Intelligent conditional collaborative private data sharing

Giuseppe Bianchi, Tooska Dargahi, Alberto Caponi, Mauro Conti

PII: S0167-739X(18)30774-X
DOI: https://doi.org/10.1016/j.future.2019.01.001
Reference: FUTURE 4694

To appear in: *Future Generation Computer Systems*

Received date : 31 March 2018
Revised date : 22 November 2018
Accepted date : 1 January 2019

Intelligent Conditional Collaborative Private Data Sharing

Giuseppe Bianchi, Tooska Dargahi, Alberto Caponi, and Mauro Conti

Abstract With the advent of distributed systems, secure and privacy-preserving data sharing between different entities (individuals or organizations) becomes a challenging issue. There are several real-world scenarios in which different entities are willing to share their private data *only* under certain circumstances, such as sharing the system logs when there is indications of cyber attack in order to provide cyber threat intelligence. Therefore, over the past few years, several researchers proposed solutions for *collaborative data sharing*, mostly based on existing cryptographic algorithms. However, the existing approaches are not appropriate for *conditional* data sharing, i.e., sharing the data *if and only if* a pre-defined condition is satisfied due to the occurrence of an event. Moreover, in case the existing solutions are used in *conditional* data sharing scenarios, the shared secret will be revealed to all parties and re-keying process is necessary. In this work, in order to address the aforementioned challenges, we propose, a *"conditional collaborative private data sharing"* protocol based on *Identity-Based Encryption* and *Threshold Secret Sharing* schemes. In our proposed approach, the condition based on which the encrypted data will be revealed to the collaborating parties (or a central entity) could be of two types: (i) threshold, or (ii) pre-defined policy. Supported by thorough analytical and experimental analysis, we show the effectiveness and performance of our proposal.

Giuseppe Bianchi
CNIT - University of Rome Tor Vergata, Italy, e-mail: giuseppe.bianchi@uniroma2.it

Tooska Dargahi
University of Salford, UK, and CNIT - University of Rome Tor Vergata, Italy, e-mail: t.dargahi@salford.ac.uk

Alberto Caponi
CNIT - University of Rome Tor Vergata, Italy, e-mail: alberto.caponi@uniroma2.it

Mauro Conti
University of Padua, Italy e-mail: conti@math.unipd.it

1 Introduction

New generation networking paradigms, such as Cloud, have made data sharing between individuals or organizations easier and simpler than ever before. However, preserving confidentiality and privacy of the shared data (which could be privacy sensitive) is an important and challenging issue in such networks. This issue becomes more significant with regard to distributed systems, in which different systems might have their own access control policies for the shared data. Therefore, providing an intelligent private data sharing method that allows the involved parties to decide when, to whom, and to what extent they should share their private data is important [9,13].

Over the past few years, collaborative data sharing has attracted attention of governments, academia, and industry, due to a multitude of real-world applications of such a data sharing need. For example, consider the promotion of cyber threat information sharing announced by the US government in 2015 [16]: *"In order to address cyber threats to public health and safety, national security, and economic security of the United States, private companies, nonprofit organizations, executive departments and agencies, and other entities must be able to share information related to cybersecurity risks and incidents and collaborate to respond in as close to real time as possible"*. As another example, consider large-scale disaster recovery scenarios, such as the *WannaCry* worldwide ransomware attack in May 2017. In such a scenario, several crisis information systems belonged to different organizations need to share their private data in order to provide Cyber Threat Intelligence (CTI) to take timely actions [8]. It should be noted that, though in these (and other similar) scenarios, collaborative data sharing is necessary, at the same time, preserving privacy of individuals and confidentiality of the business data is also important [16]. Therefore, an intelligent and secure privacy-preserving conditional data sharing method should be in place in order to ensure the confidentiality and accuracy of the shared data.

Motivation and Related Work

Recently, researchers have paid more attention to the secure data sharing issue, and proposed various cryptographical or non-cryptographical solutions for private data sharing. This ranges from (just to mention a few) cybersecurity [13], smart metering [10], cloud computing [25], cross project defect prediction [29] and statistical data analysis [24], to online social networks [17]. Most of them basically preserve privacy of shared data by applying different methods such as data aggregation [24], anonymization [11], obfuscation [29], multi-party computation [10,13], or proxy re-encryption [18,22,27]. We believe that the existing data sharing methods have two limitations: (1) *Scalability:* they are not scalable in terms of number of datasets, i.e., if the data owner wants to selectively restrict access of other entities to *different sets* of encrypted data, he should perform several key agreement procedures with *all* the other entities for *each* dataset. (2) *Conditionality:* they do not support *conditional data disclosure*, i.e., the scenarios in which the collaborating entities are willing to disclose *only a specific* set of encrypted data *if and only if* a certain condition holds (e.g., entities identify indications of a global cyber attack).

Moreover, recently some researchers proposed game–theoretic approaches [19, 20]. They consider a game between the attackers and the defenders, based on which they decide the collaboration strategy between different parties. The difference between our solution and game-theoric methods is that we consider a scenario in which the collaborating entities "must" share a specific piece of data due to a previous agreement. However, all the collaborating entities might not be available at the same time. In such a scenario, our solution will help other entities to access that piece of data, while we preserve confidentiality of other parts of the dataset. Therefore, the data owner has granular control on the amount of data that is shared in emergency situation.

Running example: In order to elaborate more on the problem definition and the importance of *scalability* and *conditionality* issues, let us make a small example of a cyber attack scenario. Consider a hierarchical banking system (see Figure 1): in the first (highest) level, there is the country's *central governmental bank*, whose main role is to provide financial, statistical, and advisory services to all banks in the country. In the second level (*Bank A* to *Bank M* in Figure 1), the central offices of all different independent banks that exist in a country. In the third (lowest) level (*Bank A.1* to *Bank A.n* in Figure 1), each bank has a large number of branches in all cities (though we could consider another forth level for classifying the branches based on the cities, for simplicity we ignore this level). It is normal to imagine that all the branches of one bank share their private data (possibly encrypted) with the central office of the corresponding bank (i.e., entities of the third level share the encrypted data with their parent entity in the second level). This data could be, monetary or non-monetary (e.g., system logs of the users accessing the PCs in each branch). Moreover, due to some reasons (e.g., country-wide cyber attack to banking system) central office of each bank (e.g., *Bank A*) might need to share its own private data with other banks (*Bank B* to *Bank M* in the second level in Figure 1), and/or with the central governmental bank (in the first level). In such a scenario, if *Bank A* releases its encrypted system log and the secret key for its decryption to the other entities at the same time, as soon as reception of the encrypted private data, other parties will be able to decrypt the data (which is not desirable due to confidentiality and privacy concerns). In fact, all banks wish to share their sensitive data *if and only if* a specific event happens, e.g., they recognize that they are under global attack. In such a situation, *Bank A* will have two options: (i) to share the encrypted data, but keep the secret key unless it recognizes the occurrence of an event; or (ii) to share both the encrypted data and the secret after occurrence of the event. However, both cases impose delay and are not efficient in emergency situations. Moreover, considering the fact that each bank might have several different datasets (e.g., security logs, financial reports, software update logs, etc.), for each dataset it requires to consider a secret key and perform a key sharing procedure with all the other entities. Otherwise, in case of considering just one secret key for all the datasets, other entities will have access to other datasets that *Bank A* is not actually willing to share.

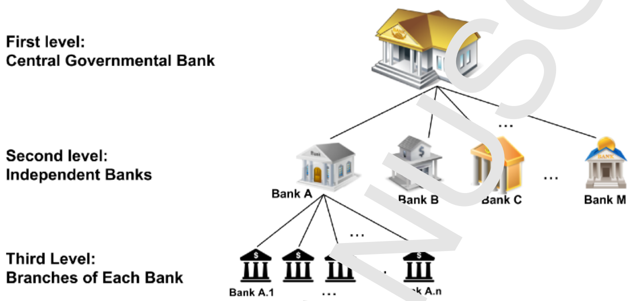

Fig. 1: Simple overview of the system model

In such scenarios, a scalable, and easy to deploy secret sharing method is required, such that it is independent from the number of involved entities and number of released datasets. A possible solution for the scalability challenge could be utilization of the Attribute-Based Encryption (ABE) [15], or Identity-Based Encryption (IBE) [6] that allow the data owner to specify the data decryptors based on their specific attributes or identities, respectively. However, the conditionality challenge is still unsolved, and to the best of our knowledge there is no solution in the literature (except our preliminary work [3, 4] that are solutions for specific networking use cases and we extend them in the current work).

Contribution

In order to address the aforementioned two challenges, in this work, we propose a *conditional collaborative private data sharing* protocol that provides a scalable and easy to deploy cryptographical method for data sharing scenarios. In particular, in order to deal with the *"scalability"* challenge, our proposal enables the data owner to encrypt each dataset with a unique secret key. Therefore, the disclosure of one dataset does not disclose information about the other datasets. We remove the requirement of key management between different parties by translating the pre-defined conditions for the authorized entities to an *access policy*, in particular *access matrix*. Based on this pre-defined access matrix, the participating entities in the data sharing process (which we call them *collaborating entities*) construct their own *share of a master secret* $\mathfrak{s}$. We use a distributed key generation scheme proposed by Pedersen [28], to construct the unique shared master secret $\mathfrak{s}$ (which is not disclosed to any entity) out of the entities' shares. The only required coordination between the collaborating entities should take place just once during the global setup phase, in which the entities decide on the dataset name that they are going to generate (we do not consider any limitation on the number of datasets that could be produced).

In order to cope with the *"conditionality"* challenge, we provide cryptographically conditioning by permitting the collaborating entities to decrypt a specific dataset *only* if they satisfy the pre-defined policy and recover the relevant secret key. We apply a fully distributed process by adopting a combination of *Identity-Based Encryption* (IBE) [6], and *Linear Secret Sharing* (LSS) [2] (in particular, *Threshold Secret Sharing* [31]). Our approach permits the reconstruction of the secret key for each dataset (i.e., for each identity) *only* when the provided shares of the relevant cryptographic material satisfy the pre-defined access policy. The pre-defined policy could be either a threshold number of collaborating entities (e.g., *if* at least three banks in level two of the hierarchy in Figure 1 report that they are under attack), or reception of the shares from specific entities (e.g., *if* $\langle$*Bank A* AND *Bank C*$\rangle$ report they are under attack; or *if* the central governmental bank signals an attack).

In our preliminary work [3, 4], we proposed conditional data sharing solutions for specific use case scenarios. In this paper, we extend our previous work and provide a comprehensive solution for conditional data sharing which could be: (i) threshold-based, or (ii) based on a specific access policy due to the network designer's choice. Moreover, we provide a security discussion in which we consider several attack models and discuss the resilience of our proposal against each of them. In addition, we enhance the security of our original proposal to cope with cheating entities on the shared secret, which we did not address in our previous work. We modified our initial proposal by permitting each party participating in the setup protocol to explicitly verify each (fraction of) share received during the protocol setup. We also provide a new use case example in order to show the applicability of our proposed approach to diverse real-world scenarios. Our new use case focuses on intelligent sharing and analysis of cyber threat information and indicators of compromise (IoCs), in particular Ransomware threat intelligence (Section 4.1). In this new use case, each individual device in an organization (e.g., personal computers in university offices, computers in each branch of a bank, or a hospital) plays the role of a collaborating entity. Each device stores logs (e.g., activity on the files, user login, network traffic, contacted IP addresses) based on a specific index, and stores an encrypted version of the log in a shared database (though it can be saved locally on the user's system in a shared folder). Upon receiving an indication of a cyber attack, the admin of the system (e.g., in Figure 1 nodes in the first or second level could be considered as the admin or the authority) should have access to all the encrypted logs in order to perform analysis of the attack entry point, and in some cases it might be possible to prevent malware spreading on network shares by investigating the vulnerable points extracted from the security logs.

2 Models and Priliminaries

In this section we describe our considered system model (Section 2.1), and attack model (Section 2.2), and we explain the assumptions that will be used in the remainder of the paper. Moreover, we provide relevant background information (Section 2.3) about the cryptographical tools that we utilize in the proposed approach. Table 1 reports the notations that we use in this paper.

Table 1: Notation table.

Notation	Description
E_i	Collaborating entities, $i = \{1, ..., n\}$
ID_j	Unique identifier of the target $\mathcal{T}_j$
$D_{i,j} \rightarrow \{E_i, ID_j\}$	Dataset stored by E_i for the target ID_j
$K_{i,j} = PRF(S_i, ID_j)$	Secret key used by E_i to encrypt the dataset related to target ID_j, using random secret S_i
$\mathbb{E}_{ID_j}$	Identity based encryption using the ID_j as identity
$\mathfrak{s}$	Unique shared master secret key
$g^{\mathfrak{s}}$	Global public key associated to the $\mathfrak{s}$
$H(ID_j)^{\mathfrak{s}}$	IBE private key of the target ID_j
$H(ID_j)^{x_i}$	E_i's share of the IBE private key associated to the target ID_j
H	Cryptographic hash function
$\mathcal{A}$	Access Structure
$\mathbb{F}$	Finite field

2.1 System Model

Figure 2 shows a simple overview of the considered system model. We consider a set of n independent entities $E = \{E_1, ..., E_n\}$, each of which being recognizable by a unique name or identity. We assume that each entity E_i dynamically logs information about each possible target $\mathcal{T}_j$ from a set of targets. Note that we do not consider any limit on the number of tracked targets and so no limit for j. Targets are identifiable by a unique identifier ID_j, and choice of the targets is application-dependent, e.g., in the "running example" in Section 1, a target could be a daily *Security log* of a system. The discussion about how a target could be specified is out of the scope of the paper. The target identifier could be any pre-defined string, such as *Security.evtx* in our running example. We also assume that each E_i stores *at most one unique* dataset $D_{i,j}$ for each target identifier ID_j (i.e., each $D_{i,j}$ is identifiable by a pair $\{E_i, ID_j\}$). In our model, we do not assume any a priori agreement between the entities on the targets that they are going to track (i.e., the datasets that they are going to generate), but we assume that datasets/targets are identifiable through their (global) unique identifier ID_j known by all the entities.

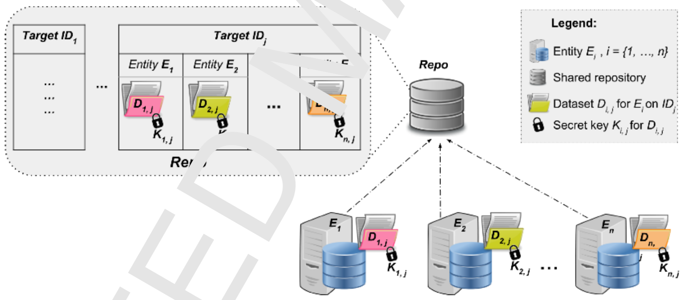

Fig. 2: Simple overview of the system model

Data Sharing Model:

As it can be seen in Figure 2, we assume that entities store their *"encrypted"* datasets in a shared repository, we call it *Repo*. In fact, following [3,4], the use of shared repository is just for presentation simplicity and is not necessary. Instead, the collaborating entities can broadcast their encrypted datasets, which obviously will impose communication overhead, though this choice is application-dependent and does not hinder our proposal. We assume that the stored data in the Repo is organized based on two indices: i) Target identifier,

ID_j, as primary index, and ii) Entity name (or identifier), E_i, as secondary index (see top left part of Figure 2).

Moreover, we assume that symmetric (e.g., AES [1]) and asymmetric (e.g., IBE [6]) encryption algorithms, as well as Threshold Secret Sharing [31], and Pedersen key generation [28] schemes are available in the system to be used by the entities. Therefore, we consider each entity E_i to encrypt the content of each dataset $D_{i,j}$, for target identity ID_j, using a fast symmetric encryption method (e.g., AES). To do so, E_i uses a distinct key $K_{i,j}$ per dataset that is *only* known by the entity E_i itself, and is computed using the following function:

$$K_{i,j} = PRF(S_i, ID_j), \tag{1}$$

where S_i is a random secret chosen by E_i, and PRF is a secure pseudo-random function. We consider such a key derivation method in order to enable entities to compute per target keys on-the-fly without the need for complex stateful key management methods. Then each entity sends an IBE-encrypted version of the target-specific key, i.e., $\mathbb{E}_{ID_j}(K_{i,j})$, to the shared repository.

Furthermore, we assume the Pedersen key generation scheme [28] will be used in the system in order to construct a unique shared master secret $\mathfrak{s}$ (which is not disclosed to any entity). And each target $\mathcal{T}_j$ with identity ID_j will be associated with an IBE private key $H(ID_j)^{\mathfrak{s}}$, where H is a cryptographic hash function mapping the ID_j into a point of a cyclic group G_p. Each entity E_i is able to compute its own *share* of the master secret $\mathfrak{s}$, the global public key $g^{\mathfrak{s}}$, as well as its share of the IBE private key associated to each target, i.e., $H(ID_j)^{x_i}$, where x_i is the local per party share (for more details on x_i refer to Section 3.1). All the details regarding the key generation and sharing is explained in Section 3.

Data Disclosure Model

Every entity in our model is in charge of evaluating its own status and correspondingly deciding when to share the credentials related to one (or more) of its targets, by triggering a *"disclosure signal"*. In particular, when an entity E_i decides to disclose its dataset $D_{i,j}$ on the target identity ID_j, it sends its own share, $H(ID_j)^{x_i}$, of the IBE private key associated to target ID_j. When the Repo receives required number of private key shares for target ID_j which satisfy the pre-defined access policy on $\mathcal{T}_j$, it will be able to reconstruct the $H(ID_j)^{\mathfrak{s}}$ and access all the datasetes related to target ID_j generated by all the entities (as we detail in Section 3). This way, we guarantee that the encrypted datasets will be disclosed "if and only if" the pre-defined condition is satisfied, and neither the shared Repo, not the colluding entities are able to access the datasets.

To elaborate more on this matter consider our running example (see Figure 1). Each bank in the third level (e.g., *Bank* A_i where $i \in \{1..n\}$) is in charge of assessing its local security status and storing system *Security Log*. Each A_i upon recognizing an attack indicator, triggers a *"disclosure signal"* and sends its own share of the secret related to its encrypted *Security Log* (which is identified by *Security.evtx* identifier) to a high level corresponding authority or repository (e.g., *Bank A*, or even the central bank). Upon reception of sufficient number of key shares on the *Security.evtx* by the collaborating entities, the Repo will be able to recover the IBE private key associated to the *Security.evtx* and access the *Security Log* of all the collaborating banks. Obviously, if an entity E_k (for any reason) does not gather information about a target $\mathcal{T}_j$, will neither send corresponding dataset $D_{k,j}$ to the Repo, nor participate in the disclosure policy of $\mathcal{T}_j$ dataset.

2.2 Attack Model

In our attack model, we consider two types of attackers: *external* and *internal* attackers. (i) *External attackers* are the external entities that do not collaborate in our protocol. We consider this adversary as a *weak attacker*, since the only capability of the attacker is eavesdropping the communication between the collaborating entities, and between the entities and the shared repository. A secure protocol against external attacker should not leak any information to the eavesdropper. (ii) In contrary, we consider the *internal attackers* to be *strong attackers* and our main focus in this work is to strengthen the security of the proposed scheme against these attackers.

We consider the *internal attacker* to be of four types, as we explain in the following.

(1) *Untrusted shared repository:* We consider the shared repository to be untrusted. Therefore, it attempts to gain information about the encrypted datasets that it receives from the entities. A desirable secure protocol should be resistance against this adversary, i.e., a security requirement is that the shared repository "must not" be able to decrypt any of the datasets unless the disclosure condition (the pre-defined policy) is met by the entities.

(2) *Honest-but-curious collaborating entities:* We assume the collaborating entities to be honest-but-curious adversaries, meaning that they honestly follow the protocol (as we will explain in Section 3) in generating their own shares of the secret and distributing the required public parameters. However, they are curious to obtain information about other entities' dataset on the same or different targets. To elaborate more on this attacker consider Figure 2. Assume that E_1 has two datasets $D_{1,1}$ and $D_{1,2}$ on the targets with ID_1 and ID_2, respectively. Moreover, E_2 has stored two datasets $D_{2,1}$ and $D_{2,3}$ on the targets with ID_1 and ID_3, respectively. Now assume

that E_1 is *"curious"* to access the data stored in $D_{2,1}$ and $D_{2,3}$. In this example, E_1 and E_2 both track the target ID_1. So, a possible attack from E_1 on $D_{2,1}$ is to use its own secret on ID_1 to have unauthorized access to the $D_{2,1}$. Another possible attack from E_1 to $D_{2,3}$, is as follows: assume that the entities agreed to reveal the dataset related to ID_1. Now, E_1 tries to use the revealed share of the E_2's secret to gain information about the $D_{2,3}$.
Therefore, two security requirements regarding this attacker are: (a) two entities who track the same target "must not" be able to access each other's dataset unless the disclosure condition is met; (b) the disclosure of one dataset "must not" reveal any information about another dataset.

(3) *Colluding entities:* As the third attack, we consider a set of entities where each of them does not satisfy a dataset disclosure's condition per se. Hence, they collude with each other to satisfy the disclosure policy (e.g., by sending a false "disclosure signal") in order to have unauthorized access to other entities' datatset. Now, we have two scenarios: (a) the colluding entities *do not* satisfy the disclosure policy, and (b) the colluding entities *satisfy* the disclosure policy. Therefore, security requirement related to this attacker is: a set of colluding entities "must not" be able to access information about the dataset and the master secrets.

(4) *Cheating entity on the secret.* The last attacker that we consider is an internal entity that sends a fake/altered shares of the secret secret as its share to the Repo (or other entities) to avoid letting disclosure of its dataset. Therefore, the security requirement is that "no" collaborating entity can submit an incorrect share of a secret.

2.3 Background on the Cryptographic Algorithms

In the proposed scheme, we take advantage of Identity-Based Encryption (IBE), and Linear Secret Sharing (LSS) schemes. In the following we provide required background knowledge on these two schemes.

2.3.1 Identity-Based Encryption

Identity-Based Encryption proposed by Boneh and Franklin [6, 7] is a public key encryption that allows the data owner to encrypt the data using an arbitrary string, *ID*, (as the public key). This way, any pair of users are able to securely communicate without exchanging any key materials (i.e., public and private keys), and without the need to involve any trusted third party for the key management purpose.

An IBE scheme is composed of four functions [7]:

- **Setup:** takes as input a security parameter; and outputs a set of system *public* parameters, $\mathcal{P}$, and a *private* master key, MK, that is known only by the key generator. The system public parameters include a description of a finite message space $\mathcal{M}$, and a description of a finite ciphertext space $\mathcal{C}$.
- **Extract:** takes as input $\mathcal{P}$, the master key MK, and an arbitrary string $ID \in \{0,1\}^*$; and outputs a private key SK. In particular, this function takes the ID as a public key, and extracts the corresponding private key SK.
- **Encrypt:** takes as input $\mathcal{P}$, an ID, and a message $m \in \mathcal{M}$; and outputs a ciphertext $CT \in \mathcal{C}$.
- **Decrypt:** takes as input $\mathcal{P}$, a ciphertext CT, and a private key SK; and outputs the message m.

In an IBE scheme, if SK is a private key that is generated by the **Extract** algorithm for the string ID as the public key, then the corresponding encryption and decryption functions must satisfy the following consistency constraint:

$$\forall m \in \mathcal{M} : \mathbf{Decrypt}(\mathcal{P}, CT, SK) = m \quad \text{where} \quad CT = \mathbf{Encrypt}(\mathcal{P}, ID, m)$$

2.3.2 Linear Secret Sharing

Secret sharing scheme, first introduced by Shamir [31] as *threshold secret sharing*, is a building block for several cryptographic methods and secure protocols, such as attribute-based encryption, and multiparty computation. In a (t, n) "threshold" secret sharing scheme [31], a dealer who has a secret s, divides the secret into n pieces and distributes shares of his secret to n parties. Any subset of parties whose cardinality is greater or equal to a predefined threshold, t (where $1 \leq t \leq n$), can reconstruct the s from its shares. However, knowledge of any $t-1$ or less shares of the secret reveals *"no"* information about the s. A generalization of the threshold secret sharing would be distributing the shares of s based on an access structure $\mathcal{A}$ (i.e., a subset of parties). Such a secret sharing scheme satisfies the following conditions [2]: i) Correctness: any subset of $\mathcal{A}$ (authorized parties) can reconstruct the secret from its shares; and ii) Perfect privacy: any subset of parties that is not in $\mathcal{A}$ (unauthorized parties) cannot gain *any* information about the secret. A "linear" secret sharing scheme (LSSS) is defined over a finite field $\mathbb{F}$. The dealer chooses a secret which is an element of the $\mathbb{F}$, and the shares of the secret are vectors over $\mathbb{F}$. The shares are computed using some independent random field elements (chosen by the dealer) while applying a linear mapping to the secret [2].

3 Proposed Approach

Our proposal for conditional collaborative data sharing composed of two phases: 1) offline setup phase, and 2) online credential and dataset management. The main difference between our proposal and the state-of-the-art LSSS [2] that *might be* considered by a layman reader as a similar approach for collaborative data sharing is the *"conditionality"* requirement. This requirement, as we explained in Section 1 and Section 2 (second attacker model), is that disclosure of a dataset related to target identifier ID_j *must not* reveal any information related to *any other* target identifier ID_k. Therefore, our proposal is different from the state-of-the-art in that, in our approach we consider *unique* dynamic keys for each target (adopting IBE [6, 7]), that can be defined at runtime. Further, such keys will be mixed with exiting secret sharing schemes in order to provide distributed collaborative data sharing. We explain the details in the following.

3.1 Offline Setup

In the setup phase, the participating entities, E_i where $i \in \{1..n\}$, share a secret according to the considered access structure $\mathcal{A}$. The entities agree on the following public parameters:

- Two large primes p and q such that q divides $p-1$;
- An $m \times \ell$ access matrix[1] $\mathcal{A}$ on the participating entities, representing the specified policy to access the secret;
- A cyclic group G_p of prime order p, and a generator $g \in G_p$ for the group. The group G_p is specifically chosen as the domain of a non degenerative bilinear map $e : G_p \times G_p \rightarrow G_T$.

Then, each party E_i performs the following steps:

1. Chooses a random secret $\sigma_i \in Z_q$ over the ring of integers modulo prime q;
2. Chooses a random vector $\mathbf{v}_i \in Z_q^\ell$ with σ_i as first entry;
3. For each access matrix row j, computes the share $w_{i,j} = \mathcal{A}_j \cdot \mathbf{v}_i$, and sends (through a secure unicast) it to party associated to the row j;
4. Computes $g^{\sigma_i} \in G_p$, and broadcasts it to all entities.

After all, none of the entities knows the shared *master secret* $\mathfrak{s}$, where $\mathfrak{s} = \sum_{i=1}^{n} \sigma_i$. Rather, each entity is able to compute the following important quantities:

[1] In general, depending on the access structure, m can be greater than the number of domains n (i.e., an entity may require to be given multiple shares for satisfying the access policy). Here, for presentation simplicity we non restrictively consider $m = n$.

- The *global public key* $g^{\mathfrak{s}} = \prod_{i=1}^{n} g^{\sigma_i} \in G_p$;
- The *local per entity share* $x_j = \sum_{i=1}^{n} w_{i,j} \in Z_q$.

The system is secure as long as the master secret $\mathfrak{s}$ remains unknown to all the involved parties. In essence, the master secret will never be reconstructed, and we will reconstruct quantities of type $H(\mathcal{ID}_j)^{\mathfrak{s}}$, namely identity based private keys associated to target identity $H(\mathcal{ID}_j)$.

3.2 Online Credential and Dataset Management

Monitoring a new target: We recall that an entity E_i encrypts its dataset associated to a target ID_j , using a pseudo-random key $K_{i,j}$ on-the-fly generated according to (1). Each entity upon deciding to generate a dataset regarding a target $\mathcal{T}_j$, with identity ID_j, must deliver suitable cryptographic information to the shared repository for decrypting the dataset. In order to do so, E_i transmits once-for-all to the Repo (or once everytime it changes its own secret S_i used in Equation (1) an IBE-encrypted version $\mathbb{E}_{ID_j}(K_{i,j})$ of the key $K_{i,j}$. The $\mathbb{E}_{ID_j}(K_{i,j})$ is constructed using the IBE method [6], where the identity (IBE public key) is the string ID_j, and the PKG's public key is the $g^{\mathfrak{s}}$ that is computed in the offline setup phase (Section 3.1). The adopted IBE equation is as follows:

$$\mathbb{E}_{ID_j}(K_{i,j}) = (g^r, K_{i,j} \oplus H_2(e(H(ID_j)^r, g^{\mathfrak{s}}))) \tag{2}$$

where $r \in Z_q$ is a random value, $e : G_p \times G_p \to G_T$ is the agreed bilinear map, $H : \{0,1\}^* \to G_p$ is a cryptographic hash mapping a target name ID_j into a point of the group G_p, and $H_2 : G_T \to \{0,1\}^h$ is a cryptographic hash mapping a point of the group G_T into a string of same size as $K_{i,j}$.

Dataset disclosure: Whenever an entity E_i, desires to share its dataset related to the target identity ID_j with the other entities, it generates a *disclosure signal* as follows:

$$Signal_{i,j} = H(ID_j)^{x_i} \in G_p \tag{3}$$

Upon reception of a sufficient number of shares (satisfying the access policy) delivered inside the signals $Signal_{i,j}$ for ID_j, the repo computes the coefficient c_i such that

$$\sum_{i \in \mathcal{Q}} c_i \cdot A_i = (1, 0, ..., 0) \tag{4}$$

where $\mathcal{Q}$ is the set of secret shares for the target ID_j received from the entities, and A_i is the row of the access matrix associated to the secret share disclosed by the entity E_i. Having sufficient number of secret shares and their corresponding coefficients, the Repo can reconstruct the IBE private key associated to the target ID_j , i.e., $H(ID_j)^{\mathfrak{s}}$ as follows:

$$H(ID_j)^{\mathfrak{s}} = \prod_{i\in\mathcal{Q}} signal_{i,k}^{c_i} = \prod_{i\in\mathcal{Q}} H(ID_j)^{c_i x_i} = H(ID_j)^{\sum_{i\in\mathcal{Q}} c_i x_i} \quad (5)$$

Finally, *all* the IBE-encrypted keys $K_{*,j}$, used to encrypt each entities dataset, can be decrypted by computing

$$H_2(e(H(ID_j)^{\mathfrak{s}}, g^r)) \quad (6)$$

Due the properties of bilinear pairings, we have

$$H_2(e(H(ID_j)^{\mathfrak{s}}, g^r)) = H_2(e(H(ID_j)^r, g^{\mathfrak{s}})), \quad (7)$$

Considering the Equation (2), we can now recover all the keys (i.e., $K_{*,j}$):

$$K_{i,j} \oplus H_2(e(H(ID_j)^r, g^{\mathfrak{s}})) \oplus H_2(e(H(ID_j)^{\mathfrak{s}}, g^r)) = K_{i,j} \quad (8)$$

The keys $K_{*,j}$ now can be used to decrypt all the datasets associated to the target ID_j. Hence, even the dataset of the entities that have not decided to share their dataset and have not sent the relevant *Signal* can be decrypted (due to satisfying the pre-defined access policy during the offline setup phase).

3.3 Security Discussion

In this section, we provide a security analysis of our proposed approach against the considered attack model in Section 2.2. First, considering an *external attacker*, eavesdropping could be easily eliminated by assuming a secure communication channel between each pair of entities, and between an entity and the shared repository. Therefore, our proposed approach is safe against such an attacker.

Regarding an *internal attacker* and considered four types of attacks (refer to Section 2.2), we discuss the resilience of our proposed approach against each of these attacks in the following.

3.3.1 Untrusted shared repository

Due to the usage of IBE for encrypting the datasets based on their target identifier (i.e., per ID_j), for each dataset we have a unique IBE private key, $H(ID_j)^{\mathfrak{s}}$. The shared repository (Repo) would only be able to decrypt the datasets if either the $\mathfrak{s}$ is disclosed, or $H(ID_j)^{\mathfrak{s}}$ has been reconstructed as a whole. On the one hand, as we explained earlier, the master secret $\mathfrak{s}$ will *never* be reconstructed, and the Repo will not have access to the $\mathfrak{s}$. On the other hand, if the shares of the IBE private key, $H(ID_j)^{x_i}$, that the Repo receives do not satisfy the access policy, the Repo would not be able to recover the $H(ID_j)^{\mathfrak{s}}$. This feature is due to the usage of LSSS for sharing the master secret $\mathfrak{s}$ between the collaborating parties. Therefore, the shared repository is not able to decrypt the datasets *unless* a sufficient number of entities send their shares of the IBE private key; hence the proposed approach meets the security requirements related to an "untrusted shared repository" attack.

3.3.2 Honest-but-curious collaborating entities

In Section 2.2 we defined two security requirements. (a) considering any subset Ω' of the entities (unauthorized parties) that are not involved in the dataset disclosure access policy, should not be able to gain any information about the secret [2]. This requirement is satisfied due to the usage of LSSS. Therefore, a malicious entity cannot access other entity's share and datasets unless the disclosure condition is met.

(b) Based on the features of IBE [6], it follows that the disclosure of the IBE private key $H(ID_j)^{\mathfrak{s}}$ associated to a target ID_j does not reveal any information about the remaining targets.

3.3.3 Colluding entities

In Section 2.2 we considered two different scenarios. First case is a set Ω' of colluding entities that do not satisfy the disclosure policy. In this case, the outcome of their collusion will be a key $\mathfrak{s}' = \sum_{i\in\Omega'} c'_i x'_i$, which would be $\mathfrak{s}' = \mathfrak{s}+\varepsilon$ and could not be used to decrypt the dataset. In the second case, in which the colluding entities "satisfy" the disclosure policy on a dataset, they will obviously be able to reconstruct the $H(ID_j)^{\mathfrak{s}}$ and decrypt the corresponding dataset.

3.3.4 Cheating entity on the secret

In this section, we focus on cheating parties who aim to break the protocol by delivering fake/altered shares during the crucial offline setup operation[2] (described in Section 3.1). We show that how our proposed system could be further improved to cope with this kind of attack. Note that such an attacker

[2] Note that fake shares are critical only during the offline setup operation. Indeed, a party wishing to cheat during the *online* operation, i.e., sending a fake signal $H(ID_j)^{x_i}$ in Equation (3), would be considered as a party who "refuses" to send her share. In other words, knowing that a signal is fake does not solve the problem of disclosing a target dataset - in any case the dataset would be decrypted only when a sufficient number of *valid* signals (Equation (3)) are received.

may play havoc with the proposed protocol. Undetected injection of fake shares $w_{i,j}$ -with reference to the notation introduced in Section 3.1- by a party i would affect *all* of the local per-entity shares x_j that are built upon such fake shares. In this case, it will be impossible to reconstruct the key for any target dataset as detailed in Section 3.2.

A possible solution against such an attacker would be extending the protocol by permitting each party participating in the setup phase to explicitly verify each (fraction of the) share received during the setup. Then, each party could follow up with its own local share computation only when the correctness of the received information is guaranteed. This goal can be accomplished by adapting a non-interactive verifiable secret sharing scheme to our proposed offline setup protocol. To this purpose, in the remainder of this section we show how a basic solution based on Feldman's commitments [12] can be designed. Our proposed approach is simple and perfectly compatible with the Pedersen Distributed Key Generation scheme [28], but it is more general than previous proposals restricted to threshold-based secret sharing. Our proposed solution does not require any special structure or restriction in the access matrix. We here focus on such a baseline approach and leave extensions to more elaborated zero-knowledge approaches for further work. In fact, despite some known limitations [14] a Feldman-based commitment within a distributed Pedersen-type protocol is considered reasonable and, for instance, it has been employed also for distributing the Private Key Generator in IBE schemes [7].

Let us first recall, from Section 3.1, that each participating entity E_i chooses a random secret $\sigma_i \in Z_q$ and a random vector in Z_q^ℓ, with σ_i as the first entry. Let us denote such a vector as

$$\mathbf{v}_i = [\sigma_i, r_{2,i}, \cdots, r_{l,i}].$$

With such a notation, the computation of the share $w_{i,j}$ for each access matrix row j can be rewritten as

$$w_{i,j} = a_{1,j}\sigma_i + \sum_{k=2}^{l} a_{k,j} \cdot r_{k,i}$$

The offline setup phase, described in Section 3.1, now requires each participating entity E_i to compute and broadcast $c_{1,i} = g^{\sigma_i} \in G_p$. To permit verifiability of the shares $w_{i,j}$, each party E_i have to *further* compute and broadcast the additional $l-1$ commitments $c_{k,i} = g^{r_{k,i}} \in G_p$, for all $k = 2, \cdots l$.

The participating entity E_i is now given the possibility to verify that the above computed share $w_{i,j}$ associated to row j of the access matrix is indeed a valid share, by checking whether the following equality holds:

$$\prod_{k=1}^{l} c_k^{a_{k,j}} \stackrel{?}{=} g^{w_{i,i}}$$

In fact, if the protocol is correctly executed and no shares are altered, it readily follows that:

$$\prod_{k=1}^{l} c_k^{a_{k,j}} = (g^{\sigma_i})^{a_{1,j}} \cdot \prod_{k=2}^{l} (g^{r_{k,i}})^{a_{k,j}} = g^{\left(a_{1,j}\sigma_i + \sum_{k=2}^{l} a_{k,j} \cdot r_{k,i}\right)} = g^{w_{i,i}}. \quad (9)$$

Otherwise, if the Equation 9 does not hold, it means that one (or more) of the collaborating entities has shared a fake/altered share of the secret.

4 Use Case Example

In this section we explain two use case scenarios for which we adopt our proposed conditional collaborative data sharing approach. As we discuss in Section 4.1, during a global or large scale cyber attack scenario, intelligently sharing the threat feeds provides security analysts with threat intelligence. Cyber Threat Intelligence (CTI) helps security analysts, victims, and defenders to gain knowledge about adversaries, their intentions and methods [21]. This knowledge is achieved by processing the shared information regarding the Indicators of Compromise (IoCs), e.g., through sharing system logs, security alerts, network traffic information and so on [30]. An important challenge in the area of CTI is legal issues regarding the sharing of CTI-related information, specially the information within the government's possession and within the possession of the private sector [26]. As an example, assume a scenario in which Federal Bureau of Investigation (FBI) provides the privately owned banks with the IP addresses that are known to deliver ransomware to financial sector [26], or an infected bank provides such information to other banks.

The second use case (in Section 4.2) is borrowed from our previous work [3], in which we show how the proposed approach helps in mitigating a distributed denial of service attack through whitelisting legitimate traffic.

4.1 Ransomware Threat Intelligence

In order to make this use case more clear let us consider the WannaCry Ransomware attack, that emerged in May 2017. WannaCry exploited a vulnerability in Windows machines (Windows SMB remote code execution vulnerability) and was able to spread itself across an organization's network without user intervention [32]. We believe that it would be possible to mitigate the WannaCry attack (and similar worm-like malwares) and block its spreading throughout an organization's network by sharing the attack indicators and system logs within the organization (and with different organizations).

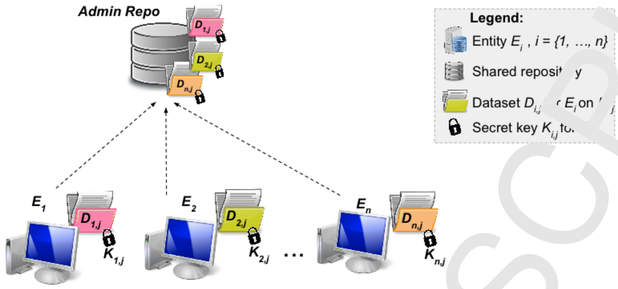

Fig. 3: System log (i.e., IoCs regarding a ransomware attack) sharing

In this use case, we consider two scenarios. 1) *a fine grained data sharing scenario*, in which each individual device in an organization, e.g., personal computers in wards of a hospital, in each office of a department in a university, or in each branch of a bank, is considered as a collaborating entity (e.g., Figure 3). In this scenario, each device stores logs (e.g., network traffic, or application information) based on a specific index (which should be agreed a priori). For example, each device logs the requested DNS resolution, or contacted websites, or list of software updates. Moreover, there is usually a central logging database that stores an encrypted version of the logs of each device (though it can be saved locally on the user's system in a shared folder). Upon receiving an indication of a cyber attack, the security analyst of an organization should have access to all the encrypted logs in order to perform analysis of the attack entry point. For example, having access to the contacted IP addresses by each device in an organization, the analyst might be able to detect the Command and Control (C2) server which the malware connects to, and blacklist the IP address. However, considering a university example, all the staff and professors may not wish to share their private information on their activities, e.g., contacted web sites. In the WannaCry attack scenario, this has happened for those computers that were using outdated/unpatched Windows. While, if at the first point of recognizing the indicators of compromise, the first victim (or a number of initial victims) had shared the vulnerability and attack information within their organization, or with other organizations, it could be possible to search for those vulnerabilities and mitigate it before being distributed through the organization's network.

2) *a coarse-grained data sharing scenario:* in which each private hospital, each department in a university, or each private bank plays the role of a collaborating entity (e.g., Figure 4). Our proposed private data sharing method is of great help in such scenarios. Imagine each hospital in a city or country being an entity. All of the hospitals log their security related information (e.g., alerts, DNS requests, software patches, etc) and store an encrypted copy of them in a shared repository. Moreover, assume that the dataset disclosure

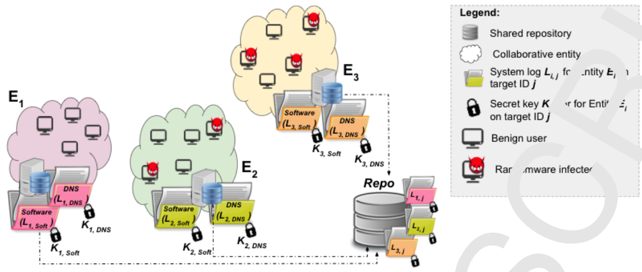

Fig. 4: Ransomware threat data sharing in large scale scenario

condition is that at least two hospitals trigger the *Signal* alert. Upon recognition of an attack, E_2 and E_3 in Figure 4 trigger the data disclosure *Signal*. As soon as receiving sufficient number of *Signals*, the admin of the shared repository is able to recover the encryption key and access the logged information by all the entities. In fact, Equation 8 which leads to disclosure of all the $K_{*,j}$ keys regarding a target ID_j (recalling that a target could be any of these logs which is identifiable through a unique public index) provides an important feature for emergency scenarios. This is useful in situations where the log of all the hospitals should be investigated (for vulnerability analysis against the ransomware attack), while some of the hospitals have not detected an attack (e.g., E_1 in Figure 4), or their IT manager (or whoever responsible) is not available to share their credentials or for any reason (which could be privacy issues) they are not able/willing to share their logged information. In such a scenario, in order to mitigate a large scale cyber attack, and due to the fact that the disclosure policy has been agreed by all the participating entities (i.e., hospitals) during the initial setup, the unavailability or unwillingness of some entities does not matter, since the shared repository will be able to access those information for further investigation.

4.2 Distributed Denial of Service Attack Mitigation

In this use case example we consider a DDoS mitigation approach through establishing and sharing whitelists of *good* addresses that should not be filtered under DDoS attack conditions. Figure 5 shows an example scenario in which the target has been considered to be a web server, identifiable by its unique *name*. Each domain generates a whitelist for any target that it is willing to monitor. Each whitelist includes a set of IP addresses of benign users whose access to the target server should be guaranteed even in the presence of a DDoS attack. However, whitelists require domains to share sensible information. Our proposed approach helps in managing a large num-

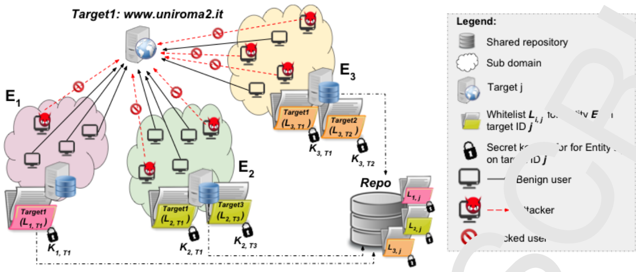

Fig. 5: DDoS mitigation via whitelist example

ber of fine-grained organized whitelists and selectively disclose them under precisely specified attack conditions.

Whenever a domain E_i, based on its internal monitoring of a target ID_j, detects that the target is under DDoS attack, it triggers the *disclosure signal* and shares its own share of the private key associated to the target ID_j . When "sufficient" number of *Signals* which satisfy the access policy (e.g., $\langle E_1 \wedge E_3 \rangle \vee E_2$) is received, *all* the domains involved in the system operation will be able to retrieve (i.e., decrypt) all the whitelists associated to the target ID_j, and will be able to instruct their

firewalls accordingly (e.g., block all traffic except the whitelisted IP addresses).

5 Performance Evaluation

In this section, we present the performance evaluation of our approach borrowed from our previous work [3]. As our proposal is mostly a cryptographic approach and its performance assessment highly relates to specific application scenario, in this section we present the computational time required to perform each cryptographic primitive. The performance evaluation is performed on an Intel Xeon X5650 (2.67 GHz, 6 cores) equipped with 16 GB RAM and an Ubuntu Server operating system. We implemented the system using C++ programming language. We adopted XML to structure and transport exported data between different entities. For cryptographic operations (e.g., hashing with SHA-256, symmetric encryption/decryption with AES-128), we used the OpenSSL library. For pairings, elliptic curve generation, elliptic curve arithmetic, and hash functions required by IBE, we adopted Pairing Based Cryptography (PBC) library [23].

Even if our current preliminary implementation has not been specifically optimized for performance and multi-core exploitation, results are very promising and suggest the feasibility of our system in a realistic setting. Table 1 depicts performance analysis of the cryptographic primitives. The re-

Function	**Computation Time (ms)**
IBE encryption, Equation (2)	2.2051
IBE decryption, Equation (6)	2.3070
Symmetric key derivation, Equation (1)	0.0086
Key share computation, Equation (3)	0.3101
AES-cbc-256 encryption	0.00927
AES-cbc-256 decryption	0.01334
Key Reconstruction, Equation (5) (1 share)	0.[illegible]5697
Key Reconstruction, Equation (5) (2 shares)	0.98[illegible]11
Key Reconstruction, Equation (5) (3 shares)	1.18959

Table 2: Cryptographic operation performance analysis

sults are obtained by averaging 1M executions of a same primitive. As it can be seen, IBE-related operations impose the highest computation overhead, especially those involving pairings. While, Symmetric encryption of datasets is the least expensive operation. However, as encryption of datasets should be performed in real time, it may become a bottleneck. The IBE-related operations are actually computed only once for every newly considered target ID_j (we assume there is not a periodic rekeying process per target). Similarly, key reconstruction functions per target should be performed once, only if there is an ongoing attack. Therefore, the fact that complexity grows with the number of shares is not an issue. It should be noted that complexity of the proposal does not rely on the number of entities involved in the system, this will only affect the initial offline setup phase. While, the number of shares that are required to satisfy the access policy affects performance of the system. It is evident that if we consider the *threshold-based secret sharing*, the increase in the threshold value (and consequently number of required shares) will lead to the increase in the complexity of Equation (5) which reconstructs the IBE private key associated to a secret. While, if we consider the *policy-based secret sharing*, the complexity of Equation (5) depends on the complexity of the access policy. For example, if we consider a policy $\cup_{i \in N} E_i$, the number of required shares for key reconstruction would be one; however, if we consider the following access policy $(E_1 \vee E_2) \wedge E_3 \wedge E_4 \wedge E_5$, the number of required shares to reconstruct the key would be four. The performance of our system also relies on the rate of considering new targets (ID_j), since each entity requires to perform Equation (1) and the IBE encryption (Equation (2)) and deliver the decryption key to the Repo. However, these operations can be offloaded and scaled to gbps speed [5].

6 Conclusion

The *conditional collaborative private data sharing* method proposed in this paper is a cryptographical method that is applicable to secure privacy-preserving collaborative data sharing scenarios. In such scenarios a priori known entities would *only* like to share their encrypted data conditionally on special occasions, defining uniform access structures. In particular, our proposed method adopts a combination of *Identity-Based Encryption (IBE)* and *Linear Secret Sharing(LSS)* schemes in order to provide scalability, and efficiency. Our proposal is *scalable* in the sense that there is no limit on the number of *distinct* datasets that each entity is willing to share, since the datasets are independently encrypted using IBE, and their unique identi

er is the public key. The proposed approach is *efficient* in the sense that there is no need for any interaction between different entities on deciding encryption/decrytion keys for each dataset, as long as they consider a *unique public identifier* for each dataset that is publicly available to all the collaborative entities. The important distinguishing feature of the proposed approach compared to the existing collaborative data sharing methods is that, the master secret will never be revealed/reconstructed by any entity or central shared storage; rather, some quantities of the identity-based private keys associated to each dataset will be shared and reconstructed by the collaborative entities. Therefore, disclosure of one dataset D_j , will not reveal any information about the other datasets with different identifiers (i.e., ID), eliminating the need for perform re-keying process for other undisclosed datasets.

References

1. AES– advanced encryption standard. URL http://www.nist.gov/aes
2. Beimel, A.: Secret-sharing schemes: a survey. In: Coding and Cryptology, pp. 11–46 (2011)
3. Bianchi, G., Rajabi, H., Caponi, A., Picierro, G.: Conditional disclosure of encrypted whitelists for ddos attack mitigation. In: Proc. of the 2013 GLOBECOM Workshops (GC Wkshps), pp. 200–206. IEEE (2013)
4. Bianchi, G., Rajabi, H., Sgorlon, M.: Enabling conditional cross-domain data sharing via a cryptographic approach. In: Proc. of the 5th International Conference on Internet Multimedia Systems Architecture and Application, IMSAA'11, pp. 1–6. IEEE (2011)
5. Bianchi, G., Wolkerstorfer, J., Teofili, S., Gojmerac, I., Jung, O.: Feasibility of wire-speed hardware-based conditional per-flow encryption for on-the-fly protection of monitored traffic. In: Future Network and Mobile Summit, 2010, pp. 1–9. IEEE (2010)
6. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Advances in Cryptology CRYPTO 2001, pp. 213–229 (2001)
7. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. SIAM journal on computing **32**(3), 586–615 (2003)
8. Cinque, M., Cotroneo, D., Esposito, C., Fiorentino, M., Russo, S.: A reliable crisis information system to share data after the event of a large-scale disaster. In: Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on, pp. 941–946. IEEE (2015)
9. Clifton, C., Kantarcioğlu, M., Doan, A., Schadow, G., Vaidya, J., Elmagarmid, A., Suciu, D.: Privacy-preserving data integration and sharing. In: Proceedings of the 9th

ACM SIGMOD workshop on Research issues in data mining and knowledge discovery, pp. 19–26. ACM (2004)
10. Danezis, G., Fournet, C., Kohlweiss, M., Zanella-Béguelin, S.: Smart meter aggregation via secret-sharing. In: Proc. of the first ACM workshop on Smart energy grid security, pp. 75–80. ACM (2013)
11. Dunning, L.A., Kresman, R.: Privacy preserving data sharing with anonymous id assignment. IEEE Transactions on Information Forensics and Security **8**(2), 402–413 (2013)
12. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: Foundations of Computer Science, 1987., 28th Annual Symposium on, pp. 427–438. IEEE (1987)
13. Freudiger, J., De Cristofaro, E., Brito, A.E.: Controlled data sharing for collaborative predictive blacklisting. In: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, pp. 327–349. Springer (2015)
14. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. In: International Conference on the Theory and Applications of Cryptographic Techniques, pp. 295–310. Springer (1999)
15. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM conference on Computer and communications security, pp. 89–98. Acm (2006)
16. House, T.W.: Executive order – promoting private sector cybersecurity information sharing (2015). URL `https://www.whitehouse.gov/the-press-office/2015/02/13/executive-order-promoting-private-sector-cybersecurity-information-shari`
17. Huang, Q., Yang, Y., Fu, J.: Precise: Identity-based private data sharing with conditional proxy re-encryption in online social networks. Future Generation Computer Systems **86**, 1523–1533 (2018)
18. Jiang, L., Guo, D.: Dynamic encrypted data sharing scheme based on conditional proxy broadcast re-encryption for cloud storage. IEEE Access **5**, 13,336–13,345 (2017)
19. Jin, R., He, X., Dai, H.: On the tradeoff between privacy and utility in collaborative intrusion detection systems-a game theoretical approach. In: Proceedings of the Hot Topics in Science of Security: Symposium and Bootcamp, pp. 45–51. ACM (2017)
20. Jin, R., He, X., Dai, H., Dutta, R., Ning, P.: Towards privacy-aware collaborative security: A game-theoretic approach. In: Privacy-Aware Computing (PAC), 2017 IEEE Symposium on, pp. 72–83. IEEE (2017)
21. Jon Friedman, M.B.: Definitive guide to cyber threat intelligence. Tech. rep., CyberEdge Group, LLC (2015)
22. Liang, K., Susilo, W., Liu, J.K.: Privacy-preserving ciphertext multi-sharing control for big data storage. IEEE transactions on information forensics and security **10**(8), 1578–1589 (2015)
23. Lynn., B.: Pbc library. URL `https://crypto.stanford.edu/pbc/`
24. Melis, L., Danezis, G., De Cristofaro, E.: Efficient private statistics with succinct sketches. In: Proc. of the 23nd Annual Network and Distributed System Security Symposium, NDSS'16 (2016)
25. Nabeel, M., Shang, N., Bertino, E.: Privacy preserving policy-based content sharing in public clouds. IEEE Transactions on Knowledge and Data Engineering **25**(11), 2602–2614 (2013)
26. Nolan, A.: Cybersecurity and information sharing: Legal challenges and solutions. Tech. rep., Congressional Research Service - Informing the legislative debate since 1914 (2015)
27. Nuez, D., Agudo, I., Lopez, J.: Proxy re-encryption. Journal of Network and Computer Applications **87**(C), 193–209 (2017)
28. Pedersen, T.P.: A threshold cryptosystem without a trusted party. In: Workshop on the Theory and Application of of Cryptographic Techniques, pp. 522–526. Springer (1991)

29. Peters, F., Menzies, T., Layman, L.: Lace2: better privacy-preserving data sharing for cross project defect prediction. In: Proc. of the 37th International Conference on Software Engineering, *ICSE'15*, vol. 1, pp. 801–811. IEEE Press (2015)
30. Shackleford, D.: Who's using cyberthreat intelligence and how? SANS Institute InfoSec Reading Room (2015)
31. Shamir, A.: How to share a secret. Communications of the ACM pp. 612–613 (1979)
32. Symantec Security Response: What you need to know about the wannacry ransomware (2017). URL `https://www.symantec.com/blogs/threat-intelligence/wannacry-ransomware-attack`

Giuseppe Bianchi is a Full Professor of Networking at the School of Engineering of the University of Rome Tor Vergata since 2007. His research activity includes wireless networks, programmable network systems, privacy and security, performance evaluation, and is documented in more than 200 peer-reviewed international journal and conference papers, having received to date more than 15000 citations (source: Google scholar). He has carried out pioneering research work on WLAN modeling and assessment, and is currently interested in network programmability and security monitoring in both wireless and wired domains. He has been a General or Technical Co-Chair for several major conferences and workshops (IEEE INFOCOM 2014, ACM CoNext 2015, IEEE LANMAN 2016, IEEE WoWMoM 2007 and 2010, ACM WinTech 2011, ACM SRIF 2013,
Track Chair at IEEE PIMRC 2008, etc). He has held general or technical coordination roles in several European Projects (FP6-DISCREET, FP7-FLAVIA, FP7-PRISM, FP7-DEMONS, H2020-BEBA, and H2020-SCISSOR). He
has been (or still is) an Editor for top journals in his field, including the IEEE/ACM Transactions on Networking, the IEEE Transactions on Wireless Communications,
and the IEEE Transactions on Network and Service Management.

Tooska Dargahi is a Lecturer at University of Salford, Manchester, UK. She was a Post-Doc researcher (2016-2017) at CNIT - University of Rome Tor Vergata research unit, Rome, Italy. She received her PhD in Computer Engineering from Azad University, Science and Research Branch, Tehran, Iran in 2014. From 2009 to 2015 she was a Lecturer at Azad University, West Tehran Branch, Tehran, Iran. In 2014, she was a visiting PhD at University of Padua, Italy. Her research interests are in the area of network security, cybersecurity and privacy.

Alberto Caponi received the Master and PhD degrees at the University of Rome “Tor Vergata” in 2011 and 2015, respectively. He joined the Research Unit of Rome of Consorzio Interuniversitario per le Telecomunicazioni (CNIT) in 2016 as a Researcher. He is currently involved in two H2020 research projects (ReCRED and SCISSOR) founded by the European Commission. His main research area is network security and data-centric security using Identity-Based and Attribute-Based schemes for authentication, privacy-preserving and access control. His research interests include Cybersecurity, Network Security and Cryptography.

Mauro Conti is an Associate Professor at the University of Padua, Italy.
He obtained his Ph.D. from Sapienza University of Rome, Italy, in 2009.

After his Ph.D., he was a Post-Doc Researcher at Vrije Universiteit Amsterdam, The Netherlands. In 2011 he joined as Assistant Professor the University of Padua, where he became Associate Professor in 2015. In 2017, he obtained the national habilitation as Full Professor for Computer Science and Computer Engineering. He has been Visiting Researcher at GMU (2008, 2016), UCLA (2010), UCI (2012, 2013, 2014, 2017), TU Darmstadt (2013), UF (2015), and FIU (2015, 2016). He has been awarded with a Marie Curie Fellowship (2012) by the European Commission, and with a Fellowship by the German DAAD (2013). His research is also funded by companies, including Cisco and Intel. His main research interest is in the area of security and privacy. In this area, he published more than 200 papers in topmost international peer-reviewed journals and conference. He is Associate Editor for several journals, including IEEE Communications Surveys & Tutorials, IEEE Transactions on Information Forensics and Security, and IEEE Transactions on Network and Service Management. He was Program Chair for TRUST 2015, ICISS 2016, WiSec 2017, and General Chair for SecureComm 2012 and ACM SACMAT 2013. He is Senior Member of the IEEE.

In this paper:
- We present a conditional collaborative private data sharing protocol that provides a scalable and easy to deploy cryptographical method for data sharing.
- We provide scalability by enabling the data owner to encrypt each dataset with a unique secret key. Therefore, the disclosure of one dataset does not disclose information about the other datasets.
- We provide cryptographically conditioning by permitting the collaborating entities to decrypta specific dataset only if they satisfy a pre-defined policy and recover the relevant secret key.
- We apply a fully distributed process by adopting a combination of Identity-Based Encryption (IBE), and Linear Secret Sharing (LSS) (in particular, Threshold Secret Sharing).