Evidence-Based Trust Mechanism Using Clustering Algorithms for Distributed Storage Systems

Giulia Traverso*, Carlos Garcia Cordero*, Mehrdad Nojoumian†,
Reza Azarderakhsh†, Denise Demirel*, Sheikh Mahbub Habib*, Johannes Buchmann*
* Technical University of Darmstadt (Germany)
† Florida Atlantic University (USA)

***Abstract*—In distributed storage systems, documents are shared among multiple Cloud providers and stored within their respective storage servers. In social secret sharing-based distributed storage systems, shares of the documents are allocated according to the trustworthiness of the storage servers. This paper proposes a trust mechanism using machine learning techniques to compute evidence-based trust values. Our mechanism mitigates the effect of colluding storage servers. More precisely, it becomes possible to detect unreliable evidence and establish countermeasures in order to discourage the collusion of storage servers. Furthermore, this trust mechanism is applied to the social secret sharing protocol AS^3, showing that this new evidence-based trust mechanism enhances the protection of the stored documents.**

Keywords: Trust management, social secret sharing, applied cryptography, distributed storage systems, cloud computing, and clustering algorithms.

I. Introduction

Nowadays, it has become common practice to rely on Cloud infrastructures for the storage of data. In fact, for large amount of data or shared data, it is more convenient for a user to outsource these data to the Cloud. In distributed storage systems, documents are shared among multiple Cloud providers and stored within their respective storage servers. Cloud providers have to ensure the protection of the stored documents with respect to two aspects: confidentiality and retrievability. For example, assume health records are stored and managed. They contain sensitive information whose disclosure harm patients' reputation (e.g. sexual or genetic diseases). Furthermore, fast retrieval is of vital importance in case of an emergency (e.g. blood group for urgent transfusions). Ensuring confidentiality and retrievability is inherent to the trustworthiness of the storage servers involved. In fact, untrustworthy storage servers are a risk for confidentiality in case they leak information to third parties. Moreover, they might not respond, respond late, or even lose the data.

A. Problem Description

Distributed storage systems are able to allocate data according to the trustworthiness of the storage servers when they are based on so called *social secret sharing* [17], [16]. Social secret sharing is a primitive that can be used in the Cloud-based systems in order to distribute more informative shares of a document to more trustworthy storage servers [18]. In fact, social secret sharing relies on trust management systems [15] to periodically update the trust value assigned to each storage server.

More precisely, every time the storage servers interacted, they evaluate each other by submitting either positive or negative evidence, depending on how well they behaved. This evidence is collected and processed to output the new trust values. Then, the system is reset and shares are redistributed according to the updated trust values. In this setting, confidentiality and retrievability of the data are provided because untrustworthy storage servers do not have enough shares to get any information about the data nor to cause its loss in case they do not respond. Note that having more shares to store and manage leads to a greater income for the Cloud. Thus, Cloud providers may enforce their storage servers to submit wrong evidence. That is, positive evidence are submitted for storage servers from the same Cloud provider and negative evidence are submitted for storage servers owned by other Cloud providers. As a result, the trust values might not communicate the actual trustworthiness of the storage servers, and consequently, more informative shares are granted to the wrong storage servers putting confidentiality and retrievability of the data at risk.

B. Contribution

Our evidence-based trust mechanism aims at enhancing the protection capability of social secret sharing-based distributed storage systems. More precisely, the *credibility* of the trust value associated with each storage server is evaluated through clustering techniques by taking into account not only the evidence submitted by the other storage servers, but also the *reputation* of those storage servers (also named evaluators). Thus, the computed trust values are built out of two measurements. The first measurement is the trustworthiness of a storage server according to all the other storage servers. It relies on so called unsupervised machine learning techniques, which have the advantage of not requiring the manual specification of the type of evidence observed. In particular, we use clustering algorithms and mixture models. The second measurement is the reliability of the storage server with respect to its submitted evidence. It is computed taking into account the reputation of the storage server and the accuracy of its submissions. Furthermore, we apply this trust mechanism to the social secret sharing protocol AS^3 [25] and show how the protection of the data stored is enhanced.

The rest of the paper is organized as follows. Section II discusses the related works. Section III provides some preliminaries on distributed storage servers and on the machine learning techniques we use. This is followed by Section IV, where our new evidence-based trust mechanism is introduced. In Section V, this new trust mechanism is applied to the social secret sharing protocol AS^3. Finally, Section VI discusses concluding remarks and future work.

II. Related Work

Evidence-based trust mechanisms rely on both direct and indirect evidence derived from past interactions. We are interested here in computational evidence-based trust mechanisms where past evidence is taken into account to estimate the trustworthiness of trustees in the future. In particular, the trust mechanism we propose builds on Bayesian probabilities-based models ([5], [2], [14], [4], [20]) and machine learning-based models ([8], [9], [12], [22], [1], [7]) and aims at detecting dynamic behaviors of the target trustee in different interactions. A complete overview of the related work can be found in the extended version of this paper [26].

III. Preliminaries

A. Distributed Storage Systems

In distributed storage systems [10] ,[21], the to be stored document is split into shares, which are distributed to several Cloud providers. Each share is stored within the storage servers belonging to Cloud providers. The shares are generated such that only a certain amount of them is needed to reconstruct the document. Cloud providers have to guarantee the confidentiality and the retrievability of documents at any point in time and, thus, have to cope with possibly untrustworthy storage servers. In fact, confidentiality and retrievability are inherent to the trustworthiness of the storage servers involved. More precisely, untrustworthy storage servers with respect to confidentiality are referred to as *honest but curious* and untrustworthy storage servers with respect to retrievability are referred to as *faulty*. These might not respond or respond with a significant delay, preventing the retrieval of the document. A solution to overcome this problem is to generate shares with different reconstruction capabilities. That is, more informative shares are distributed to the more trustworthy storage servers and vice versa. In this way, untrustworthy storage servers do not have enough reconstruction power to retrieve the document by themselves and have to collaborate with other storage servers. *Social secret sharing* [18] is a cryptographic primitive enabling exactly this type of generation and distribution of the shares. Each storage server is associated with a trust value and trust function is periodically called to update these trust values, as the trustworthiness of the storage servers may improve or worsen over time. Afterwards, new shares are distributed according to the updated trust values. More details about the redistribution procedure can be found in [24].

B. K-Means Clustering and Mixture of Gaussians

We first review the machine learning technique of K-means clustering. Note that for the definition of this technique, as well as for the definition of the technique of mixture of Gaussians in the next section, we refer to [1].

Let us define a data set $\mathcal{P} = \{P_1, \dots, P_n\}$ of n points in a D-dimensional Euclidean space, with $P_i = (x_{1_i}, \dots, x_{D_i})$, for $i = 1, \dots, n$. The *k-means clustering* is the problem of grouping these points into K clusters $\mathcal{C}_1, \dots, \mathcal{C}_K$. These clusters are identified such that the distances of points within the same cluster are smaller than the distances to points outside the cluster. This means that together with the clusters $\mathcal{C}_1, \dots, \mathcal{C}_K$, so called *center points* $M_1, \dots, M_K$ are identified, where $M_j = (x_{1_j}, \dots, x_{D_j})$, for $j = 1, \dots, K$. Each center point M_j satisfies the property that the sum of the squares of the distances of each data point P_i to the closest point M_j is a minimum. This concept can be formalized by the so called *distortion measure* J, an objective function defined as follows:

$$J = \sum_{i=1}^{n} \sum_{j=1}^{K} r_{i,j} ||P_i - M_j||^2,$$

where $||P_i - M_j||$ is the distance between points P_i, M_j and where $r_{i,j} = 1$ if point P_i is assigned to cluster $\mathcal{C}_j$ and $r_{i,l} = 0$ for $l \neq j$. Thus, the K-means clustering problem consists of finding values $r_{i,j}$ and centers M_j, for $i = 1, \dots, n$ and $j = 1, \dots, K$, such that the distortion measure J is minimized. This is achieved by means of so called EM algorithms, consisting of an expectation step E, where the values $r_{i,j}$ are adjusted, and a maximization step M, where, instead, the points M_j are adjusted. In fact, the distortion measure J can be minimized through multiple iterations, where after each iteration an expectation step and a maximization step are performed. A concrete instantiation of the above strategy can be found in [1] and [11]. Details about how to make this K-means clustering more efficient can be found in [19] and [13].

In the following, we review the machine learning technique of *mixture of Gaussians*. This technique is meant to model real data set $\mathcal{P} = \{P_1, \dots, P_n\}$ of points, which otherwise could not be fully described by a single Gaussian distribution. Consider K Gaussian distributions $\mathcal{N}_1(\mu_1, \sigma_1^2), \dots, \mathcal{N}_K(\mu_K, \sigma_K^2)$, where μ_i is the mean and σ_i^2 is the variance, for $i = 1, \dots, K$. Denoted by $p(\mathcal{P})$, a mixture of Gaussians with respect to data set $\mathcal{P}$ is defined as a linear combinations of the given Gaussian distributions:

$$p(\mathcal{P}) = \sum_{j=1}^{K} \pi_j \mathcal{N}(\mu_j, \sigma_j^2),$$

where each Gaussian $\mathcal{N}(\mu_j, \sigma_j^2)$ is said a *component* of the mixture and each parameter π_j is the respective *mixing coefficient*. In addition, if $\sum_{j=1}^{K} \pi_j = 1$, then also $p(\mathcal{P})$ is a probability distribution. It is possible to find a Gaussian distribution $\mathcal{N}_{\mathcal{P}}(\mu, \sigma^2)$ computed as the approximation of the mixture of Gaussians $p(\mathcal{P})$. That is, distribution $\mathcal{N}_{\mathcal{P}}(\mu, \sigma^2)$ is the closest distribution to $p(\mathcal{P})$ that is not a mixture of Gaussians, according to the Kullback-Leibler distance (see [6]). The mean μ and the variance σ^2 of $\mathcal{N}_{\mathcal{P}}(\mu, \sigma^2)$ can be easily computed as:

$$\mu = \sum_{j=1}^{K} \pi_j \mu_j \quad \text{and} \quad \sigma^2 = \sum_{j=1}^{K} \pi_j (\sigma_j^2 + \mu_j^2) - \mu^2,$$

where $\pi_1, \ldots, \pi_K$ correspond to the mixing coefficients of the mixture of Gaussians $p(\mathcal{P})$, as described in [23].

IV. Our Trust Mechanism

A. Description of the Framework and Assumptions

Let us assume a distributed storage system involving a set $\mathcal{S} = \{S_1, \ldots, S_n\}$ of n storage servers. Trust values $\tau_1^{(t)}, \ldots, \tau_n^{(t)} \in [0, 1]$ are assigned, respectively, to storage servers $S_1, \ldots, S_n$ at time t. These trust values convey information about how trustworthy they are expected to be. More precisely, these storage servers periodically interact with each other to manage the storage of documents. After each interaction at time t, the storage servers evaluate each other and update trust values $\tau_1^{(t-1)}, \ldots, \tau_n^{(t-1)}$ at time $t-1$ taking into account the current performance. To simplify the notation, from now on, trust values $\tau_1^{(t)}, \ldots, \tau_n^{(t)}$ at time t are simply denoted by $\tau_1, \ldots, \tau_n$. Storage servers $S_1, \ldots, S_n$ are owned by multiple Cloud providers. That is, the set $\mathcal{S}$ can be seen as the union $\mathcal{S} = \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_m$, with $m \leq n$, where $\mathcal{S}_l = \{S_{l,1}, \ldots, S_{l,m_l}\}$ is the set of storage servers owned by the l-th Cloud provider, with $n = \sum_{l=1}^{m} m_l$, for $l = 1, \ldots, m$. Cloud providers are interested in maximizing the trust values of their storage servers, because this leads to a greater income. Thus, it is not only necessary that Cloud providers provide the distribute storage system with high performing and trustworthy storage servers. In fact, it might also be convenient to blame other storage servers owned by different Cloud providers, no matter how well they actually behave. This is regarded as collusion among storage servers owned by the same Cloud provider. We make the following two assumptions with respect to how storage servers may collude.

1) Assumption 1: Only storage servers owned by the same Cloud provider can collude.

2) Assumption 2: The behavior of storage servers is assumed to be consistent among all storage servers. That is, a storage server does not choose to behave differently according to the provenance of the storage servers it is interacting with. On the contrary, a storage server may submit dishonest evidence for other storage servers according to their provenance.

3) Assumption 3: All evidence is submitted to a central peer responsible for computing the trust values. The central peer is always honest and does not tamper with the evidence.

Assumption 1 and Assumption 2 emphasize that what we focus on here is coping with the possibility of unreliable evidence (i.e. unreliable ratings) submitted by storage servers with the aim of maximizing the overall trustworthiness of their Cloud provider. Our trust mechanism aims at mitigating the collusion of the storage servers belonging to the same Cloud provider under Assumption 1, Assumption 2, and Assumption 3. More precisely, the computation of trust value τ_i is computed by two measurements: the so called first and second partial trust values. The *first partial trust value* τ_i' for storage server S_i is the result of evidence submitted by storage server S_j, for $j = 1, \ldots, n$ and $j \neq i$ (see Section IV-B). This first step is somewhat similar to what has been done for Bayesian models (see Section II), except that the evidence processed for the computation of the first partial trust value has different relevance depending on the reputation of the source. The *second partial trust value* τ_i'' depends on how reliable the evidence submitted by storage server S_i are with respect to storage server S_j, with $j \neq i$ (see Section IV-C). This second step is where unreliable evidence is detected and the submitter is discouraged to do so because its trustworthiness is decreased. In Section IV-D, it is described how to merge these two measurements to obtain trust value τ_i.

B. Computation of the First Partial Trust Value τ_i'

This section describes how the first partial trust value τ_i' for storage server S_i is computed, which takes into account the evidence submitted by all the other storage servers. The computation of the first partial trust value τ_i' is performed in an Euclidean space of dimension $D = 2$. For readability, we divide this computation into steps.

1) Collection of the evidence: Each storage server S_j submits point $P_j^{(i)} = (x_{P_j^{(i)}}, y_{P_j^{(i)}})$ with respect to storage server S_i. The first coordinate of point $P_j^{(i)}$ is $x_{P_j^{(i)}} = \tau_j^{(t-1)} \in [0, 1]$, where $\tau_j^{(t-1)}$ is the reputation of the evaluator storage server S_j. In fact, being $\tau_j^{(t-1)}$ the trust value computed at time $t-1$, it can be seen as the reputation gained by storage server S_j up to that moment. The second coordinate of point $P_j^{(i)}$ is $y_{P_j^{(i)}} = \sigma_j^{(i)}$, where $\sigma_j^{(i)} \in [0, 1]$ is the evidence by which storage server S_j evaluates storage server S_i. In other words, $\sigma_j^{(i)}$ is the expectation that storage server S_j has with respect to the future behavior of storage S_i.

2) Representation of τ_i': Since the evidence relative to the trustworthiness of storage server S_i is represented as a value between 0 and 1, the first partial trust value τ_i' is also a value between 0 and 1. The idea is to define the data set $\mathcal{P}^{(i)} = \{P_1^{(i)}, \ldots, P_{i-1}^{(i)}, P_{i+1}^{(i)}, \ldots, P_n^{(i)}\}$ of points submitted by storage server S_j, for $j = 1, \ldots, n$ and $j \neq i$. Then, the machine learning techniques of K-means clustering and mixture of Gaussians (see Section III-B) are used to extract the first partial trust value $\tau_i^{(i)}$ from coordinate $y_{P_j^{(i)}}$ of each point in the data set $\mathcal{P}^{(i)}$.

3) Classes of credibility: K classes of evidence are distinguished with respect to their credibility by the K-means clustering algorithm. The points in the data set $\mathcal{P}^{(i)}$ are grouped into K clusters $\mathcal{C}_1, \ldots, \mathcal{C}_K$. In fact, each point in the data set $\mathcal{P}^{(i)}$ is a tuple corresponding to the values "reputation of the rater" and the values "the submitted rate/evidence". Therefore, the clustering algorithm finds classes which take into account both values. The center points $M_1, \dots, M_K$ of clusters $\mathcal{C}_1, \dots, \mathcal{C}_K$ simplify these classes of credibility with fewer, yet more informative points.

4) Assigning a weight $\omega_j^{(i)}$ to point $P_j^{(i)}$: Each point $P_j^{(i)}$ is submitted by storage server j, which has a reputation $\tau_j^{(t-1)}$. We wish to use this reputation to weight point $P_j^{(i)}$. We define weight $\omega_j^{(i)}$ as $\omega_j^{(i)} = \frac{F(\tau_j^{(t-1)})}{\sum_{j=1}^{n-1} F(\tau_j^{(t-1)})}$, where $F : [0,1] \to \mathbb{R}$ is a positive and increasing function over the interval $[0,1]$, which assigns higher scores for larger trust value $\tau_j^{(t-1)}$. The meaning of this function is to define how to balance the influence of a low-weighted reputation against a high-weighted reputation. In other words, function $F(x)$ determines how many low-weighted reputations are needed to have as much influence so as to overcome one high-weighted reputation. In fact, storage servers with a high reputation might also submit incorrect evidence. If many storage servers, even with a low reputation, submit different evidence in contrast to the highly reputable storage server, then their opinion have also an impact. A possible approach to define function $F(x)$ is to choose a known increasing function (such as the logarithmic function) and adjust the eccentricity according to the number of low-weighted reputations necessary to balance higher-weighted reputations. Note that function $F(x)$ does not need to be continuous. In fact, another possible approach to define function $F(x)$ is to create a step function over disjoint subintervals of the interval $[0,1]$. Thus, two trust values within the same subinterval are considered equivalent with respect to the reputation of the storage server they represent.

5) Computation of τ_i': The first partial trust value τ_i' is computed as a weighted combination of coordinates $y_{M_1}, \dots, y_{M_K}$ of the center points $M_1, \dots,\ M_K$, respectively. Center points $M_1, \dots, M_K$ are not equivalent: together with the classes of credibility they distinguish, they depend on the cardinality of the respective clusters. The idea is to associate values $\pi_1, \dots, \pi_K$ to center points $M_1, \dots, M_K$ in quantitative and qualitative manner. Values $\pi_1, \dots, \pi_K$ are regarded as the mixing coefficients of a mixture $p(\mathcal{P}^{(i)})$ of K Gaussian distributions $\mathcal{N}_1(\mu_1, \sigma_1^2), \dots, \mathcal{N}_K(\mu_K, \sigma_K^2)$. More precisely, the points within each cluster $\mathcal{C}_l$ can be seen as following a Gaussian distribution with $\mu_l = M_l$, for $l = 1, \dots, K$. That is because the mean and the variance of a Gaussian distribution convey information about where the points are mostly concentrated and how they are spread, which is comparable to the information conveyed by the clusters. Weights $\omega_j^{(i)}$ are used to compute the mixing coefficients $\pi_1, \dots, \pi_K$. In more detail, $\pi_l = \sum_{j=1}^{n_l} \omega_{l_j}^{(i)}$, where n_l is the cardinality of cluster $\mathcal{C}_l$ and $\omega_{l_j}^{(i)}$ is the weight assigned to point $P_{l_j}^{(i)} \in \mathcal{C}_l$, for $l = 1, \dots, K$. Note that $\sum_{l=1}^{K} \pi_l = 1$ and thus the mixture $p(\mathcal{P}^{(i)})$ is a probability distribution. The weighted sum of means $\mu_1, \dots, \mu_K$ represents the mean μ of the closets Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ approximating mixture $p(\mathcal{P}^{(i)})$ (see Section III-B). And this is exactly what we aim at: since the means $\mu_1, \dots, \mu_K$ are the center points $M_1, \dots, M_K$, we can now compute the first partial trust value τ_i' as the weighted sum of coordinates $y_{M_1}, \dots, y_{M_K}$ according to the mixing coefficients $\pi_1, \dots, \pi_K$. That is,

$$\tau_i' = \sum_{l=1}^{K} \pi_l \cdot y_{M_l} \in [0,1].$$

Basically, the first partial trust value is computed as coordinate y_μ of the mean μ of the fitting Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$. One can argue that the first partial trust value τ_i' could be computed directly after clusters $\mathcal{C}_1, \dots, \mathcal{C}_K$ were distinguished, without passing through the step of computing the mixture of Gaussians. This is, in fact, what one would practically do when computing τ_i'. However, we highlight that this computation is possible because the center points of the clusters model are the means of Gaussian distributions.

C. Computation of the Second Partial Trust Value τ_i''

This section describes how the second partial trust value τ_i'' for storage server S_i is computed, which takes into account the reliability of the evidence submitted by storage server S_i itself with respect to all the other storage servers. We recall that the second partial trust value is meant to distinguish submitted reliable evidence from unreliable evidence and to, respectively, encourage and discourage such submission. Just as with the first partial trust value τ_i', the computation of the second partial trust value τ_i'' is performed in an Euclidean space of dimension $D = 2$. For readability, we divide this computation into steps.

1) Collection of the evidence: The evidence collected is $\sigma_i^{(j)} \in [0,1]$, i.e. the coordinate $y_{P_i^{(j)}}$ of point $P_i^{(j)}$ submitted by the evaluator storage server S_i with respect to storage server S_j, for $j = 1, \dots, n$ and $j \neq i$, where point $P_i^{(j)} = (x_{P_i^{(j)}}, y_{P_i^{(j)}})$ is defined in Section IV-B.

2) Representation of τ_i'': Since the evidence relative to the reliability of the submissions of storage server S_i is represented as a value between 0 and 1, the second partial trust value τ_i'' is also a value between 0 and 1. The reliability of evidence $\sigma_i^{(j)}$ is measured by distance $d(\tau_j', \sigma_i^{(j)})$, where τ_j' is the first partial trust value of storage server S_j. Distance $d(\tau_j', \sigma_i^{(j)})$ is at most 1. If distance $d(\tau_j', \sigma_i^{(j)})$ is close to 1, this is an indicator of the dishonesty of storage server S_i when rating storage server S_j, i.e. it is an indicator that $\sigma_i^{(j)}$ is an unreliable piece of evidence. On the contrary, if distance $d(\tau_j', \sigma_i^{(j)})$ is close to 0, this is an indicator of the honesty of storage server S_i when rating storage server S_j, i.e. it is an indicator that $\sigma_i^{(j)}$ is a reliable piece of evidence.

3) Ranges of reliability: Ranges of reliability are spanned to grant trustworthiness to storage server S_i when distance $d(\tau_j', \sigma_i^{(j)})$ is small and vice versa. More precisely, a score $o \in [0,1]$ is defined to represent how much the trustworthiness increases or decreases when the submission of score $\sigma_i^{(j)}$ is reliable or unreliable, respectively. Three ranges of reliability are defined by $\alpha_1, \alpha_2 \in [0,1]$. If $0 \leq d(\tau_j', \sigma_i^{(j)}) \leq \alpha_1$, then the trustworthiness of storage server S_i increases by o. If $\alpha_2 \leq d(\tau'_j, \sigma_i^{(j)}) \leq 1$, then the trustworthiness of storage server S_i decreases by o. If $\alpha_1 < d(\tau'_j, \sigma_i^{(j)}) < \alpha_2$, then the trustworthiness of storage server S_i remains unchanged. We denote by $o_i^{(j)} \in \{-o, o, 0\}$ the trustworthiness lost, gained, or maintained by the evaluator storage server S_i when submitting score $\sigma_i^{(j)}$, for $j = 1, \ldots, n$ and $j \neq i$. In this way, storage servers are encouraged to submit reliable evidence and discouraged to submit unreliable evidence. In fact, if storage servers submit time after time evidence that are considered unreliable, then they progressively lose more and more trustworthiness. This is regarded as a countermeasure against submitted unreliable evidence, as the influence of the submitter decreases. Note that we choose to have three ranges of reliability for simplicity but more ranges can be spanned.

4) Computation of τ''_i: The second partial trust value τ''_i is computed from the reputation $\tau_i^{(t-1)}$ of storage server S_i at time $t-1$, taking into account the average reliability of the $n-1$ scores $\sigma_i^{(j)}$ it submitted, for $j = 1, \ldots, n$ with $j \neq i$. That is,

$$\tau''_i = \tau_i^{(t-1)} + \frac{1}{n-1} \sum_{j=1, j \neq i}^{n} o_i^{(j)}.$$

In this way, the second partial trust value τ''_i is computed by increasing $\tau_i^{(t-1)}$ if the scores $\sigma_i^{(j)}$ are on average reliable or by decreasing $\tau_i^{(t-1)}$ if the scores $\sigma_i^{(j)}$ are on average unreliable. Note that the computation of the second partial trust value τ''_i is recursive. That is, the history of the behavior of storage server S_i in the previous rounds is taken into account by the term $\tau_i^{(t-1)}$. In fact, reputation is built upon consistent increments over a long period of time and is not greatly affected, both positively and negatively, in one single round.

D. Computation of Trust Value τ_i

Trust value τ_i is computed as a convex combination of τ'_i (defined in Section IV-B) and τ''_i (Section IV-C). That is, parameters $\eta_1, \eta_2 \in [0, 1]$ are selected such that $\eta_1 + \eta_2 = 1$ and trust value τ_i is computed as:

$$\tau_i = \eta_1 \cdot \tau'_i + \eta_2 \cdot \tau''_i.$$

Parameters η_1, η_2 hold for the computation of each trust value τ_i, for $i = 1, \ldots, n$. They are chosen based on the requirements of the specific distributed storage system. In some situations, it might be more desirable to assign more weight to a server performing well rather than a server rating honestly and vice versa.

V. Application to the Social Secret Sharing Protocol AS^3

In this section, the trust mechanism introduced in Section IV is applied to distributed storage systems based on social secret sharing. In this setting, the protection of the stored document is enhanced because the trust values are more accurate with respect to the actual trustworthiness of the storage servers. Therefore, the overall system is more resilient to honest but curious and faulty storage servers, because they are provided with the least informative shares. In particular, we consider the social secret sharing protocol AS^3 presented in [25].

A. Overview of the AS^3 Protocol

The peculiarity of the AS^3 protocol is that two trust values are computed separately for each storage server. The first measures the trustworthiness with respect to confidentiality and the second measures the trustworthiness with respect to retrievability. After the update of the trust values, the amount of potential honest but curious storage servers and the amount of potential faulty storage servers are estimated. Finally, for each storage server, a unique trust value is computed as a weighted sum of the trust value for confidentiality and the trust value for retrievability. The shares are distributed according to it, i.e. more informative shares are distributed to the storage servers that have a high trust value, and vice versa. Since they are related to two independent protection goals, countermeasures against honest but curious and faulty storage servers are taken separately. The countermeasure against honest but curious storage servers is: the reconstruction threshold of the underlying secret sharing scheme is increased such that it is not possible for them to retrieve the document. The countermeasure against faulty storage servers is: to broadcast a warning message if the document cannot be retrieved in case all of them would not respond to the reconstruction. The warning messages bootstrap new and better functioning storage servers. These countermeasures are expensive, both with respect to computation overhead and to the cost for maintaining the storage servers. Distribution of shares leading to situations when these countermeasures have to be taken often must be therefore prevented.

B. Improvements and Performances

Whenever the trust values are updated, our trust mechanism is run twice: once to compute the trust values of the storage servers with respect to confidentiality and once with respect to retrievability. In the framework of social secret sharing, it is commonly assumed that the Byzantine model's requirement holds [3]. That is, the amount of colluding storage servers is bounded by the reconstruction threshold. That is, the colluding storage servers do not have enough information to reconstruct the document by themselves. Furthermore, we recall that when our trust mechanism is applied to social secret sharing, also Assumption 1, Assumption 2, and Assumption 3 must hold as well (see Section IV). Note that, the Byzantine model's requirement and Assumption 1 lead to the fact that the amount of storage servers owned by the same Cloud provider is bounded by the reconstruction threshold as well. That is, a Cloud provider, even by enforcing its storage servers to collude, cannot retrieve the document alone. Furthermore, the function $F(x)$ defined in Section IV-B can be instantiated here such that the number of low-weighted reputations balancing with one high-weighted reputation is the reconstruction threshold. In this way, collusion of storage servers owned by the same Cloud provider is overcome.

We run the AS^3 protocol using CertainTrust [20] as the trust mechanism (as it was originally presented in [25]) and we run AS^3 using the trust mechanism presented in Section IV. In both cases, the input parameters to the AS^3 protocol are $\lambda_c = \lambda_r = 0.5$, $T_c = T_r = 0.2$, and $(n_1, n_2, n_2) = (3, 5, 8)$ for $K \in \{3, 4\}$. The parameters $\lambda_c, \lambda_r \in [0, 1]$ are the weights with respect to confidentiality and retrievability used to compute the final trust value. The parameters $T_c, T_r \in [0, 1]$ are the thresholds for confidentiality and retrievability below which storage servers are considered honest but curious and faulty, respectively. The parameter n_i is the number of storage servers owned by the i-th Cloud provider. The clustering algorithm distinguishes K classes of credibility (see Section IV).

We run 1000 iterations and observe the average round when the first and second countermeasures are applied. Here, "round" corresponds to the notion of "time" used in Section IV to denote when the trust values are updated. Table I shows that, when using our new trust mechanism, both countermeasures are performed at later rounds. That is, the two countermeasures are performed less often. Thus, when our trust mechanism is used, the AS^3 protocol becomes more resilient to honest but curious and faulty storage servers because the distribution of the shares is more accurate. In fact, the function $F(x)$ (Section IV-B) and the second partial trust value (Section IV-C) are defined such that it is possible to mitigate the effect of colluding storage servers providing unreliable evidence up to the reconstruction threshold. Assumption 1 and the Byzantine's model requirement ensure that no more than this amount of storage servers is interested in colluding. In Table I, the AS^3 protocol is denoted by "CT-AS^3" when CertainTrust is used and is denoted by "K-AS^3" when our new trust mechanism based on K-means clustering is used.

TABLE I: Average round where the first and second countermeasures are applied (the higher, the better).

	CT-AS^3	K-AS^3
1^{st} countermeasure	3.4	5.7
2^{nd} countermeasure	23.86	31.16

Extensive discussions about the result can be found in the extended version of this [26].

VI. Conclusion and Future Work

In this paper, an evidence-based trust mechanism for distributed storage systems was proposed that detects unreliable evidence and establishes countermeasures in order to discourage the collusion of storage servers owned by opportunistic Cloud providers. The mechanism was applied to the social secret sharing protocol AS^3, outperforming the trust mechanisms previously used in this context. We remark that our evidence-based trust mechanism can also benefit other frameworks where peers have to evaluate each other and can be extended to compute the trust values according to additional criteria, such as the amount of past interactions. Furthermore, we plan to design a new bootstrapping procedure that improves upon [8] so as to better address the scenario of distributed storage systems.

Acknowledgments

We thank G. Salvaneschi, M. Horsch, and A. Palesandro for useful discussions. Work co-funded by the DFG as part of projects "Scalable Trust Infrastructures" and "Long-Term Secure Archiving" within the CRC 1119 CROSSING and the European Union's Horizon 2020 research and innovation program under Grant Agreement No 644962.

References

[1] C. Bishop. Pattern recognition and machine learning. 2006.

[2] S. Buchegger and J-Y Le Boudec. A robust reputation system for peer-to-peer and mobile ad-hoc networks. In *P2PEcon 2004*, 2004.

[3] R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. The round complexity of verifiable secret sharing and secure multicast. In *STOC*, 2001.

[4] C-W Hang and M. P. Singh. Trustworthy service selection and composition. *ACM Trans. Auton. Adapt. Syst.*, 2011.

[5] A. Josang and R. Ismail. The beta reputation system. In *Proceedings of the 15th bled electronic commerce conference*, 2002.

[6] S. L. Lauritzen. *Graphical models*. Clarendon Press, 1996.

[7] H. W. Lauw, E-P Lim, and K. Wang. Bias and controversy: Beyond the statistical deviation. In *SIGKDD*, 2006.

[8] X. Liu, A. Datta, K. Rzadca, and E-P Lim. Stereotrust: A group based personalized trust model. In *Conference on Information and Knowledge Management*, 2009.

[9] X. Liu, G. Tredan, and A. Datta. A generic trust framework for large-scale open systems using machine learning. *Computational Intelligence*, 2014.

[10] T. Loruenser, A. Happe, and D. Slamanig. Archistar: towards secure and robust cloud based data sharing. In *CloudCom*, 2015.

[11] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Berkeley symposium on mathematical statistics and probability*, 1967.

[12] M. E. G. Moe, B. E. Helvik, and S. J. Knapskog. *Comparison of the Beta and the Hidden Markov Models of Trust in Dynamic Environments*. 2009.

[13] A. W. Moore. The anchors hierarchy: Using the triangle inequality to survive high dimensional data. In *Conference on Uncertainty in artificial intelligence*, 2000.

[14] M. Nielsen, K. Krukow, and V. Sassone. A bayesian model for event-based trust. *Electronic Notes in Theoretical Computer Science*, 2007.

[15] M. Nojoumian and T. C. Lethbridge. A new approach for the trust calculation in social networks. In *International Conference on E-Business*, 2008.

[16] M. Nojoumian and D. R. Stinson. Brief announcement: Secret sharing based on the social behaviors of players. In *PODC*, 2010.

[17] M. Nojoumian, D. R. Stinson, and M. Grainger. Unconditionally secure social secret sharing scheme. *IET*.

[18] Mehrdad Nojoumian and Douglas R. Stinson. Social secret sharing in cloud computing using a new trust function. In *PST*, 2012.

[19] V Ramasubramanian and KK Paliwal. A generalized optimization of the kd tree for fast nearest-neighbour search. In *TENCON*, 1989.

[20] S. Ries. Extending bayesian trust models regarding context-dependence and user friendly representation. In *SAC*, 2009.

[21] D. Slamanig, A. Karyda, and T. Lorünser. Prismacloud–privacy and security maintaining services in the cloud. *ERCIM NEWS*, 2016.

[22] J. Tang, H. Gao, H. Liu, and A. Das Sarma. etrust: Understanding trust evolution in an online world. In *SIGKDD*, 2012.

[23] L. Trailovic and L. Y. Pao. Variance estimation and ranking of gaussian mixture distributions in target tracking applications. In *Decision and Control*, 2002.

[24] G. Traverso, D. Demirel, and J. Buchmann. Dynamic and verifiable hierarchical secret sharing. In *ICITS*, 2016.

[25] G. Traverso, D. Demirel, S. M. Habib, and J. Buchmann. As^3: Adaptive social secret sharing for distributed storage systems. In *PST*, 2016.

[26] Giulia Traverso, Carlos Garcia Cordero, Mehrdad Nojoumian, Reza Azarderakhsh, Denise Demirel, Sheikh Habib, and Johannes A Buchmann. Evidence-based trust mechanism using clustering algorithms for distributed storage systems. *IACR Cryptology ePrint Archive*, 2017.