CARDS - Collaborative Audit and Report Data Sharing for A-Posteriori Access Control in DOSNs

Leila Bahri, Barbara Carminati, and Elena Ferrari

STRICT Social Lab, Insubria University, Italy

***Abstract*—Accountability and transparency have been commonly accepted to deter bad acts and to encourage compliance to rules. For this, auditing has been largely, and since ancient times, adopted to ensure the well running of systems and businesses within which duties are governed by set rules. Recently, an a-posteriori approach to data access control has been investigated for information systems as well across number of critical domains (e.g, healthcare systems). Besides, privacy advocates started calling for the necessity of accountability and transparency in managing users' privacy in nowadays connected and proliferated web data. Following this line of thought, we suggest a system for collaborative a-posteriori access control to data dissemination in decentralized online social networks based on reporting and auditing. We demonstrate the usability of our suggested model using a real OSN graph.**

***Index Terms*—Collaborative audit, DOSNs, A-posteriori access control, Data accountability, Collaborative data sharing.**

I. Introduction

Access control (AC) is a main building block in information security that has been commonly approached by a lock-it-to-protect-it frame of thought. In almost all information systems, AC solutions have focused on keeping sensitive resources securely locked in, by ensuring that only authorized entities can unlock their way to them [1]. Users get access to data through authorizations that can be specified according to a variety of AC models. This approach ensures the protection of data as much as the locks are stronger than the will of intruders, and fails short when the shields are broken. Moreover, it works under the assumption that all the granted accesses can be implicitly or explicitly coded into a set of authorizations, and this can be hardly achieved in today web-based collaborative environments. As a result, information security advocates started calling for an alternative approach to data protection, based on accountability, operated by transparency, and regulated by adequate and enforced laws and systems [2][3].

The birth of auditing for accountability can be referenced to very ancient times according to historians of accounting who revealed evidence that "the accounting system in China during the Zhao dynasty (1122-256 BC) included audits of official departments" [4]. Subsequently, accountability and auditing have been a major field and a necessary practice in business as well as in the general legal and social structures of our societies. Indeed, as argued in [2], though transparency and accountability can be not enough to ensure compliance by simply uncovering bad actions and possibly rewarding good ones, they have been the basis for our general respect to most of our legal and social rules. This is mostly because we are implicitly aware that there are records holding us accountable, should we break the rules. As such, a-posteriori AC has been considered for information systems as well. For instance, there are works on audit-based solutions for AC in healthcare systems [5], and others on a-posteriori AC for systems within which some accesses cannot be fully anticipated [7], or for information that requires more than traditional AC enforcement [6].

Online Social Networks (OSNs), given the personal nature and granularity of data shared in them, make one of the scenarios where the standard a-priori AC paradigm better shows its limits, due to the huge user population and to the fact that access to personal or sensitive information is not always under the control of the data owner but can be influenced by the actions of other users in the network (e.g., a user tagging another user in a picture). This problem is pronounced even more seriously in decentralized online social networks (DOSNs).

A DOSN is a system that offers OSN services in a peer to peer manner. The concept of DOSNs aims at bringing back control to OSN users and freeing them from the observance of the central service providers. Indeed, there is no doubt that users' personal data (and hence, their privacy) is the currency used to get advantage of the socializing services offered by current commercialized OSNs [8]. As such, many researchers believe that the solution relies in designing platforms for collaborative service provision. However, one of the still open challenges to DOSNs is access control, that has been mainly addressed using cryptography based solutions [9], [10]. Whilst such solutions might ensure high data security levels, they are also not flexible enough to support the fine granularity and complex access scenarios required for data dissemination in DOSNs [11].

With a belief in the power of accountability and transparency in ensuring better informed data security management and in increasing awareness towards privacy issues, we suggest, in this paper, a deterrent a-posteriori AC mechanism based on collaborative auditing and reporting of data sharing (CARDS) in DOSNs. Instead of deploying hard preventive mechanisms, we suggest an open sharing environment based on collaboration and on trust, where an auditing mechanism coupled with a reputation management system is put in place to encourage good behavior and to deter bad actions. More-

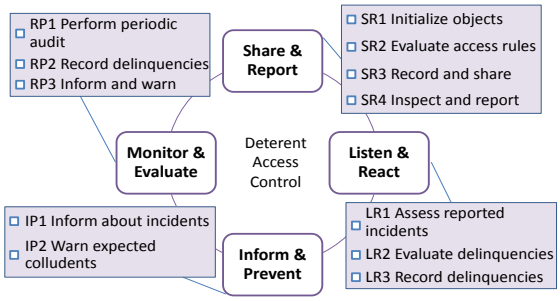

Fig. 1: The four steps of the proposed CARDS framework

over, a proactive activity continuously takes place to prevent continuity of damage in cases of detected bad actions.

The remainder of the paper is structured as follows. In Section II, we provide an overview of the system with its main building blocks; whereas in Section III, we formally define the model. Section IV deals with the security properties and the complexity analysis. In Section V, we present performance experiments using a real world OSN dataset. We survey related work in Section VI. Finally, in Section VII, we conclude the paper and discuss future work.

II. CARDS Overview

In line with AC common definitions, in this paper, we address the problem of granting/denying access to a resource (i.e., *object*) in the system by a requesting entity (i.e., *subject*). More precisely, we target the scenario of DOSNs, where subjects establish friendship links, and collaborate to disseminate their objects over these established relationships without relying on a central data repository/manager. We consider that each object in the system is solely owned by the user that created it (i.e., the owner), and is under the protection of any user that gets access to it (i.e., custodians). Owners specify access requirements for their objects, and the system needs to ensure the objects are disseminated in alignment with their corresponding AC requirements.

Differently from the common apriori approach to AC, in this paper, we adopt an a-posteriori paradigm that follows a framework of four connected processes, depicted in Figure 1. These processes are described as follows:

- **Share & Report:** this process aims at recording data sharing transactions. Data owners specify access rules for their objects by activity SR1 and objects are shared based on evaluating their access rules (activity SR2). Custodians might share an object against its access rule, thus activity SR3 records any sharing transaction taking place. Finally, activity SR4 reports illegitimate sharing transactions and can be performed by custodians, in case they have access to the object's sharing log, or by any entity safekeeping it.
- **Listen & React:** with three activities, LR1, LR2, and LR3, this process concerns the reaction to illegitimate sharing (i.e., *delinquencies*) communicated using activity SR4. It is recommended to have LR1 and LR2 taken by different entities than the one performing SR4, since such a separation of tasks would ensure dual control that provides stronger security guarantees. In fact, dual control requires actions from more than one entity to grant access, basing on the premise that for a breach to happen, all entities need to collude. It is also recommended that these entities do not share a common interest so as to avoid collusion [12].
- **Inform & Prevent:** this process aims at deterring suspected entities from performing further illegitimate sharing of a victimized object. The main challenge is the timeliness to have a preventive action and also the ability to predict future behavior that is to be prevented.
- **Monitor & Evaluate:** the motive of this process is to ensure the cleanliness of the environment under the assumption that not all entities will willingly abide by the reporting mechanism of the first process. This is ensured by three activities that should be carried out periodically on all the nodes in the system.

Based on these processes, we model our suggested system by considering the DOSN users as its major active players. As exemplified on Figure 2, DOSN users establish friendship links with each other such that they are aware of and can only contact their direct friends, create objects, and share them with their fellows based on their privacy preferences. Generally, in OSNs and in DOSNs, privacy preferences are defined following a relationship-based model [1] in the sense that users gain access to information based on the links they establish with its owners. Similarly, in designing our solution, we assume a rule-based model for the formulation of relationship-based privacy settings for an object. That is, upon creating an object, its *owner* assigns to it a relationship-based access rule encoding its privacy requirements and then shares the object, in plain format, with direct eligible friends as allowed by the rule. Users that receive an object (i.e., *custodians*) can share it further with their contacts. To allow for activity SR3 from the CARDS framework (Figure 1), we attach to every object a log that traces its *sharing trajectory* (*Bichon*[1] *Chain* as on Figure 2). A *Bicon Chain* keeps record of every sharing transaction that the object has been subject to through a given path since leaving its owner. The first sharing record is added by the owner when first releasing the object, then records are added to the chain by every custodian who shares the object. The chain should allow an object's receiver to check: 1) whether the sharing transaction by which it received the object is legitimate or is a *delinquency*, and 2) which are the valid sharing transactions that it can still perform on the object.

When a delinquency is detected, the node needs to report it and appropriate actions against the delinquent node should be taken. To manage delinquencies, we suggest the exploitation of a central trusted register manager (i.e., *TReMa*) that manages a central *delinquencies register*. The *TReMa* listens to all the nodes in the system, receives reported susceptible Bichon chains, evaluates them, and records in the *delinquencies regis-*

[1]Bichon is the name of a company dogs' breed.

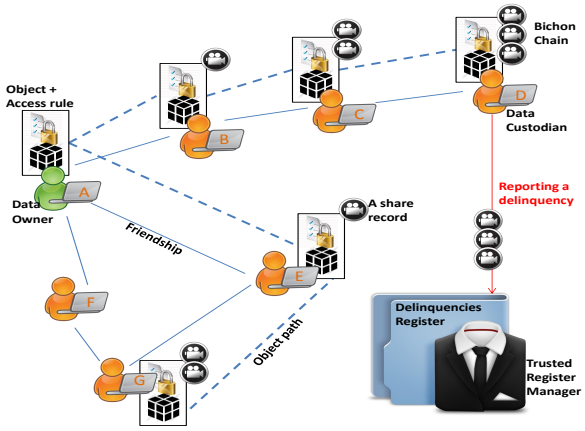

Fig. 2: The CARDS architecture for a decentralized social network with a trusted monitor

ter any delinquency they contain (Figure 2). The *delinquencies register* contains records of confirmed delinquencies as a pair *[actor, delinquency's severity]* (refer to Algorithm 2 for the severity computation). The *TReMa* also performs the activities in the *Inform& Prevent* and *Monitor& Evaluate* processes from the CARDS framework (cfr. Figure 1), to ensure the sanity of the system and to catch unreported collusion, if any. Since we target collaborative DOSNs, we limit the activities of the *TReMa* to managing the reported delinquencies and to performing periodic audits, ensuring that it does not get hold of the objects themselves. In this paper, we assume the trustworthiness of the *TReMa*, and we plan to completely remove it from the system for future works (see Section VII).

The CARDS framework is to be deployed with a reputation management system that uses the recorded delinquencies in computing entities' reputation. This system has also to provide a punish/reward mechanism that provides incentives for good actions and deters from bad ones and has to be aligned with the architecture and the requirements of the system instantiating the CARDS framework. Typically, a reputation management system should provide a mechanism for the collection of information and computation of reputation scores, a mechanism for the dissemination of these scores, and security guarantees against manipulation of scores. Reputation management is a due discipline in itself that is being subject to a number of research work [13] with proposals for both decentralized [14] and centralized solutions [15]. As such, we consider, in this paper, that a reputation management system is put in place keeping its study outside of the paper's scope.[2]

III. The CARDS Model

In this section, we detail our proposed CARDS model for DOSNs. We start by defining its basic blocks.

A. Basic Definitions

We note that we differentiate between the reference monitor of a node, represented by the software, and the end-user (the person) who manipulates it. Hereafter, we refer to the software as the *node* and to the person using it as the *end-user*. We formally define a DOSN as follows:

Definition 3.1: **DOSN**. A DOSN is a directed graph, G=(V, E, R, T), where:

- V is the set of nodes (or users), where each node $v_i \in V$ has a unique network identity (denoted $v_i.id$), a digital identity signature expressed by the ownership of an identity key-pair (idk_i, $sidk_i$), and a reputation score denoted as $v_i.rep$.
- E is the set of relationship edges such that $e^r_{ij} \in E$ denotes a relationship of type $r \in R$ from node $v_i \in V$ to node $v_j \in V$.
- R is the set of available relationship types in the DOSN.[3]
- T is a function that assigns to each edge $e \in E$ a trust value, denoted by $e.trust$, ($e.trust \in [0, 1]$).

The *TReMa* is the only entity in the system aware of all the available nodes' identity signatures[4] and it is the only entity that can verify them. The reputation of a node is public in the DOSN and can be computed based on the number and severity of delinquencies recorded against it in the *delinquencies register* and in alignment with the adopted reputation management system. Given that the *TReMa* is responsible of computing, safekeeping, and disseminating reputation scores, a centralized reputation system could be deployed. However, and for performance purposes, the dissemination of reputation scores should also be enabled at a peer to peer level such that a node could retrieve this information from its neighbors, if they have it locally, instead of contacting the *TReMa*. Solutions such as those suggested in [14][15] could be considered. We plan tackling this issue in more details in future extensions of this work.

Regarding edges' trust values, and given they will be used for evaluating access to personal data, we consider them to be assigned by the users involved in the direct relationship. Users might consider the reputation of a node when assigning their trust values to their links with it. Finally, and for computational purposes, we consider that every relationship type in R corresponds to an integer in $[1, \ldots, |R|]$.

Nodes generate content in the DOSN in the form of objects and are considered their owners. An object can be of different types[5] depending on what is supported in the DOSN. Owners set the privacy requirements for their objects and encode them into access rules. We formally define an object as follows:

Definition 3.2: **Object**. Let $v_i \in V$ be a node in the DOSN. An object owned by v_i is denoted by the tuple, Ob^i_k = $(OID, t, val, ow, AcsR)$, where:

- OID = $v_i.id\#t\#k$ is a unique ID in the DOSN referring to the object Ob^i_k, where k is a sequence number computed locally at node v_i.
- t is an integer indicating the type of the object.
- val is the bit-value of the object.

[2]We discuss this more in [16].

[3]Example of relationship types can be: *partner, colleague, family*.

[4]The identity key-pair can be implemented using any cryptographic signature scheme.

[5]Examples of object types are *text*, *picture*, *video*, etc.

- $ow = v_i.id$ is the owner of the object.
- $AcsR$ is the access rule encoding the privacy requirements of the object.

For access rules modeling, we adopt the model in [17]. By this model, every resource is linked to an access rule that expresses conditions on the type, the depth (distance between owner and requestor) and the trust level that a relationship needs to satisfy so that its members are eligible to access the resource. That is, given an object Ob, its access rule ($Ob.AcsR$)[6] is a list of access conditions among which at least one has to be met for a legitimate access to the object. An access condition is denoted by $acsc$ = $(ow, RType, MaxD, MinT)$, where ow refers to the owner of the object $RType$, $MaxD$, and $MinT$ respectively refer to the type, the maximum distance, and the minimum trust of the relationship between the owner and a legitimate receiver.

In a DOSN, no node would be able to trace paths of distance higher than 1. For that, we need to consider an indirect trust value for indirect relationships. The literature offers several ways for computing indirect trust values [18]. In our model, and since the access rules refer to the existence of specific paths, we only consider the path in question for the computation of indirect trust. That is, given the trust values, $e_{ij}.trust$ and $e_{jk}.trust$, between directly connected nodes v_i and v_j, and v_j and v_k, the trust value of the path $p_{ijk} = (e_{ij}, e_{jk})$ is expressed by: $p_{ijk}.trust = e_{ij}.trust * e_{jk}.trust$.

Objects are shared between nodes exploiting the edges established between them. To log an object's sharing activity, a *Bichon chain* is attached to the object since it first leaves its owner. A Bichon chain is a list of chained sharing records, each signed by the node performing it. A sharing record is defined as follows:

Definition 3.3: **A sharing record**. Let $v_o, v_i, v_{i+1} \in V$ be three consecutively connected nodes in the DOSN via relation types $r, r_1 \in R$ respectively, i.e., $e_o = e^r_{oi} \in E$ and $e_1 = e^{r_1}_{i(i+1)} \in E$. Let Ob^o be a resource owned by v_o. The sharing records of object Ob^o are generated as follows:

- When node v_o shares Ob^o with its direct neighbor v_i using the edge e_o, it creates the initialization tuple $ShR_0 =< Ob^o.OID, tr, dist, ty >$ s.t. $Ob^o.OID$ is the object's ID, $tr = e_o.trust$, $dist = 1$, and $ty = r$. ShR_0 is called the *reference sharing record*.
- The sharing record of Ob^o from v_i to v_{i+1} is the tuple: $ShR_1 =< Ob^o.OID, tr_1, dist_1, ty_1 >$ s.t. $tr_1 = ShR_0.tr * e_1.trust$, $dist_1 = ShR_0.dist + 1$, and $ty_1 = r_1$.

As by Definition 3.3, a sharing record is created, by the sharing node, for every passing of the object from one node to another. It contains aggregate information on the path traversed by the object so far and it is meant to publish the criteria based on which the sharing happened.

The sharing records of an object are all chained to make its sharing trajectory comprised in the *Bichon chain*. This chain is a list of blocks (i.e., *Bichon rings*) each containing a sharing record for a given object's trajectory and a corresponding

[6]We use the dot notation to denote an object components.

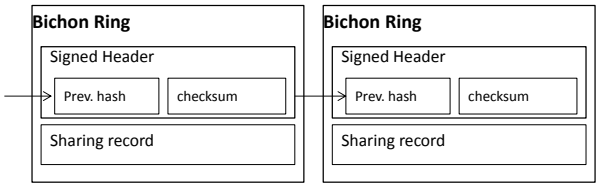

Fig. 3: The Bichon chain for an object's share trajectory

Bichon header. This header serves for the verification of both the correctness and the integrity of the chain, and for the identification of the block's creator by the *TReMa* when performing the audit. We formalize a *Bichon header* as follows:

Definition 3.4: **A Bichon header**. Let $v_o, v_i, v_{i+1} \in V$ be three consecutively connected nodes in the DOSN and let $ShR_0 =< Ob^o.ID, tr, dist, ty >$ and $ShR_1 =< Ob^o.ID, tr_1, dist_1, ty_1 >$ be the sharing records of object Ob^o, owned by node v_o, from v_o to v_i and from v_i to v_{i+1}.

- The *reference Bichon header* for ShR_0 is denoted by the signed element: $header_0 = sign_{sidk_o}(checksum_0)$; where $sign_{sidk_o}(X)$ is a function that signs message X with the identity secret-key $sidk_o$ of node v_o and $checksum_0 = tr + dist + ty$.
- The *Bichon header* for ShR_1 is: $header_1 = sign_{sidk_i}(concat(checksum_1, H(header_0)))$ where $sidk_i$ is the identity secret-key of node v_i, $concat(f, g)$ is the concatenation of strings f and g, and $H()$ is a hash function. $checksum_1 = tr_1 + dist_1 + ty_1$.

Every Bichon ring contains one sharing record plus its corresponding Bichon header. As per Definition 3.4, every header comprises a hash of the previous Bichon ring's header. This has the effect of creating a chain of sharing records making the trajectory of the traced object. Sharing records (rings) are guaranteed to be chronologically ordered in the Bichon chain because the previous Bichon ring's header would otherwise not be known. The Bichon rings can be verified for integrity using the checksum value included in the signed header, and the signed headers serve for accountability recording and for audit tracking by the *TReMa*. Figure 3 depicts a simplified sketch of a Bichon chain. Formally, a Bichon chain is defined as follows:

Definition 3.5: **Bichon chain**. The *Bichon chain* of an object Ob, denoted as *BC_Ob*, is a list of *BC_Ob.length* Bichon rings: *BC_Ob* = $\{ring_i \mid ring_i = [header_i, ShR_i], i \in [0, BC\_Ob.length]\}$, where ShR_i is a sharing record and $header_i$ is its corresponding Bichon header. $ring_0 = [header_0, ShR_0]$ is the reference ring of *BC_Ob*.

B. Sharing and Reporting

In this section, we detail the sharing and reporting processes.

1) Sharing objects: When a node receives an object, it also receives its corresponding Bichon chain. Upon viewing the object, the end-user might choose to share it with one of its friends. At such event, the node uses the object's access rule along with the last Bichon ring in the Bichon chain to evaluate the legitimacy of the intended sharing transaction. Let us first define the following:

Definition 3.6: **Legitimate sharing record**. Let Ob^o be an object and let $ShR_n =< Ob^o.ID, tr_n, dist_n, ty_n >$ be the sharing record in the n-th Bichon ring of its Bicon chain. Given the access rule of the object, encoded as a list of access conditions, $Ob^o.AcsR$ = $\{acsc \mid acsc = (Ob^o.ow, RType, MaxD, MinT)\}$, the sharing record ShR_n is legitimate if and only if: $\exists\ acsc_h = (ow, RType_h, MaxD_h, MinT_h) \in Ob^o.AcsR$, such that: $tr_n \geq MinT_h \wedge dist_n \leq MaxD_h \wedge ty_n = RType_h$.

The evaluation of the legitimacy of a sharing transaction is performed by Algorithm 1, that takes as input the access rule ($Ob.AcsR$), the last sharing record in the Bichon chain of the target object (ShR_n), and the trust and the type of the edge along which the sharing is to be evaluated ($e^r.trust$ and r). The algorithm computes the sharing record to be evaluated by making a call to the *CreateShR* function (line 1). Then, it evaluates its legitimacy by the *MatchRule* function (line 2). If the sharing record is legitimate, the algorithm returns *Legitimate* (line 4); otherwise it returns a *Delinquency* message (line 7).

Algorithm 1: Evaluation of the legitimacy of a sharing

Input : The access rule of the object to be shared: $Ob.AcsR$
Input : The last sharing record in the Bichon chain of the object to be shared: $ShR_n =< Ob.ID, tr_n, dist_n, ty_n >$
Input : The edge along which the object is to be shared: e^r
Output: Message $legitimate$ or $delinquency$.

1 *new_ShR* = `CreateShR(`$e^r.trust, r, ShR_n$`)`;
2 *match* = `MatchRule(`*new_ShR, Ob.AcsR*`)`;
3 **if** *match* **then**
4 | **return** *Legitimate*;
5 **else**
6 | **return** *Delinquency*;
7 **end**
8 **Function** `CreateShR(`$e^r.trust, r, ShR_n$`)`
9 | $SR\_tr = ShR_n.tr_n * e^r.trust$;
10 | $SR\_ty = r$;
11 | $SR\_dist = ShR_n.dist_n + 1$;
12 | $SR =< Ob.ID, SR\_tr, SR\_dist, SR\_ty >$;
13 | **return** *SR*;
14 **end**
15 **Function** `MatchRule(`*new_ShR, Ob.AcsR*`)`
16 | **foreach** *acsc* **in** *Ob.AcsR* **do**
17 | | **if** $new\_ShR.ty = acsc.RType \wedge$ $new\_ShR.tr \geq acsc.MinT \wedge$ $new\_ShR.dist \leq acsc.MaxD$ **then return** *True*;
18 | **end**
19 | **return** *False*;
20 **end**

Once the node has the answer to the legitimacy of the intended sharing, it prompts the end-user for the last decision. If the end-user chooses to share the object, regardless of the legitimacy of the transaction, the node creates the Bichon ring corresponding to this sharing and appends it to the object's Bichon chain. Afterwards, the node initiates a secure communication channel[7] with the target node (i.e., the next receiver of the object). There are two possible scenarios, that are exemplified by Example 1: 1) the sharing transaction is legitimate, or 2) the sharing transaction is a delinquency.

[7]We consider any communication between nodes to be over a secure channel whose establishment is not covered in the described flows.

Example 1. Let us assume Kate and Jane are DOSN friends and that the relationship edge from Kate to Jane has type *colleague* and trust value 1. Assume that Kate receives a video (denoted as V) for which an access condition allows sharing it with *colleague*, with minimum trust= 0, and maximum distance= 10. Assume that the value of the distance parameter in the last ring of the video's Bichon chain is 5 and that Kate wants to send the video to Jane. Figure 4(a) depicts the communication flow between Kate's and Jane's nodes to perform this sharing. First, the Bichon chain of the video, *BC_V*, is updated at Kate's node to add the ring for the sharing transaction to be performed and it is then encapsulated, with the video's access rule ($V.AcsR$), in a *newShare()* request to Jane's node (see Figure 4(a)). The received Bichon chain is verified at Jane's node by extracting the last ring's sharing record from the Bichon chain and evaluating it using function *MatchRule* from Algorithm 1. In this case, the sharing is legitimate, so V is accepted and the transaction is fulfilled.

Assume now that Kate received a photo (denoted as P) owned by her cousin Ryan, with whom she is friend on the DOSN with a *family* relationship type. Ryan wants the photo to be viewed by family members only. Kate decides to share the photo with Jane though this sharing is a delinquency (Kate does not have a *family* relationship with Jane). Figure 4(b) presents the communication flow between Kate's and Jane's nodes for this delinquent sharing. This starts by Kate's node sending a *newShare()* request to Jane's node encapsulating the picture's Bichon chain (*BC_P*) and its access rule ($P.AcsR$). The last ring in the chain is verified at Jane's node and is found to be delinquent. Jane's node prompts Jane to take the decision. If Jane accepts, the picture is sent and the operation is terminated. If she rejects, the operation terminates without transferring the picture.

2) Reporting a detected delinquency: When a node receives a request for a delinquent sharing, it can report it to the *TReMa*, regardless of whether it accepted receiving the object or not. In fact, even in the event of accepting a delinquent sharing, the receiver might want to redeem for having colluded in the delinquency by reporting it. This would be an illustration of a honest-but-curious user. To report the detected delinquency, the node needs to send to the *TReMa* the Bichon chain in question along with the concerned object's access rule. To demonstrate a reporting operation, we continue with the scenario in Example 1. In fact, when Jane's node finds that the sharing request for object P is a delinquency, it prompts Jane for whether it has to report it to the *TReMa* or not. If Jane chooses to report it, a *report delinquency message - rdm* is sent to the *TReMa*. A report delinquency message is defined as follows:

Definition 3.7: **Report delinquency message (*rdm*)**. Let $v_i \in V$ be a node in the DOSN, Ob be an object, and *BC_Ob* its Bichon chain. An *rdm* on *BC_Ob* communicated by node v_i to the *TReMA* is defined as follows:

rdm = $sign_{sidk_i}(Ob.AcsR, BC\_Ob)$; where $sidk_i$ is the identity secret-key of node v_i.

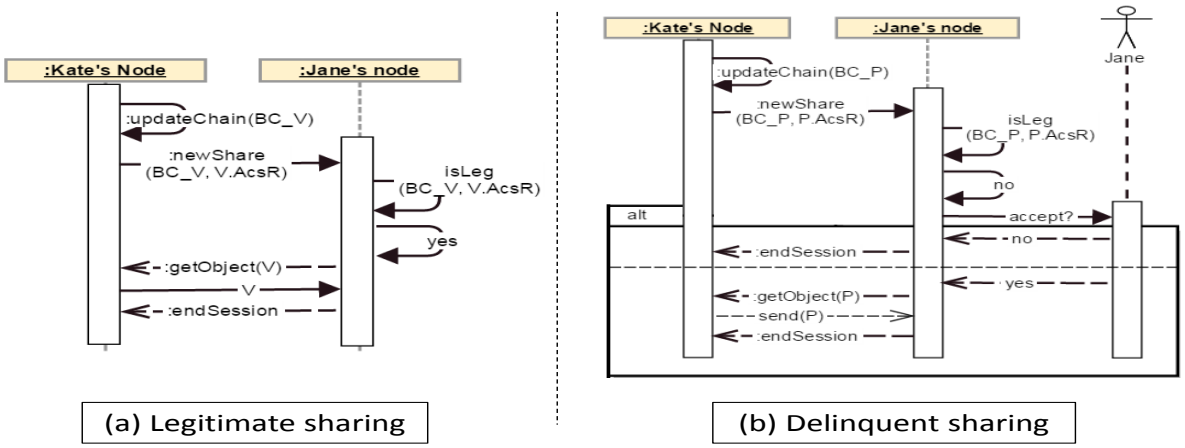

Fig. 4: Communication flow for a legitimate and a delinquent sharing

C. The TReMa's Operations

The *TReMa* reacts to reported delinquencies by executing the processes *Listen& React* and *Inform& Prevent* (cfr. Figure 1). It verifies the reported message, records the confirmed delinquencies, informs the faulty nodes, and runs targeted auditing on other nodes that could have possibly received the victim object. Before detailing the *TReMa*'s operations, we first define a delinquency record.

Definition 3.8: **Delinquency record (*dr*)**. A delinquency record (*dr*) is a pair [*actorID, severity*], where *actorID* = $v.id$ ($v \in V$) and *severity* $\in \mathbb{N}^*$.

1) Listen and React: The *TReMa* evaluates a received rdm from node v_i by running Algorithm 2. The algorithm takes as input the message rdm and the identity-key idk_i of the reporting node v_i. It initializes an empty *log* list (its output parameter) and checks the validity of rdm by verifying its signature (by calling function *CheckSignature* in line 1). If the message is valid, function *CheckChainCorrectness* is called (line 3) to verify the integrity of the reported chain by checking the conformity of each ring's header to its corresponding sharing record (via the checksum value) and to the hash of the previous ring's header, as from Definition 3.4 (lines 13-22). If the verification of the chain or of the rdm's signature fails, the algorithm returns a null value. Otherwise, a call to the *AuditChain* function is made (line 5). This latter evaluates the legitimacy of the rings of the target chain (i.e., *BC_Ob*) starting from the last one (lines 24-36). It then continues evaluating the previous rings as long as delinquencies are detected (lines 26-32). To evaluate a ring, its enclosed sharing record is extracted (line 27) and it is checked for legitimacy using function *MatchRule* from Algorithm 1 (line 28). If the sharing record is not legitimate, the actor of that sharing transaction is extracted from the corresponding ring's header using function *GetSignerID* (line 30) and a delinquency record is added to the log with a severity level equal to 1, by using function *RecordDelinq* (line 31). When the algorithm finds a legitimate ring in the chain, it stops its evaluation as rings previous to it are also legitimate (the break statement at line 33). If the number of delinquent rings is more than 1 (line 35), the severity levels of the logged delinquency records need to be adjusted. This is what is achieved by invoking function *AdjustSeverityLevels* at line 35. This function increases the severity level of each delinquency record based on its position in the series of delinquent rings (lines 42-46). That is, the actor of the first delinquent sharing is punished with a severity equal to 1, the second colludent with a severity of 2, and so on. This adjustment ensures that nodes that collude by receiving a delinquent sharing without reporting it, get a higher punishment than the first initiator of the collusion. Finally, the algorithm returns the list of found delinquency records.

If the reported delinquency is a false alarm (i.e., the output Log is empty), the *TReMa* notifies the reporter node v_i and logs the false alarm. Moreover, a log of all received notifications with the result of their evaluation is kept by the *TReMa*. This information can be used to reward the nodes reporting valid delinquencies by increasing their reputation.

2) Inform and Prevent: After recording a delinquency, the *TReMa* informs the object's owner about the incident. The owner can use this information to reconsider the assigmnent of access rules to its future shared objects. Besides, it also informs all the nodes against whom a delinquency record has been logged, as returned by Algorithm 2. Afterwards, the *TReMa* takes a protective procedure to prevent eventual delinquencies to happen with the same object by starting a targeted auditing on all these nodes's direct contacts (next potential colludents). It then continues auditing on any other of their contacts' contacts found to have received the object until the sharing of the object is found to be stopped.

Algorithm 3 presents the steps to perform the targeted auditing based on an identified object. The algorithm takes as input the log of delinquency records found for an object (i.e., the output of Algorithm 2) and the id of the object in question ($Ob.OID$). Using these two parameters, it goes through all the delinquency records, *dr* in the input list. For each *dr*, it retrieves all the friends of its actor node (line 3). For each of these friends, it checks whether they have received the target object from the actor node or not (line 5). If a friend is found to have received the object from the actor node,[8] a delinquency record is created against it using function *RecordDelinq* from Algorithm 2 by passing to it a severity input parameter incremented by 1 compared to the severity by which the previous *dr* has been recorded (line 6).

[8]A friend might have received the same object but from another node via a legitimate sharing.

Algorithm 2: Processing an rdm by the *TReMa*

Input : reported delinquency message from node v_i: $rdm = sign_{sidk_i}(Ob.AcsR, BC\_Ob)$.
Input : identity-key of node v_i: idk_i.
Output: A list of delinquency records for delinquencies in the chain: Log.

```
1  Log = ∅; validMsg = CheckSignature(rdm, idk_i);
2  if validMsg then
3      validChain = CheckChainCorrectness(BC_Ob);
4      if validChain then
5          Log = AuditChain(BC_Ob, Ob.AcsR);
6          return Log;
7      else
8          return null;
9      end
10 else
11     return null;
12 end
13 Function CheckChainCorrectness(BC_Ob)
14     foreach ring_i in BC_Ob do
15         header_i = UnsignMsg(ring_i.header_i);
16         if is ring_0 then
17             if header_0.checksum_0 is not equal
               ShR_0.tr + ShR_0.dist + ShR_0.ty then return false;
18         else
19             if header_i is not equal
               concatenate(header_i.checksum_i, H(header_{i-1}))
               then return false;
20         end
21     end
22     return true;
23 end
24 Function AuditChain(BC_Ob, Ob.AcsR)
25     Log = ∅; pos = 0; len = BC_Ob.length - 1;
26     for ring_len in BC_Ob do
27         ShR = ShR_len;
28         legit = MatchRule(ShR, Ob.AcsR);
29         if not legit then
30         actor = GetSignerID(header_len);
31         RecordDelinq(Log, pos, actor, 1);
32         pos = pos + 1; len = len - 1;
33         else break;
34     end
35     if pos ≥ 2 then Log = AdjustSeverityLevels(Log, pos) ;
36     return Log;
37 end
38 Function RecordDelinq(Log, n, actor, severity)
39     dr = [actor, severity]; Log.Add(dr, n);
40     return Log;
41 end
42 Function AdjustSeverityLevels(Log, pos)
43     foreach dr_i in Log (i starts from 0) do
44         dr_i.severity = pos; pos = pos - 1;
45     end
46     return Log;
47 end
```

This new recorded delinquency is appended to the initial input *colludents* list so that the friends of its actor are audited as well (line 10). Finally, the algorithm returns the log of all new found delinquency records.

The aim of the targeted auditing on an identified object is to track all other possible colludents after the first delinquency report received on it. It is admitted that the prevention is not strong as it does not guarantee full protection of the victimized object. However, it works on minimizing the risk of it being subject to further mis-shares by notifying possible future colludents. Given that the system is based on a reputation spirit, we believe that warning nodes and/or punishing them through detecting and recording their delinquencies would have a deterrence preventive outcome.

Algorithm 3: Targeted auditing based on identified objects

Input : list of delinquency records of colluding nodes on mis-sharing an object: $colludents = \{dr_1, .., dr_n\}$.
Input : the target object id: $Ob.OID$.
Output: A list of delinquency records for delinquencies detected on the object: Log.

```
1  Log = ∅; pos = 0;
2  foreach dr in colludents do
3      targets = GetFriendsOf(dr.actor);
4      foreach friend in targets do
5          if HasReceivedObject(Ob.OID) then
6              RecordDelinq(Log, pos, friend, dr.severity+1);
7              pos = pos + 1;
8          end
9      end
10     colludents = colludents ∪ Log;
11 end
12 return Log;
```

3) Monitor and Evaluate: Besides reacting to reported delinquencies, the *TReMa* takes proactive actions by executing the activities under the *Monitor& Evaluate* process (cfr. Figure 1). Periodically, the *TReMa* starts general auditing on three different groups of nodes, of fixed sizes, selected randomly and uniformly from their respective pools. The first group is selected from the pool of nodes that have no delinquency record registered yet. The second one is selected from nodes having at least one record in the delinquencies register. As for the third group, it is selected from the nodes for which the number, the severity level, and the frequency of recorded delinquencies is the highest since the last performed periodic audit. The rationale behind running periodic auditing on these three groups is to ensure the cleanness of the DOSN assuming that not all the nodes will abide by the reporting mechanism. For the first group, the aim is to double check that there are no malicious nodes hidden uncaught among good ones; whereas targeting the second group aims at checking how suspicious nodes behaved since the last recorded delinquency against them. As for the third group, the aim is to insist on deterring them considering that they do not seem to be compliant with the system yet.

A general periodic audit is performed by requesting all the Bichon chains in custody of the target nodes. Each of these chains is then verified, as by function *AuditChain* from Algorithm 2, to check and record any delinquencies it contains. We rely on the underlying software to ensure the communication of all the Bichon chains in a nodes' custody. However, if the node is broken through and the *TReMa* cannot get hold of its Bichon chains, it immediately informs all the network about the invalid node.

IV. Security and Complexity Analyses

In this section, we provide the security properties and we discuss the complexity of the system. Proofs and detailed discussions for the provided results are available at [16].

A. Security Properties

We consider a malicious adversary model with three possible attacks. First, malicious nodes can collude to propagate a delinquent sharing and never report it to the *TReMa*. Second, a malicious node can create fake valid Bichon chains to compromise the reputation of other honest nodes. Finally, a malicious node can alter valid Bichon chains either by inserting new fake rings or by changing the information already contained in the chains.

Let $v_m \in V$ be a malicious node with the aim of propagating an illegitimate sharing on an object Ob. Let S be the number of nodes randomly selected from V for the *TReMa*'s periodic general audit. CARDS satisfies three security properties:

- **P1:** v_m cannot remain uncaught given T general audit cycles, with T following a geometric distribution of $p = \frac{S}{|V|}$.
- **P2:** Fake but valid Bichon chains cannot be created.
- **P3:** Compromising the integrity of a valid Bichon chain will result in an invalid chain.

B. Complexity Analysis

For Algorithm 1, its complexity is $\mathcal{O}(\textit{Rule-size})$, with *Rule-Size* = $|Ob.LAcsR|$ being the number of the conditions in the longest object access rule in the system. Given that access rules are set by users to express their privacy requirements on an object, the length of a rule is logically not expected to be big.

For Algorithm 2, its complexity is $\mathcal{O}(|V| * LBC\_Ob.length)$, where *LBC_Ob.length* is the length of the longest Bichon chain in the system, and $|V|$ is the total number of nodes in the DOSN. *LBC_Ob.length* is determined by the maximum hops an object has to go through before it reaches all the nodes. Given the principle of six-degrees of separation, which has been demonstrated by experimental results to exist in today's online social networks with even a shorter scale (i.e., four-degrees) [19], *LBC_Ob.length* can be estimated to be less than 6. Concerning the $|V|$ parameter, it comes from the fact that the algorithm performs a search for users' identity keys, to validate rings, among all available keys. This might be considered costly, but can also be easily addressed by adopting a distributed hierarchical paradigm in designing the *TReMa*, similar to the architecture of the DNS[9] protocol, for example. We further discuss this under the following section where we also present results on the scalability and performance of Algorithm 2.

Finally, Algorithm 3's complexity is $\mathcal{O}(\textit{LBC\_Ob.length} \times |C|)$, where $|C|$ is the number of detected colludents by the targeted audit.

V. Performance Results

As the *TReMa*'s operations might seem quite demanding (analysis of Algorithms 2 and 3), we perform experiments simulating these operations to study the scalability of these two algorithms.

We run our experiments using the OSN's Pokec[10] public dataset[11] that comprises 1.63 M nodes and 30.6 M edges. Its diameter, longest shortest path, is equal to 11, and so we set the maximum length of a Bichon chain to 11. We implemented both Algorithms 2 and 3 using the Java language. All the experiments have been conducted on a dual core PC of 3.4 GHZ each and 4 MB of RAM.

1) Scalability of Algorithm 2: Algorithm 2 processes a reported Bichon chain against the access rule of its object. We have simulated this reporting scenario by creating 10 different objects (i.e., 10 different access rules) and 10 Bichon chains, each of length 11, per object. For a given object, each of the 10 Bichon chains contains from 0 (false alarm) to 10 (max possible delinquencies in the chain) delinquent rings. We run Algorithm 2 20 times for each of the simulated objects against its 10 simulated Bichon chains. Since the algorithm performs a linear search for a ring's signer ID, in the case of a delinquent ring, we have set the pool of IDs over which this search is performed to 1K nodes, 10K nodes, and 20K nodes for three independent executions using the same settings w.r.t Bichon chains and objects. The sizes of the pools of IDs have been set based on the assumption of having a distributed *TReMa* architecture by blocks of communities. In fact, based on analyses of famous OSN datasets in the literature, we find that the average community size is in the order of hundreds of nodes [20]. As such, we run our simulations under the assumption of one *TReMa* node for every block of ten, hundred, and two hundred communities, respectively.

Figure 5 presents the results of this experiment. The x-axis refers to the number of found delinquencies in the reported chain, whereas the y-axis refers to the average run time in seconds of the algorithm. The figure depicts three lines each corresponding to one of the set pool sizes for the search on signer IDs. The first thing we read on the figure is that the higher the size of the pool is, the higher the run time is. This goes in line with the complexity of the algorithm (i.e., $\mathcal{O}(|V| * BC\_Ob.length)$). $|V|$ here represents the pool size. The second thing we can notice is that the higher the number of delinquencies is, the higher is the run time (the lines follow a linear sloped function). This is also expected as the search operation is performed per every delinquency found. However, the run time with the number of delinquencies under 5 (50% of the chain) is very low (35s as the highest recorded point for a pool of 20K nodes). Finally, even under the worst case of all the chains being delinquent (which is also not realistic) and with a pool size of 20K nodes, the algorithm converges in less than 90 seconds on a standard PC.

2) Number of audited nodes and scalability of Algorithm 3: Algorithm 3 takes a list of delinquency records (*drs*) found w.r.t a target object and performs an audit on the friends of their actors and subsequently on their friends as long as

[9]Domain Name System [https://www.netbsd.org/docs/guide/en/chap-dns.html]

[10]A Slovakian OSN: https://pokec.azet.sk/.

[11]http://snap.stanford.edu/data/soc-pokec.html

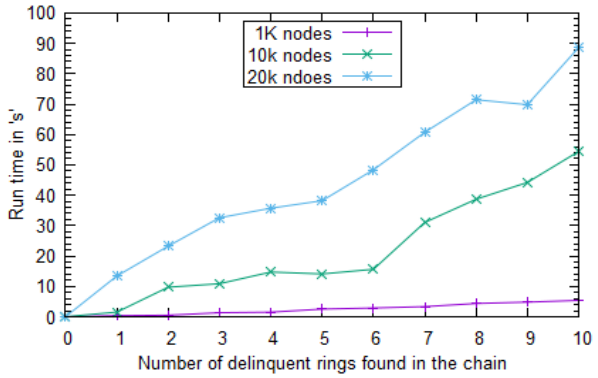

Fig. 5: Run time in seconds of Algorithm 2

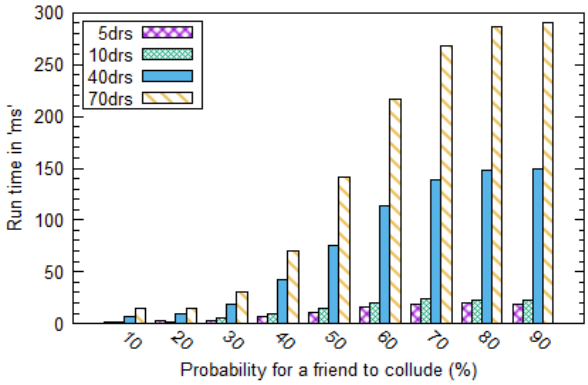

Fig. 6: Run time in milliseconds of Algorithm 3

they are found to have received the object in question. The length of the input *drs* list is expected to be the same as the number of delinquent rings found in the reported chain that fired Algorithm 3. For experimental purposes, we have considered 4 scenarios with this length being of 5, 10, 40, and 70 drs, respectively. 5 drs corresponds to 50% of the reported chain being delinquent, and 10 drs to 100% of it. The other 40 drs and 70 drs have been set to stretch the system and study it's performance under more exhaustive conditions. Since the algorithm performs the targeted audit on a nodes' contacts as long as they are found to have received the object, we have simulated this by considering a varying probability for this to happen. That is, when a node is audited, a biased coin is flipped first. Depending on the flip result, the node is considered as has received the object or not. The bias in the coin is represented by a probability for the node to have received the object. We vary this probability from 10% to 90%. We simulate the algorithm for 20 times and report the average run time results in Figure 6.

From Figure 6, we can notice that the run time of the algorithm is in terms of milliseconds only and that the worst recorded time of 291ms is achieved with 70 initial drs (value used just to stretch the simulation) and a probability for friends to collude of 90%. We can also read on the same figure that the run time of the algorithm remains under 20ms for 5 drs and 10 drs with the probability of a friend to collude being lower than 60%. We think that this is a very good result as the worst case scenario for Algorithm 3 is when all the reported chain

	5 drs	**10 drs**	**40 drs**	**70 drs**
1 hop($90 \geq PCol \geq 60$)	79	192	1360	1441
2 hops($60 \geq PCol \geq 20$)	143	326	2316	2595
3 hops($20 \geq PCol \geq 10$)	7	18	82	134
4 hops($PCol \leq 60$)	0	0	1	3

TABLE I: Number of audited nodes by Algorithm 3

is delinquent (i.e., 10 drs) and a non majority of colludents (i.e., probability to collude is less than 50%).

In addition to studying the performance of Algorithm 3 in terms of its running time, we also computed the number of audited nodes. Since the algorithm keeps auditing all the contacts of a node as long as they have received the object, we run our simulation by considering a probability of collision (*PCol*) of $90 \geq PCol \geq 60$ for the first hop friends, $60 \geq PCol \geq 20$ for the second hop, $20 \geq PCol \geq 10$ for the third hop, and $PCol \leq 60$ for the fourth hop. Table I presents the results on the number of audited friends by hop and by considering the same number of drs as before (5drs, 10drs, 40 drs, and 70drs). As we can read on the table, the number of audited nodes converges to 0 by the fourth hop. Besides, this number scores 326 audited nodes as the highest value with 10 initial drs. Recalling that the maximum initial drs the algorithm can have as input is 10 (the other values used to stretch the system), our experiment shows that the worst case scenario results in 326 audited nodes only.

VI. Related Work

AC in OSNs is mainly deployed following the relationship-based model, which, in DOSNs, is often enforced using cryptographic means [9], [10]. Shared objects are encrypted by their owners to be read only by the subjects to whom they have preliminary granted the needed tools to decipher them. This adds a highly undesirable overhead to the system due to the number of secure keys to be managed and to the cost of objects' encryption [11]. Although several research work have addressed this performance issue, by adopting differing levels of data encryption [10], [21], almost all of the existing solutions for DOSNs still do not demonstrate the flexibility and the usability levels available in centralized OSNs. For this, we believe that an a-posteriori and managerial based approach to protecting users' data in DOSNs might be the solution to providing both flexible and safe p2p socializing environments. Indeed, in our proposal, DOSN users, who initially collaborate to create the underlying socializing platform, can also collaboratively manage the insurance of secure data sharing. This approach relies, like it is for almost all p2p systems, on the spirit of honest majorities willing to collaborate to create and benefit from the intended services.

Another body of work that might be thought of as crossing with our proposal is related to the design of access policies that continuously accompany their corresponding data to ensure their privacy requirements across the different services and platforms they travel through. Known as sticky policies, the research on this field concerns the problem of ensuring the enforcement of the privacy requirements of data owners even when the data goes to third parties, mainly by having the

access policies stick to their corresponding data [22]. Whilst the access rules attached to objects in our proposal might be seen as similar to this concept of sticky policies, our work does not concern the pre-enforcement of these policies, their formulation, or their transporatbility across different services as it is in the research on that field. Indeed, in our suggested model access rules are attached to objects for access legitimacy evaluation only, whilst the enforcement is ensured in a post-sharing manner by relying on the Bichon chains accompanying objects and recording sharing operations that they have been subject to.

Finally, there are proposals for collaborative access control, such as the work in [23], that introduced some levels of transparency by showing when an object is shared against its original access policies. However, that work does not tackle the problem from an a-posteriori approach and is not based on auditing with reward/punish mechanisms, as it is the case for our proposal. In general and to the best of our knowledge, this is the first proposal for collaborative a-posteriori and audit based access control for DOSNs.

VII. Conclusion and Future Work

In this paper, we proposed a framework for collaborative audit based data sharing in DOSNs. Our approach allows for data accountability, as opposed to the traditional pre-enforced access control models. We believe that a-posteriori access control might be the future to personal data security management in DOSN realms. In fact, besides ensuring accountability, we also believe that a-posteriori control would help enhance users' awareness regarding the management of their privacy. This would happen by closing the audit loop and providing feedback to users regarding the destiny of their shared objects.

We admit that the suggested base model should be complemented by proper privacy-preserving techniques. Indeed, the data in an object's Bichon chain is communicated between nodes in plain format and so any node can learn from it the details of the path the object has traversed. Being aware of this, we are working on a privacy-preserving version of our model.

In addition to this, we also plan to study the usability of the system and the effect of the suggested a-posteriori AC on privacy awareness. Moreover, we plan to redesign the system to be fully decentralized without having the *TReMa*, by harvesting on technologies similar to the alternative chains provided within the Bitcoin protocol,[12] for example. Finally, we also consider designing an appropriate accompanying reputation/trust management system.

Acknowledgment

This work is partially supported by the iSocial EU Marie Curie ITN project (FP7-PEOPLE-2012-ITN).

[12] $https://en.bitcoin.it/wiki/Alternative\_chain$

References

[1] E. Ferrari, *Access Control in Data Management Systems*, ser. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2010.

[2] D. J. Weitzner, H. Abelson, T. Berners-Lee, J. Feigenbaum, J. Hendler, and G. J. Sussman, "Information accountability," *Communications of the ACM*, vol. 51, no. 6, pp. 82–87, 2008.

[3] C. Saran, "Tim Berners-Lee: Data sharing needs accountability," 2014. [Online]. Available: http://www.computerweekly.com/news/2240232292/Tim-Berners-Lee-Data-sharing-needs-accountability

[4] M. L.Pava, "Encyclopedia Britannica-Auditing, Accounting," 2013. [Online]. Available: http://www.britannica.com/EBchecked/topic/42575/auditing

[5] M. Dekker and S. Etalle, "Audit-based access control for electronic health records," *Electronic Notes in Theoretical Computer Science*, vol. 168, pp. 221–236, 2007.

[6] K. Padayachee and J. H. Eloff, "Adapting usage control as a deterrent to address the inadequacies of access controls," *computers & security*, vol. 28, no. 7, pp. 536–544, 2009.

[7] S. Etalle and W. H. Winsborough, "A posteriori compliance control," in *Proceedings of the 12th ACM symposium on Access control models and technologies*. ACM, 2007, pp. 11–20.

[8] M. Kelly, "Paying for privacy: Why it's time for us to become customers again," 2013. [Online]. Available: http://venturebeat.com/2013/09/06/paying-privacy/

[9] L. A. Cutillo, R. Molva, and T. Strufe, "Safebook: A privacy-preserving online social network leveraging on real-life trust," *Communications Magazine, IEEE*, vol. 47, no. 12, pp. 94–101, 2009.

[10] O. Bodriagov, G. Kreitz, and S. Buchegger, "Access control in decentralized online social networks: Applying a policy-hiding cryptographic scheme and evaluating its performance," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on*. IEEE, 2014, pp. 622–628.

[11] S. Buchegger and A. Datta, "A case for p2p infrastructure for social networks-opportunities & challenges," in *Wireless On-Demand Network Systems and Services, 2009. WONS 2009. Sixth International Conference on*. IEEE, 2009, pp. 161–168.

[12] H. F. Tipton and M. Krause, *Information security management handbook*. CRC Press, 2003.

[13] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decision support systems*, vol. 43, no. 2, pp. 618–644, 2007.

[14] P. Dewan and P. Dasgupta, "P2p reputation management using distributed identities and decentralized recommendation chains," *Knowledge and Data Engineering, IEEE Transactions on*, 2010.

[15] T. Ala-Kleemola and S. Tolvanen, "Reputation management system," 2012, uS Patent 8,112,515.

[16] L. Bahri, B. Carminati, and E. Ferrari, "Cards - collaborative audit and report based data sharing in dosns (extended version of this paper)," http://strict.dista.uninsubria.it/?page_id=765.

[17] B. Carminati, E. Ferrari, and A. Perego, "Enforcing access control in web-based social networks," *ACM Transactions on Information and System Security (TISSEC)*, vol. 13, no. 1, p. 6, 2009.

[18] G. Liu, Y. Wang, and M. A. Orgun, "Trust transitivity in complex social networks." in *AAAI*, vol. 11, 2011, pp. 1222–1229.

[19] E. Y. Daraghmi and S.-M. Yuan, "We are so close, less than 4 degrees separating you and me!" *Computers in Human Behavior*, vol. 30, pp. 273–285, 2014.

[20] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowledge and Information Systems*, vol. 42, no. 1, pp. 181–213, 2015.

[21] S. Jahid, S. Nilizadeh, P. Mittal, N. Borisov, and A. Kapadia, "Decent: A decentralized architecture for enforcing privacy in online social networks," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*. IEEE, 2012, pp. 326–332.

[22] S. Pearson and M. C. Mont, "Sticky policies: an approach for managing privacy across multiple parties," *Computer*, 2011.

[23] S. Damen, J. den Hartog, and N. Zannone, "Collac: Collaborative access control," in *Collaboration Technologies and Systems (CTS), 2014 International Conference on*. IEEE, 2014.