is not desirable since it threats immediacy and requires memory resources on AF to manage an incident queue.

Despite this subset of settings, Table 5 shows that PRACIS can work under most configurations, supporting big incident rates with reasonable amounts of D_i. Conversely, PRACIS can also work with smaller incident rates in bigger CIS networks.

Table 5: Incident creation and aggregation scalability analysis

Message overhead. Message overhead refers to the amount of extra bytes that are inserted into STIX incident packets to achieve the desired goals. This overhead is caused by four issues: (1) the homomorphically encrypted value stored in *Incident:id*; (2) the HMAC value stored in *Victim:id*; (3) the encrypted type stored in *Description:id*; and (4) the values concealed with FPE regarding confidence, effect and asset. The remaining data of STIX messages do not cause overhead because its size is not altered from its creation to its reception.

Regarding the first three items, note that no reference value can be considered. STIX data model for incidents[6] does not impose any minimum or maximum lengths for these fields. Nevertheless, it is still possible to carry out a worst-case analysis by comparing the length of PRACIS fields against the smallest length of these fields (i.e., 1 byte). In our experiments, *Victim:id* is 28 bytes long and *Description:id*is 16 bytes long. Therefore, their overhead is 27 and 15 bytes, respectively.

[6] http://stixproject.github.io/data-model/1.2/incident/IncidentType/

As for *Incident:id*, its size is determined by the amount of D_i at stake and the aggregation interval (i.e., how frequent AF has to compute the aggregation). Table 6 shows the analysis for 10, 100 and 1000 D_i and 1, 5 and 10 minutes. Using said average of incidents per minute and assuming $N_{b_{rand}} = 20$, the maximum size for Incident:id is 88 bytes. In order to have the maximum encryption size, we adopt the highest value for *Incident:id* (i.e. of $2^{88} - 1$). According to our experiments, the encrypted value of *Incident:id* is 1136 bytes long. Therefore, a maximum overhead of 1135 bytes is introduced. Both figures are reasonable in today's storage capabilities and networking speeds.

Table 6: Overhead in *Incident:id* size after encryption depending on D_i and aggregation intervals

With respect to the confidence, effect, and asset fields, there is an implicit reference value. By applying FPE results to one value of the set, it is possible to assess the maximum overhead. For example, the values of confidence are *low*, *medium*, *high*, *none*, and *unknown*. Therefore, if the actual value is *low* and after FPE it turns into *unknown*, an overhead of 4 bytes is introduced. Following this approach, Table 7 presents the minimum, average, and maximum amount of bytes of the three fields, as well as the involved overhead. The results show that the maximum overhead for all three fields is 76 bytes with an average of 29 bytes. For the sake of generality, in these figures we omit some implementation-dependent extra bytes that are needed for our prototype, as a consequence of the chosen libraries.

Putting all values together, the worst-case overhead is 1253 bytes. Using the average of 81 incidents per minute discussed above, the imposed overhead for each entity is 101,493 bytes per minute; or, equivalently, 13,533 bits per second, which is negligible for existing transmission technologies.

Table 7: Overhead of FPE fields (bytes)

6 Related Work

In the cybersecurity field information sharing is a requirement in a threat intelligence management system [9]. For instance, an initiative has been developed to facilitate CIS between the UK industry and the government [47]. Indeed, CIS is a challenging issue which has been studied from different angles. For comparison purposes we distinguish approaches that analyze the need of information sharing, present a theoretical approach to information sharing or propose a specific scheme towards information sharing. The comparison study is presented in Table 8. Concerning the first group, [49] studies the problems and challenges of information sharing in coalitions formed with international

cooperation. [63] discusses the necessity of information sharing and a guidance of the type of information to be shared. [25] analyses, from an economic point of view based on the investment, how sharing cybersecurity related information among firms has the potential to offset the tendency by firms to defer much of their cybersecurity investments until a cybersecurity breach occurs. Finally, works such as [21, 24, 32, 59] study the interests, possibilities and implications of cybersecurity information sharing and to do so, they use, as many other works, game theory.

Several proposals have focused on developing theoretical foundations for information sharing. A model to share classified security information between organizations with the lowest possible risks is proposed in [33]. This is a theoretical model that relies on the use of STIX and TAXII. [14] presents Cyber Security Data Exchange and Collaboration Infrastructure (CDXI) capability, a knowledge management tool for the cyber security domain. A set of high level requirements are introduced. One last remarkable approach in this regard is SKALD [61], an architecture to create systems that can perform feature extraction at a scale and provide a robust platform for analytic collaboration and data sharing. It provides the infrastructure, from a theoretical point of view, needed to allow industry peers to perform analyses across collective knowledge while protecting sensitive data.

Conversely, several works propose specific schemes to share cybersecurity information. In [52] proxy-re-encryption is applied to ensure secure submission and storage of private information. Also concrete but quite theoretical, [29] presents a platform to share security information over multiple service infrastructures keeping damages of a DDoS attack at a minimal. Though some techniques are described, e.g. identifying well-known ports, many details are omitted. Besides, though not particularly focused on cybersecurity, privacy in relation to information sharing has been addressed through different means in this context. Such concerns include the need to prevent third parties from identifying shared information, and the need to prevent the identification of information providers. The first issue is mainly solved by the used of encryption techniques, either asymmetric [12] or symmetric [50] schemes. By contrast, guaranteeing providers' anonymity in this context has been mostly explored through aggregation [2,35]. Several techniques have been suggested based on identity-based ring signatures [30]; iterative anonymous assignation of identifiers to nodes [16]; group signatures and broadcast encryption [39]; or certificateless encryption [56].

Table 8: Related work in cybersecurity information sharing

FPE emerged as a cryptographic technique initially focused on simple data types such as numbers, since this is the case of credit cards [5]. A credit card number can be encrypted by FPE to get another credit card number. Similarly, IP addresses can be also encrypted with this scheme [11] as well as social security numbers [62]. Several proposals have applied FPE to other data formats, e.g., data and times [40] or strings of characters of a fixed length [36]. Recently, Weiss et al. [62] have looked for practical solutions for FPE including numbers and other elements like variable-length strings.

Despite the availability of recent FPE algorithms [6, 43], in most cases they have to be adapted to each particular context. Agbeybor et al. have recently explored the application of FPE in the field of critical infrastructure protection [1]. Although the domain and goals are very different from ours, the motivation for using is FPE in this context is very similar.

The use of homomorphic cryptography for privacy-preserving aggregation has been explored in multiple applications, including wireless sensor networks [53], smart metering [3], and collaboration systems [54]. In all these cases, the use of homomorphic encryption and the adversarial models follow a common pattern and are very similar among them and to our proposal.

From this analysis we can conclude that while some works present information sharing schemes, no approach support privacy-preserving and aggregatable CIS. Indeed, just [2] mentions the necessity of privacy together with aggregation but any specific mechanism is proposed. Concerning the approach taken, not a single proposal discusses the application of homomorphic encryption in this field. Moreover, though [17] points out the potential and significance of FPE for CIS, no previous proposals apply this technique either.

7 Conclusions

Sharing cybersecurity information has been identified as a key element to develop cooperative cyberdefense strategies and to prepare against cyberthreats. Privacy is paramount to foster cooperation, particularly when insecure infrastructures are used to support sharing. In this paper, we have addressed this issue by proposing PRACIS, a protocol that provides privacy-preserving and aggregatable cybersecurity information sharing. PRACIS provides these properties by leveraging existing format-preserving and homomorphic encryption techniques and adapting them to the particularities of standard message formats such as STIX. To the best of our knowledge, this is the first scheme that addresses this issue. To evaluate its feasibility in a real-world setting, we have developed a freely available proof-of concept implementation of PRACIS. Our experimental results suggest that members of an information sharing network can afford the overhead introduced by our scheme even in a regular desktop PC. At the moment, we are extending our prototype to support sharing other STIX objects, such as Indicators-of-Compromise (IoC), and Courses-of-Action (CoA). Studying and optimizing the performance of PRACIS in high performance scenarios (such as cloud computing infrastructure) is a potential research work direction, relevant to assess the real-world suitability of the proposal.

Acknowledgments

Funding: This work was partially supported by the MINECO grant TIN2013-46469-R (SPINY); the CAM grant S2013/ICE-3095 (CIBERDINE), which is co-funded by European FEDER; J. M. de Fuentes and L. González were also supported by the Programa de Ayudas para la Movilidad of Carlos III University of Madrid, Spain.

Authors would like to thank the anonymous reviewers for their comments and suggestions, which helped us to significantly improve this work.

References

1. Agbeyibor, R., Butts, J., Grimaila, M., Mills, R.: Evaluation of format-preserving encryption algorithms for critical infrastructure protection. In: Critical Infrastructure Protection VIII, pp. 245–261. Springer (2014)
2. Atabakhsh, H., Larson, C., Petersen, T., Violette, C., Chen, H.: Information sharing and collaboration policies within government agencies. In: Intelligence and Security Informatics, pp. 467–475. Springer (2004)
3. Bae, M., Kim, K., Kim, H.: Preserving privacy and efficiency in data communication and aggregation for ami network. Journal of Network and Computer Applications 59, 333–344 (2016)
4. Barnum, S.: Standardizing cyber threat intelligence information with the structured threat information expression (stix) (2014)
5. Bellare, M., Ristenpart, T., Rogaway, P., Stegers, T.: Format-preserving encryption. In: Selected Areas in Cryptography. pp. 295–312. Springer (2009)
6. Bellare, M., Rogaway, P., Spies, T.: The ffx mode of operation for format-preserving encryption. NIST submission 20 (2010)
7. Benaloh, J.D.C.: Verifiable secret-ballot elections. Yale University. Department of Computer Science (1987)
8. Black, J., Rogaway, P.: Ciphers with arbitrary finite domains. In: Preneel, B. (ed.) Topics in Cryptology — CT-RSA 2002: The Cryptographers' Track at the RSA Conference 2002 San Jose, CA, USA, February 18–22, 2002 Proceedings. pp. 114–130. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)
9. Brown, S., Gommers, J., Serrano, O.: From cyber security information sharing to threat management. In: Proceedings of the 2nd ACM Workshop on Information Sharing and Collaborative Security. pp. 43–49. ACM (2015)
10. CERT-UK: Cyber-security information sharing partnership (cisp) (2013)
11. Chandrashekar, P., Dara, S., Muralidhara, V.: Efficient format preserving encrypted databases. In: Electronics, Computing and Communication Technologies (CONECCT), 2015 IEEE International Conference on. pp. 1–4. IEEE (2015)
12. Choi, J.J.U., Ae Chun, S., Kim, D.H., Keromytis, A.: Securegov: secure data sharing for government services. In: Proceedings of the 14th Annual International Conference on Digital Government Research. pp. 127–135. ACM (2013)
13. Cyrus R. Mehta, N.R.P.: A network algorithm for performing fisher's exact test in r c contingency tables. Journal of the American Statistical Association 78(382), 427–434 (1983), http://www.jstor.org/stable/2288652
14. Dandurand, L., Serrano, O.S.: Towards improved cyber security information sharing. In: Cyber Conflict (CyCon), 2013 5th International Conference on. pp. 1–16. IEEE (2013)
15. Denning, D.E.: Framework and principles for active cyber defense. Computers and Security 40, 108 – 113 (2014)
16. Dunning, L.A., Kresman, R.: Privacy preserving data sharing with anonymous id assignment. Information Forensics and Security, IEEE Transactions on 8(2), 402–413 (2013)
17. Dupont, B.: The cyber security environment to 2022: trends, drivers and implications. Drivers and Implications (2012)
18. ENISA: Good practice guide network security information exchanges. ENISA reports (2009)

19. ENISA: Standards and tools for exchange and processing of actionable information. ENISA reports (2014)
20. Erkin, Z., Tsudik, G.: Private computation of spatial and temporal power consumption with smart meters. In: Applied Cryptography and Network Security. pp. 561–577. Springer (2012)
21. Gal-Or, E., Ghose, A.: The economic consequences of sharing security information. In: Economics of information security, pp. 95–104. Springer (2004)
22. Gal-Or, E., Ghose, A.: The economic incentives for sharing security information. Information Systems Research 16(2), 186–208 (2005)
23. Garca-Teodoro, P., Daz-Verdejo, J., Maci-Fernndez, G., Vzquez, E.: Anomaly-based network intrusion detection: Techniques, systems and challenges. Computers and Security 28(12), 18 – 28 (2009), http://www.sciencedirect.com/science/article/pii/S0167404808000692
24. Garrido-Pelaz, R., González-Manzano, L., Pastrana, S.: Shall we collaborate?: A model to analyse the benefits of information sharing. In: Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security. pp. 15–24. ACM (2016)
25. Gordon, L.A., Loeb, M.P., Lucyshyn, W., Zhou, L.: The impact of information sharing on cybersecurity underinvestment: a real options perspective. Journal of Accounting and Public Policy 34(5), 509–519 (2015)
26. Groat, M.M., He, W., Forrest, S.: Kipda: k-indistinguishable privacy-preserving data aggregation in wireless sensor networks. In: Proceedings of INFOCOM. pp. 2024–2032. IEEE (2011)
27. He, D., Wang, H., Wang, L., Shen, J., Yang, X.: Efficient certificateless anonymous multi-receiver encryption scheme for mobile devices. Soft Computing pp. 1–10 (2016)
28. Hernandez-Ardieta, J.L., Tapiador, J.E., Suarez-Tangil, G.: Information sharing models for cooperative cyber defence. In: Cyber Conflict (CyCon), 2013 5th International Conference on. pp. 1–28. IEEE (2013)
29. Hu, B., Murata, Y., Murayama, J.: Security information sharing platform over multiple services. In: Information and Telecommunication Technologies (APSITT), 2015 10th Asia-Pacific Symposium on. pp. 1–3. IEEE (2015)
30. Huang, X., Liu, J.K., Tang, S., Xiang, Y., Liang, K., Xu, L., Zhou, J.: Cost-effective authentic and anonymous data sharing with forward security. Computers, IEEE Transactions on 64(4), 971–983 (2015)
31. ISO/IEC JTC 1/SC 27: Information technology – security techniques – information security management for inter-sector and inter-organizational communications (2015), iSO/IEC 27010:2015
32. Khouzani, M., Pham, V., Cid, C.: Strategic discovery and sharing of vulnerabilities in competitive environments. In: International Conference on Decision and Game Theory for Security. pp. 59–78. Springer (2014)
33. Kokkonen, T., Hautamỳki, J., Siltanen, J., Hỳmỳlỳinen, T.: Model for sharing the information of cyber security situation awareness between organizations. In: Telecommunications (ICT), 2016 23rd International Conference on. pp. 1–5. IEEE (2016)
34. Kumar, M., Verma, S., Lata, K.: Secure data aggregation in wireless sensor networks using homomorphic encryption. International Journal of Electronics 102(4), 690–702 (2015)
35. Li, H., Xiong, L., Zhang, L., Jiang, X.: Dpsynthesizer: differentially private data synthesizer for privacy preserving data sharing. Proceedings of the VLDB Endowment 7(13), 1677–1680 (2014)
36. Li, M., Liu, Z., Li, J., Jia, C.: Format-preserving encryption for character data. Journal of Networks 7(8), 1239–1244 (2012)
37. Li, X., Chen, D., Li, C., Wang, L.: Secure data aggregation with fully homomorphic encryption in large-scale wireless sensor networks. Sensors 15(7), 15952–15973 (2015)

38. Liu, Q., Wang, G., Wu, J.: Time-based proxy re-encryption scheme for secure data sharing in a cloud environment. Information Sciences 258, 355–370 (2014)
39. Liu, X., Zhang, Y., Wang, B., Yan, J.: Mona: secure multi-owner data sharing for dynamic groups in the cloud. Parallel and Distributed Systems, IEEE Transactions on 24(6), 1182–1191 (2013)
40. Liu, Z., Jia, C., Li, J., Cheng, X.: Format-preserving encryption for datetime. In: Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on. vol. 2, pp. 201–205. IEEE (2010)
41. Luchaup, D., Dyer, K.P., Jha, S., Ristenpart, T., Shrimpton, T.: Libfte: A toolkit for constructing practical, format-abiding encryption schemes. In: 23rd USENIX Security Symposium (USENIX Security 14). pp. 877–891. USENIX Association, San Diego, CA (Aug 2014)
42. Makedon, F., Sudborough, C., Baiter, B., Conalis-Kontos, M.: A safe information sharing framework for e-government communication (2015)
43. Mattsson, U., Blomkvist, K.: Data type preserving encryption (Aug 26 2008), uS Patent 7,418,098
44. Mitchell Gail, N.M.: Counting the number of r c contingency tables with fixed margins. Journal of the American Statistical Association 72(360), 859–862 (1977), http://www.jstor.org/stable/2286475
45. MITRE: Making security measurable. https://makingsecuritymeasurable.mitre.org/, accessed: 2016-04-28
46. MITRE: Trusted automated exchange of indicator information (taxii). http://taxiiproject.github.io, accessed: 2016-04-22
47. Murdoch, S., Leaver, N.: Anonymity vs. trust in cyber-security collaboration. In: Proceedings of the 2nd ACM Workshop on Information Sharing and Collaborative Security. pp. 27–29. ACM (2015)
48. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) Advances in Cryptology — EUROCRYPT '99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings. pp. 223–238. Springer Berlin Heidelberg (1999)
49. Phillips Jr, C.E., Ting, T., Demurjian, S.A.: Information sharing and security in dynamic coalitions. In: Proceedings of the seventh ACM symposium on Access control models and technologies. pp. 87–96. ACM (2002)
50. Prasad, K., Poonam, J., Gauri, K., Thoutam, N.: Data sharing security and privacy preservation in cloud computing. In: Green Computing and Internet of Things (ICGCIoT), 2015 International Conference on. pp. 1070–1075. IEEE (2015)
51. PriceWaterHouseCoopers: Global state of information security survey 2015. http://www.pwc.com/gx/en/consulting-services/information-security-survey/assets/the-global-state-of-information-security-survey-2015.pdf, last access: 2016-11-09
52. Raj, A., Arunprasath, R., Vigneshwari, S.: Efficient mechanism for sharing private data in a secured manner. In: Circuit, Power and Computing Technologies (ICCPCT), 2016 International Conference on. pp. 1–4. IEEE (2016)
53. Ramotsoela, T., Hancke, G.: Data aggregation using homomorphic encryption in wireless sensor networks. In: Information Security for South Africa (ISSA), 2015. pp. 1–8. IEEE (2015)
54. Rieffel, E.G., Biehl, J.T., Lee, A.J., Van Melle, W.: Private aggregation for presence streams. Future Generation Computer Systems 31, 169–181 (2014)
55. Savi, M., Rottondi, C., Verticale, G.: Evaluation of the precision-privacy tradeoff of data perturbation for smart metering. Smart Grid, IEEE Transactions on 6(5), 2409–2416 (2015)

56. Seo, S.H., Nabeel, M., Ding, X., Bertino, E.: An efficient certificateless encryption for secure data sharing in public clouds. Knowledge and Data Engineering, IEEE Transactions on 26(9), 2107–2119 (2014)
57. Skopik, F., Settanni, G., Fiedler, R.: A problem shared is a problem halved: A survey on the dimensions of collective cyber defense through security information sharing. Computers & Security 60, 154–176 (2016)
58. Tandon school of enfineering: Mapping the next frontier of open data: Corporate data sharing Last access Nov. 2015
59. Tosh, D., Sengupta, S., Kamhoua, C., Kwiat, K., Martin, A.: An evolutionary game-theoretic framework for cyber-threat information sharing. In: 2015 IEEE International Conference on Communications (ICC). pp. 7341–7346. IEEE (2015)
60. USGovernment: Cybersecurity information sharing act (in progress) (2015)
61. Webster, G.D., Hanif, Z.D., Ludwig, A.L., Lengyel, T.K., Zarras, A., Eckert, C.: Skald: A scalable architecture for feature extraction, multi-user analysis, and real-time information sharing. In: International Conference on Information Security. pp. 231–249. Springer (2016)
62. Weiss, M., Rozenberg, B., Barham, M.: Practical solutions for format-preserving encryption. arXiv preprint arXiv:1506.04113 (2015)
63. Zhao, W., White, G.: A collaborative information sharing framework for community cyber security. In: Homeland Security (HST), 2012 IEEE Conference on Technologies for. pp. 457–462. IEEE (2012)
64. Zhou, C.V., Leckie, C., Karunasekera, S.: A survey of coordinated attacks and collaborative intrusion detection. Computers and Security 29(1), 124 – 140 (2010), http://www.sciencedirect.com/science/article/pii/S016740480900073X

José María de Fuentes is visiting lecturer in the Computer Science and Engineering Department at University Carlos III of Madrid, Spain. He is Computer Scientist Engineer and Ph.D. in Computer Science by the University Carlos III of Madrid. His main research interests are cybersecurity as well as security and privacy in the internet of things and ad-hoc networks. He has published several articles in international conferences and journals. He is participating in several national R+D projects.

Lorena González-Manzano is assistant professor working in the Computer Security Lab at the University Carlos III of Madrid, Spain. She is Computer Scientist Engineer and Ph.D. in Computer Science by the University Carlos III of Madrid. Her Ph.D. focuses on security and privacy in social networks. She is currently focused on Internet of Things and cloud computing security, as well as, on cybersecurity. Indeed, she has published several papers in national and international conferences and journals and she is also involved in national R+D projects.

Juan Tapiador is Associate Professor of Computer Science in the Computer Security Lab (COSEC) at Universidad Carlos III de Madrid, Spain. Prior to joining UC3M, he was Research Associate at the University of York, UK. His research interests are in computer security, including malware analysis, anomaly and intrusion detection, attack modeling and cyberdefense systems. He holds a M.Sc. (2000) and a Ph.D. (2004) in Computer Science from the University of Granada.

Pedro Peris-Lopez is Visiting Lecturer at the Department of Computer Science, Universidad Carlos III de Madrid, Spain. He holds a M.Sc. in Telecommunications Engineering and Ph.D. in Computer Science. His research interests are in the field of protocols design, primitives design, lightweight cryptography, cryptanalysis etc. Nowadays, his research is focused on Radio Frequency Identification Systems (RFID) and Implantable Medical Devices (IMD). In these fields, he has published a great number of papers in specialized journals and conference proceedings. For additional information see: www.lightweightcryptography.com/.

A PRACIS Algorithms

input : -
output: Keys are stored by the affected entities. Parameters initialized

foreach D_i *in [D]* **do**
  $D_i : K_{FPE}(D_i)$ = createFPEKey();
  $D_i : [K_{D_i}(t)]$ = createSymKeyPerType();
  D_i: [selectedIS] = findAlliedIS();
  foreach IS_j *in [selectedIS]* **do**
    $D_i \rightarrow IS_j$: send($K_{FPE}(D_i)$, $[K_{D_i}(t)]$);
    D_i : storeKeys(IS_j,$K_{FPE}(D_i)$,$[K_{D_i}(t)]$);
    IS_j : storeKeys(D_i, $K_{FPE}(D_i)$,$[K_{D_i}(t)]$);
  end
end
StS: K_H = createHomomorphicKey();
StS: $N_{b_{rnd}}$ = maxRandom();
foreach D_i *in [D]* **do**
  $StS : r_{D_i}$ = createRandom();
  $StS \rightarrow D_i$: send(K_H,r_{D_i}, $N_{b_{rnd}}$);
  D_i : storeKey(K_H);
  D_i : storeRandom(r_{D_i});
  D_i : storeMaxSizeRandom($N_{b_{rnd}}$);
  StS: storeKey(K_H);
  StS: storeRandom(D_i, r_{D_i});
end
AF: K_{HMAC} = createHMACKey();
AF: storeKey(K_{HMAC});
foreach D_i *in [D]* **do**
  $AF \rightarrow D_i$: send(K_{HMAC});
end
$AF \rightarrow StS$: send(K_{HMAC});
foreach IS_j *in [IS]* **do**
  $AF \rightarrow IS_j$: send(K_{HMAC});
end

Algorithm 1: Setup procedure.

input : stix, D_i, currentAggregation, counterPkgsOrigin, K_{HMAC}, K_H
output: STIX packet sent to all subscribers, packet aggregated by AF, or $false$

```
// # integrity check
receivedChecksum = stix.VictimID stix.VictimID = null AF: checksum =
  HMAC(stix,K_HMAC) if checksum != receivedChecksum then
    abort;
else
    // # incident forwarding
    AF : subscribers = findSubscribers(stix.type);
    foreach sbs in subscribers do
        AF → sbs : send(stix);
    end
    // # incident aggregation
    AF: currentAggregation = HAdd(stix.IncidentID, currentAggregation);
    AF: storeAggregation(currentAggregation);
    AF: counterPkgsOrigin = addCounter(D_i);
    // # this is a matrix, one value per D_i
end
```

Algorithm 2: Incident aggregation and forwarding by AF.

input : [symkey],type t_i
output: IS is subscribed to events of type t_i

```
foreach IS_i in [IS] do
    IS_i : K_s(D_i) = findKey(D_i);
    IS_i : subsPkg = E(K_s(D_i), t_i);
    IS_i → AF : send(subsPkg);
    AF : storeSubscription(IS_i,subsPkg);
end
```

Algorithm 3: Subscription to a event type.

input : t_i, $K_{FPE}(D_i)$, confidence, effect, asset, r_{D_i}, K_H, K_{HMAC}
output: STIX packet delivered to af

```
D_i : stix = createSTIXPackage();
D_i : symKey = findKeyPerType(t_i);
D_i : stix.type = E(symKey,t_i);
D_i : stix.confidence = FPE(K_FPE(D_i), confidence);
D_i : stix.effect= FPE(K_FPE(D_i), effect);
D_i : stix.asset = FPE(K_FPE(D_i), asset);
D_i : stix.id = HE(K_H, type,r_Di);
D_i : stix.victimID = HMAC(K_HMAC, stix);
D_i : r_Di = (r_Di + 1) mod (2^N_brnd − 1) ;
D_i → AF : send(stix);
```

Algorithm 4: STIX package preparation and delivery.

input : [currentAggregation,counterPkgsOrigin] sent by AF after I_{max} incidents (recall Algorithm 2), $N_{b_{rnd}}$
output: Incident statistics calculated by StS and updated random values for each r_{D_i}, or $false$

StS : K_H = loadHomomorphicKey();
StS : resultTypes, resultCount = HD(K_H, [currentAggregation,counterPkgsOrigin]);
expectedSum=0;
foreach D_i *in [D]* **do**
    StS : finalRandomDi = (loadRandom(D_i) + resultCount[D_i]) mod $(2^{N_{b_{rnd}}} - 1)$;
    StS : expectedSum = expectedSum+finalRandomDi;
end
if *expectedSum != resultCount* **then**
    abort;
else
    StS: updateCounters(resultCount);
end

Algorithm 5: STIX processing by StS.

input : stix, D_i, K_{HMAC}, $K_{FPE}(D_i)$
output: STIX packet processed by subscriber or $false$

`// # integrity check`
receivedChecksum = stix.VictimID stix.VictimID = null AF: checksum = HMAC(stix,K_{HMAC}) **if** *checksum != receivedChecksum* **then**
    abort;
else
    IS_i : $K_{FPE}(D_i)$ = findFPEKey(stix.victimName);
    IS_i : RealConfidence = FPD($K_{FPE}(D_i)$, stix.confidence);
    IS_i : RealEffect= FPD($K_{FPE}(D_i)$, stix.effect);
    IS_i : RealAsset = FPD($K_{FPE}(D_i)$, stix.asset);
end

Algorithm 6: Incident verification and decryption by IS_i.

Figures and tables

```
<stix:STIX_Package >
    <stix:Incidents>
        <stix:Incident id="example:incident-8236b4a2-abe0-4b56-9347-28800c4bb92" timestamp="2014-11-18T23:40:08.061362+00:00" xsi:type='in
cident:IncidentType' version="1.2">
            <incident:Title>Breach of Cyber Tech Dynamics</incident:Title>
            <incident:Time>
                <incident:Initial_Compromise precision="second">2012-01-30T00:00:00</incident:Initial_Compromise>
                <incident:Incident_Discovery precision="second">2012-05-10T00:00:00</incident:Incident_Discovery>
                <incident:Restoration_Achieved precision="second">2012-08-10T00:00:00</incident:Restoration_Achieved>
                <incident:Incident_Reported precision="second">2012-12-10T00:00:00</incident:Incident_Reported>
            </incident:Time>
            <incident:Description>Intrusion into enterprise network</incident:Description>
            <incident:Reporter>
                <stixCommon:Description>The person who reported it</stixCommon:Description>
                <stixCommon:Identity id="example:Identity-cd64aaa6-b1c0-4026-8ea1-14ff5a19e5fb">
                    <stixCommon:Name>Sample Investigations, LLC</stixCommon:Name>
                </stixCommon:Identity>
                <stixCommon:Time>
                    <cyboxCommon:Produced_Time>2014-03-11T00:00:00</cyboxCommon:Produced_Time>
                </stixCommon:Time>
            </incident:Reporter>
            <incident:Victim id="example:Identity-dd8637b7-51b4-48f0-9e3c-a2b23b3a2dd7">
                <stixCommon:Name>Cyber Tech Dynamics</stixCommon:Name>
            </incident:Victim>
            <incident:Impact_Assessment>
                <incident:Effects>
                    <incident:Effect xsi:type="stixVocabs:IncidentEffectVocab-1.0">Financial Loss</incident:Effect>
                </incident:Effects>
            </incident:Impact_Assessment>
            <incident:Confidence timestamp="2014-11-18T23:40:08.061379+00:00">
                <stixCommon:Value xsi:type="stixVocabs:HighMediumLowVocab-1.0">High</stixCommon:Value>
            </incident:Confidence>
        </stix:Incident>
    </stix:Incidents>
</stix:STIX_Package>
```

Fig 1. An example of a STIX message [4].

Table 1. Notation

Element	Description
D_i	Publisher i
StS	Statistics subscriber
AF	Message broker
IS_i	Incident subscriber i
I_{t_i}	Incident of type t_i
$K_{FPE}(D_i)$	Format-Preserving Encryption key of D_i
$K_{s(D_i)(t_i)}$	Secret/ Symmetric key of D_i per type of t_i
K_H	Homomorphic encryption key
K_{HMAC}	HMAC key
$HE(k, x)$	Homomorphic encryption of x using key k
$HAdd(x_1, x_2)$	Homomorphic addition of messages x_1 and x_2
r_{D_i}	Random value unique per D_i

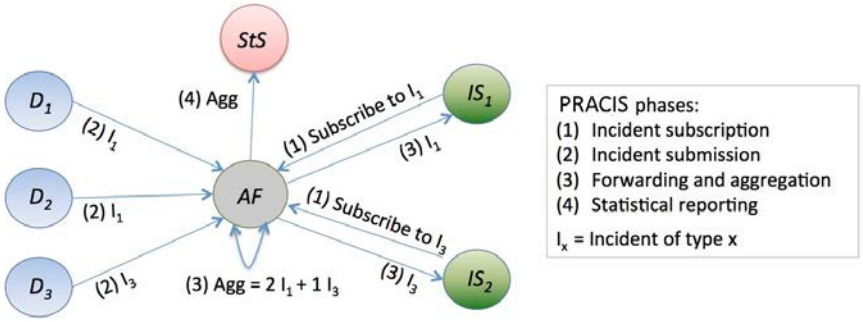

Fig 2. Overview of PRACIS

```
<stix:STIX_Package >
  <stix:Incidents>
    <stix:Incident id="example:Incident-8236b4a2-abe0-4b56-9347-288005c4bb92" timestamp="2014-11-18T23:40:08.061362+00:00"
xsi:type='Incident:IncidentType' version="1.2">
      <incident:Title>Breach of Cyber Tech Dynamics</incident:Title>
      <incident:Time>
        <incident:Initial_Compromise precision="second">2012-01-30T00:00:00</incident:Initial_Compromise>
        <incident:Incident_Discovery precision="second">2012-05-10T00:00:00</incident:Incident_Discovery>
        <incident:Restoration_Achieved precision="second">2012-08-10T00:00:00</incident:Restoration_Achieved>
        <incident:Incident_Reported precision="second">2012-12-10T00:00:00</incident:Incident_Reported>
      </incident:Time>
    <incident:Affected_Assets>
        <incident:Affected_Asset>
          <incident:Type count_affected="1">Database</incident:Type>
        </incident:Affected_Asset>
      </incident:Affected_Assets>
      <incident:Description id="example:Identity-d3a637b7-5c64-21f0-9e3c-a2b23b3a4aa7">Intrusion into enterprise network</incident:Description>
          <incident:Victim id="example:Identity-dd8637b7-51b4-48f0-9e3c-a2b23b3a2dd7">
            <stixCommon:Name>Cyber Tech Dynamics</stixCommon:Name>
          </incident:Victim>
          <incident:Impact_Assessment>
            <incident:Effects>
              <incident:Effect xsi:type="stixVocabs:IncidentEffectVocab-1.0">Financial Loss</incident:Effect>
            </incident:Effects>
          </incident:Impact_Assessment>
          <incident:Confidence timestamp="2014-11-18T23:40:08.061379+00:00">
            <stixCommon:Value xsi:type="stixVocabs:HighMediumLowVocab-1.0">High</stixCommon:Value>
          </incident:Confidence>
    </stix:Incident>
  </stix:Incidents>
</stix:STIX_Package>
```

Fig 3. Example of STIX incident format to apply PRACIS.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	r_{D_i}
$I_4(D_1)$	0000000	0000000	0000000	0000001	0000000	0000000	0000000	0000000110001
$I_4(D_2)$	0000000	0000000	0000000	0000001	0000000	0000000	0000000	0000000010100
$I_5(D_3)$	0000000	0000000	0000000	0000000	0000001	0000000	0000000	0000000010101
Aggregated	**0000000**	**0000000**	**0000000**	**0000010**	**0000001**	**0000000**	**0000000**	**0000001011010**

Table 2. Example of *Incident:id* aggregation.

Table 3: Example of contingency table of event types and deliverers. Cells in bold are the values known to StS after PRACIS, whereas values in italics are the ones to be discovered in the reidentification attack.

	$\mathbf{t_1}$	$\mathbf{t_2}$	$\mathbf{t_3}$	$\mathbf{t_4}$	$\mathbf{t_5}$	$\mathbf{t_6}$	$\mathbf{t_7}$	$\mathbf{t_8}$	$\mathbf{t_9}$	**Marginal (per $\mathbf{D_i}$)**
$\mathbf{D_1}$	*1*	*1*	*1*	*0*	*0*	*0*	*1*	*3*	*3*	**10**
$\mathbf{D_2}$	*4*	*4*	*4*	*4*	*4*	*4*	*4*	*1*	*1*	**30**
Marginal (per $\mathbf{t_i}$)	**5**	**5**	**5**	**4**	**4**	**4**	**5**	**4**	**4**	

Entity	**Tasks**	**Time (ms)**
D_i	Time setup	83.84
	Homomorphic Encryption	21.15
	Format-Preserving encryption	64.94
	HMAC computation	0.04
	STIX message fill-up	1.24
	Build STIX message	**87.37**
IS_i	Format-preserving decryption	104.94
	HMAC computation & comparison	0.04
	STIX message decryption & verification	**104.98**
AF	2 messages id aggregation	0.175
	Aggregation of $I_{max} = 100$	16.06
	Aggregation of $I_{max} = 1000$	180.153
	Aggregation of $I_{max} = 10000$	2120.660

Table 4: Creation, decryption, verification and aggregation of STIX messages (average)

Table 5: Incident creation and aggregation scalability analysis

STIX incident creation time (ms)		**STIX incident aggregation time (ms) Amount of D_i**				
Incidents / minute	**Time**	**10**	**100**	**200**	**400**	**800**
40	3494.8	54.15	540.1	1089.42	2213.19	4377.99
80	6989.6	110.63	1089.42	2213.19	4377.99	8682.56
160	13979.2	224.57	2213.19	4377.99	8682.56	17783.71
320	27958.4	450.07	4377.99	8682.56	17783.71	36100.93
640	55916.8	902.48	8682.56	17783.71	36100.93	72923.88
1280	111833.6	1831.71	17783.71	36100.93	72923.88	147306.24

$\|Di\|$	Interval	I_{max}	$N_{b_{total}}$ (bits)	$N_{b_{AV}}$ (bits)	len(Incident:id) (bits)	len(Incident:id) (bytes)
10	1	810	330	30	360	45
	5	4050	396	32	428	54
	10	8100	429	33	462	58
100	1	8100	429	33	462	58
	5	40500	528	36	564	71
	10	81000	561	37	598	75
1000	1	81000	561	37	598	75
	5	405000	627	39	666	84
	10	810000	660	40	700	88

Table 6: Overhead in *Incident:id* size after encryption depending on D_i and aggregation intervals

	Min.	**Avg.**	**Max.**	**Max. overhead**	**Avg. overhead**
Confidence	3	5	7	4	2
Effect	11	33	71	60	22
Asset	3	8	15	12	5
Total	**17**	**46**	**93**	**76**	**29**

Table 7: Overhead of FPE fields (bytes)

Table 8: Related work in cybersecurity information sharing

	Need for CIS	**Theoretical approach to CIS**	**Practical mechanism to CIS**	**Mechanism technique(s)**	**Privacy-preserving CIS**	**Aggregation of incidents**
[49]	✓					
[21]	✓					
[63]	✓					
[25]	✓					
[59]	✓					
[32]	✓					
[24]	✓					
[33]		✓				
[14]		✓				
[61]		✓				
[52]			✓	Proxy re-encryption	✓	
[29]			✓	Port scanning		
[35]			✓	Aggregation		✓
[2]			✓	Aggregation	✓*	✓*
[30]			✓	Identity-based ring signature		
[16]			✓	Iterative anonymous ID assignation	✓	
[39]			✓	Group signatures and broadcast encryption	✓	
[27]			✓	Certificateless encryption	✓	
PRACIS			✓	Homomorphic encryption and format-preserving encryption	✓	✓

* mentioned but not detailed