LoT: a Reputation-based Trust System for Long-term Archiving

Martín Vigil, Denise Demirel, Sheikh Mahbub Habib, Sascha Hauke,
Johannes Buchmann, and Max Mühlhäuser

Technische Universität Darmstadt
Hochschulstr. 10, 64289 Darmstadt, Germany
Email: {vigil, ddemirel, buchmann}@cdc.tu-darmstadt.de
{sheikh, hauke, max}@tk.tu-darmstadt.de

***Abstract*—Digital archiving systems are necessary to store documents for several years, such as electronic health records. However, security breaches in these systems may allow attackers to tamper with archived documents without being noticed. To address this threat, standardized archiving systems require a public key infrastructure, where a time-stamp authority is trusted to date and sign stored documents periodically. However, in practice a time-stamp authority may not be fully trustworthy, allowing an attacker to forge documents. Thus, in this paper, we introduce a novel reputation-based trust system for time-stamping-based archiving called *Long-term evaluation of Trust* (LoT), which alleviates the required trust assumptions. This makes LoT an important contribution to realize trust and security management for digital archiving systems using public key infrastructures. We implemented LoT showcasing its applicability to electronic health records and demonstrate its efficacy by simulations.**

Keywords–Digital Archiving; Time-Stamping; Reputation System; Trust; Electronic Health Record.

I. INTRODUCTION

In the field of long-term archiving [1], an important goal is maintaining the integrity and authenticity of stored data over long periods of time. An example of long-term stored data are electronic health records (EHRs), which should be preserved for the entire lifetime of a patient. The deployment of EHRs is already in progress: by 2020, the British National Health Service plans to make their patients' health and care records digitally available as a part of their "Personalized Health and Care 2020" strategy [2].

In the long run, attackers can gain access to archiving systems and tamper with archived data without being noticed. This can cause serious issues, such as physicians using forged electronic health records to prescribe wrong treatments. To cope with this security threat, existing long-term archiving schemes assume the existence of a trusted third party. This party periodically checks and/or signs the documents. The only standardized long-term archiving solutions are the time-stamping-based schemes (e.g., [3]). These schemes require public key infrastructures, where a trusted time-stamp authority (TSA) dates and signs documents by time-stamping them. A time-stamp allows users to verify when a document existed and whether it has been modified since. For the purpose of verification, users are bound to *trust* that the TSA has provided the correct date and time in the time-stamp. Since time-stamps have limited lifetime, e.g., TSAs' signatures expire, a single time-stamp cannot guarantee integrity and authenticity of a document for an indefinite period of time. Therefore, a chain of time-stamps is necessary. The first time-stamp authenticates the document and the subsequent time-stamps authenticate the previous ones, i.e., expired time-stamps. In this case, users have to trust *every* TSA involved in constructing the time-stamp chain.

However, the assumption of the existence of fully trustworthy TSAs raises security concerns regarding time-stamping-based schemes if TSAs turn out to be untrustworthy. For instance, a malicious TSA can time-stamp a forged signature using a particular date in the past when the corresponding signature key was still valid. Also, a single suspicious TSA is sufficient to cast doubt on the correctness of a time-stamp chain and of the corresponding archived document. It is somewhat surprising that such an issue has not been addressed yet, although TSAs are key actors of long-term archiving solutions and can be targets of attacks similar to that of certification authorities (see [4] for an overview of these attacks).

In order to alleviate the need to assume fully trusted time-stamp authorities, in this paper we propose a novel reputation-based trust system for long-term archiving solutions that is called *Long-term evaluation of Trust* (LoT). LoT evaluates the trustworthiness of each and every TSAs along with their issued time-stamps. Our proposed system provides users of long-term archiving systems the power to assess how likely it is that the TSAs provide correct time-stamps, documents are uncorrupted and authentic, as well as to determine when the documents have to be re-authenticated.

More precisely, the main contributions of this paper are as follows:

- Mechanisms to collect and process experiences regarding the TSAs and their issued time-stamps.
- Extended mechanisms for trust evaluation of the TSAs and time-stamps.
- A decision mechanism based on trust scores.
- A trust resetting mechanism.

The contributions are evaluated by means of simulation using our implemented trust system. The goal of the evaluation is to showcase the applicability and efficacy of our proposed trust system in the context of archiving EHRs.

Digital archiving systems should also guarantee the confidentiality of stored documents. However, this security goal is not addressed in this work because the current archiving systems do not provide confidentiality in the long run. More precisely, there is no solution available that guarantees both long-term integrity and information-theoretic confidentiality.

 ISBN: 978-1-61208-493-0

Nevertheless, this is a vital research field, and we plan to address this security goal in future work (see [5] for preliminary results).

The remainder of this work is organized as follows. Related work is provided in Section II. The background for the reader to understand our contribution is provided in Section III. Our contributions are presented in Section IV and evaluated in Section V. In Section VI, we draw the conclusion and discuss our future work.

II. Related work

In this section, we briefly discuss the state-of-the-art regarding long-term archiving schemes and trust systems.

Long-term Archiving. Lekkas and Gritzalis [6] propose a long-term archiving scheme where digital signatures are used to guarantee that EHRs have not been forged. Moreover, the scheme assumes the existence of trusted third parties, the so-called notaries, that verify and renew these signatures regularly. Vigil et al. [7] soften this trust assumption by proposing a peer-to-peer network of notaries, where one notary checks that another notary verifies and renews signatures properly. However, the reputation of notaries is not available for users in the long term. Besides notary based solutions, the approaches using TSAs [8] are very promising and have been even standardized (e.g., [3]). However, for this type of archiving scheme no trust evaluation is available.

Trust & Reputation. Trust models and reputation systems have been proposed in various environments [9][10][11]. For instance, electronic marketplaces, peer-to-peer systems, and cloud computing. A number of commercial instances of such systems are also available, such as eBay and Amazon. Moreover, Braun et al. [12] propose a reputation system called CA-TMS to be used in public key infrastructures. More precisely, their system allows for assessing the trustworthiness level of certificate chains. However, none of the existing systems or models have been applied to long-term archiving schemes yet.

Our Scheme. To the best of our knowledge, our scheme is the first one to allow for trust evaluation when TSAs instead of notaries are used. Note that this significantly changes the archiving procedures and correspondingly how trustworthiness is evaluated. Moreover, our scheme provides a centralized reputation-based trust system where the reputations of TSAs and time-stamps are stored indefinitely. These reputations allow users to estimate the trustworthiness of TSAs and time-stamps even in the long run. In this work, we focus on classical time-stamping-based schemes where a time-stamp is signed by a single TSA as described in the standardized solution [3]. To analyze other approaches (e.g., a time-stamp is signed by multiple TSAs) is left for future work. Furthermore, we assume that the storage system stores the time-stamps generated. In addition, they can be published in a newspaper. However, note that this requires the existence of witnesses why a reputation system for the TSAs remains a useful tool.

III. Preliminaries

In this section, we explain how to use time-stamping to authenticate long-term data. Moreover, we present an adversary model for time-stamping and discuss the trust users put in digital signatures.

A. Long-term archiving of data

When using documents that already exist for long periods of time, it is necessary to check that these documents have not been forged since they were stored. To address this issue, several time-stamping schemes have been proposed with different security goals. In general, a time-stamping scheme generates an initial time-stamp to establish that a document and its signature existed at the date and time they were stored (proof of existence) and that they have not been changed since (integrity). Moreover, this time-stamp allows to verify the document authenticity even after the document signature becomes invalid or cryptographically insecure. However, the digital signature that guarantees the authenticity of the time-stamp is also valid and secure for a certain period of time only. After this period, the time-stamp and the document can be manipulated without being noticed. Therefore, it is necessary to renew the time-stamp timely. More precisely, before the time-stamp becomes insecure, the scheme generates a new one. This new time-stamp authenticates the old time-stamp. That is, the new time-stamp can be used to verify that the old time-stamp existed when the corresponding signature was valid and secure and that the old time-stamp has not been changed since. This process generates an endless time-stamp sequence which can be used to demonstrate that the document existed and was not changed since it was archived.

Fig. 1 illustrates an example of time-stamping. A signed document d is created at a time τ_0. This document is stored in an archive at a time $\tau_1 > \tau_0$. At this time, the signature on the document is still secure and a time-stamp s_1 is created. The time-stamp s_1 authenticates the document d, that is, s_1 can be used to verify that d existed at τ_1 and has not been changed since. The next time-stamps s_2 and s_3 are created at times $\tau_2 > \tau_1$ and $\tau_3 > \tau_2$, respectively. They authenticate the previous time-stamp before it becomes insecure. For example, s_2 authenticates s_1 at the time $\tau_2 < \tau_{2'}$, where $\tau_{2'}$ is when s_1 becomes insecure.

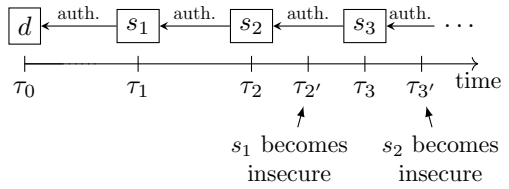

Figure 1. A signed document d and a sequence of time-stamps $s_1, s_2, \ldots$.

We now present the involved parties and procedures of time-stamping-based archiving schemes. The parties are *submitters*, *storage systems*, *TSAs*, and *retrievers*. *Submitters* send signed documents to storage systems. *Storage systems* store the submitted documents and can be realized, for instance, by building a cloud infrastructure. Additionally, storage systems request time-stamps to demonstrate that the submitted documents and their signatures have not been forged since their submission. Note that even if the document signature is no longer secure, the time-stamp ensures that neither the document nor the signature have been changed. *TSAs* are trusted third parties issuing the requested time-stamps. They are trusted to include the date and time when the time-stamps are issued in the time-stamps. *Retrievers* obtain signed documents and the corresponding time-stamps from the storage systems. They

verify the time-stamps to ensure that the retrieved data is not a forgery.

The long-term archiving of signed documents comprises the procedures *initialization*, *renewal*, and *verification*. During the initialization, a first time-stamp s_1 is generated showing that the signed document d existed at the date and time τ_1. The procedure is detailed below and illustrated in Fig. 2.

1) A submitter sends a document d to a storage system.
2) The storage system requests the first time-stamp s_1 by sending d to a TSA. (To be precise, the hash of d instead of d itself is sent. Since hash functions are only computationally secure, time-stamps are also renewed when the hash function is about to become insecure. However, since this does not affect the reputation system we omit this detail for legibility.)
3) The TSA issues $s_1 = TS(d)$, where TS is a function that creates time-stamps on the given input. More precisely, TS creates a signature σ_1 on the input d together with the current date and time τ_1 and returns $s_1 = (\tau_1, \sigma_1)$.
4) The TSA returns s_1 to the storage system.
5) The storage system stores d together with s_1.

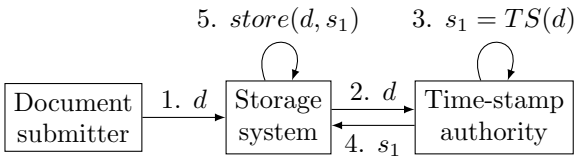

Figure 2. The initialization procedure.

The *renewal* procedure should be executed before the time-stamp s_1 becomes invalid. It generates a new time-stamp s_2 showing that s_1 existed at a date and time $\tau_2 > \tau_1$ when s_1 was still valid, and that s_1 has not been changed since. This procedure is explained next and depicted in Fig. 3.

1) The storage system requests s_2 by sending s_1 to a TSA.
2) The TSA issues $s_2 = TS(s_1)$. This time-stamp includes a signature σ_2 on s_1 together with τ_2, where $\tau_2 > \tau_1$ is the date and time when the TSA executed the function TS.
3) The TSA returns s_2 to the storage system.
4) The storage system stores d together with s_1 and s_2.

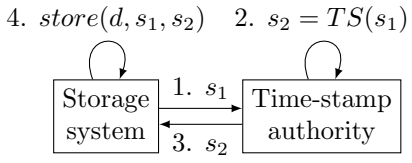

Figure 3. The renewal procedure.

The renewal procedure must also be performed for each time-stamp sequence $s_1, s_2, \ldots, s_k$ where the latest time-stamp s_k is about to become invalid. In this case, the storage system requests a time-stamp s_{k+1} by sending s_k to a TSA.

The *verification* procedure is executed by a retriever as follows. For $k > 1$, assume the retriever has obtained the signed document d and time-stamp sequence $s_1, \ldots, s_k$ from the storage system. Thus, he or she first verifies that s_k is a valid time-stamp using as time reference the date and time τ when he or she performs the verification. The verification includes checking that the signature σ_k contained in s_k is valid at the date and time τ. (Further properties are also verified, e.g., the security of key sizes, but we omit these details for ease of understanding and simplicity.) Similarly, for $j = k-1, \ldots, 1$ he or she verifies that s_j was a valid time-stamp at the date and time τ_{j+1} when the time-stamp s_{j+1} was generated. The date and time τ_{j+1} is found in the time-stamp s_{j+1}. Next, the verifier checks that the signature on d is valid at time τ_1 found in s_1. If the document signature and the k time-stamps are valid and the involved TSAs are trustworthy, then the retriever can be convinced that d is not a forgery, i.e., it existed on τ_1 and has not been changed since.

B. Adversary model

In this work, we will consider an adversary that is *active* and *mobile*. In the active adversary model, parties might deviate from the protocol. In addition, to meet long-term security, we assume that the attacker is mobile and might interact and corrupt different parties at different stages of the protocols' executions. Finally, we assume that the adversary can interact an unlimited time with the system but is computationally bounded each time he or she performs an attack (see [13] for a corresponding security model).

Long-term archiving comes with the following trust assumptions. First, the *TSAs* issue correct time-stamps, i.e., time-stamps containing the correct date and time. This must also be preserved at the presence of an attacker. More precisely, if there is an attacker who is able to control a *storage system*, then the TSAs do not collaborate with the attacker by issuing time-stamps containing the wrong date and time. Second, *certification authorities* issue correct certificates (proving the owner of a signature). In the following, we provide an example how a forgery could be carried out.

Assume our first trust assumption is not fulfilled and an attacker who gained access to the storage system is able to collaborate with a malicious TSA. In this case, the attacker can ask the TSA to issue time-stamps providing a date and time that is earlier or later than the date and time when the TSA in fact creates this particular time-stamp. This allows the attacker controlling the storage system to violate integrity and authenticity as follows. First, he or she takes a compromised signature key pair (e.g., of a doctor) and signs a forged document. Next, he or she requests a time-stamp containing a date and time from the malicious TSA, when the key pair of the signature was still secure and valid. This time-stamp can be used to convince a retriever that the attacker's document was signed by the corresponding doctor. However, such a back-dating attack needs the cooperation of a TSA, because the TSA is responsible for generating the evidence (i.e., time-stamps). Note that if an attacker controls a storage system, but there are no malicious TSAs that can collaborate with him or her, then the attacker can only forge time-stamps by using insecure and outdated signature keys and therefore will fail in generating an evidence that is valid at the current date and time. It follows that if all TSAs are trustworthy, not even a storage system that is controlled by an attacker is able to fake a time-stamp and violate integrity and authenticity. Therefore, a reputation-based

 ISBN: 978-1-61208-493-0

trust system is required to deal with untrustworthy TSAs as well as time-stamps in long-term archiving solutions.

C. Trust Opinions on Signatures

Besides the correctness of a time-stamp, an important criterion to trust or distrust in the integrity and authenticity of a document is the correctness of the signatures. Each time the submitter or the TSA generates a signature, it is necessary to trust that the signature key pair used is indeed owned by the signer. In practice, public key infrastructures are used where certification authorities (CAs) issue digital certificates proving the ownership. Since these CAs are *assumed* to be fully trusted, the reputation system CA-TMS (see Section II) can be used in addition to our proposed system, LoT. Each time the storage system receives a signature, it sends the certificate chain and a security parameter to CA-TMS and receives a trust score. Depending on the application, it either rejects "untrusted" signatures or stores the trust score and leaves the decision to the retriever.

IV. A NOVEL REPUTATION SYSTEM FOR LONG-TERM STORAGE

As discussed in the last section, if malicious TSAs issue fake time-stamps, i.e., time-stamps containing the wrong date and time, then retrievers may accept forged documents. Furthermore, the CA trust management system CA-TMS is *only* able to provide trust scores on the ownership of signature keys. Therefore, we propose a reputation-based trust system, LoT, to assess the trustworthiness of the TSAs and time-stamps associated with digital signatures. The proposed system will assign trust scores on the TSAs and time stamps based on the assessment. Typically, reputation-based trust systems are driven by direct experience and indirect experience, obtained via witness referrals [10]. Trust computation in such a system requires three distinct operational steps: i) experience or evidence collection and processing, ii) trust evaluation/assessment, and iii) making a trust-based decision. We next elaborate on these three steps in detail.

A. Experience collection and processing

In the collection phase, direct and indirect experiences provided are collected from the system's participants. An experience – realized as a binary value, either positive or negative – indicates whether a participant believes that a particular time-stamp contains the correct date and time. Participants, in the experience collection and processing phase, are document submitters, storage systems, and TSAs.

Collection phase. To provide their binary experience, participants verify whether the date and time contained in a time-stamp is correct. Initially, the LoT system provides the time-stamp to participants right after the time-stamp is created. Next, these participants verify that the date and time in the time-stamp is reasonably close to the date and time when they received the time-stamp (i.e., the current date and time). How much deviation from the current date and time is allowed is a parameter of LoT and depends on its application. For example, a deviation of half a day may be acceptable for electronic health records, but not for online auctions.

We extend the initialization and renewal procedures presented in Section III-A. The extended procedures use new parameters such as r and δ, where $r > 0$ is the number of TSAs that provide experiences and $\delta > 0$ is the maximum acceptable deviation from the current date and time. In the collection phase, the initialization procedure is as follows:

1) A submitter D sends his or her document d to a storage system.
2) The storage system selects a reputable TSA T from LoT.
3) The storage system requests a time-stamp on d from T.
4) T creates a time-stamp s containing a signature σ and the current date and time τ. T returns the time-stamp s to the storage system.
5) The storage system verifies s. More precisely, it computes an experience $e = \{0, 1\}$ on s such that $e = 1$ if $|\tau_c - \tau| \leq \delta$ otherwise $e = 0$, where τ_c is the date and time when the storage system received the time-stamp s, and τ is the date and time included in s. The storage system submits e, s, D, and T to the reputation system.
6) The reputation system stores e, s, and T together with the participant type *storageSystem*.
7) The reputation system randomly selects r different TSAs other than T that will be allowed to submit their experiences on s. We assume that this number of TSAs can always be found and that they are selected at random to reduce the chance of collusion against or in favor of T. The reputation system notifies the selected TSAs and the document submitter D. Moreover, the reputation system sets a deadline $\delta + \tau_r$ for the selected TSAs and D to submit their experiences, where τ_r is the moment when the reputation system notified D and the selected TSAs. Note that this allows to immediately identify when a time-stamp containing a wrong time has been generated.
8) The selected TSAs and the document submitter D compute their experiences on s as described in Step 5. Next, they submit their experiences to the reputation system.
9) The reputation system stores each submitted experience together with s, T, and the participant type *TSA* or *submitter*.

The same steps are performed during the renewal phase, which is used to generate the subsequent time-stamps. The only difference is that the submitter D does not participate in the renewal phase. This is because he or she may be no longer available after the document has been initially time-stamped. Moreover, LoT can publish the collected experiences, say, on a public board, for accountability reasons. Thus, participants can check that LoT has not changed their experiences.

Processing phase. The reputation system provides trust scores on TSAs and on time-stamps to participants. To compute trust scores, we rely on a well-established trust model, such as CertainTrust [14] or Subjective Logic [15]. In the following, we use CertainTrust. However, Subjective Logic can be readily substituted, as both models are isomorphic.

In CertainTrust model, trust scores are represented by means of the so-called *opinions*. They are represented as tuples $o = (t, c)$, where t represents a trust value and c a certainty value, indicating how confident one is that the trust score is representative. (We use a simplified opinion representation that omits the CertainTrust parameters f and w.)

Trust values are computed as the proportion of the sum of positive experiences, r, divided by the sum of all positive, r, and negative experiences, s, yielding $t = \frac{r}{r+s}$. If $r + s = 0$, $t = 0.5$. The computation of the certainty value is given in CertainTrust as $c = \frac{N \cdot (r+s)}{2 \cdot (N-r-s)+N \cdot (r+s)}$. The computation of the certainty value requires an additional parameter N which refers to the *maximal number of expected experiences*. How to set this parameter within LoT will be discussed below.

The trust score o_T on a TSA T and the trust score o_s on a time-stamp s are computed using the experiences the reputation system has collected. While in the latter case, only the experiences collected on s are considered, in the former case the entire history of T with respect to all the time-stamps $s_1, \ldots, s_k$ generated by T is taken into account. The reputation system collects experiences from distinct participants, where one type of participant may be more reliable than another type when providing their experiences. For instance, TSAs may be a more reliable source of experiences than document submitters. Therefore, the reputation system computes three different trust scores o^D, o^S, and o^T, where the labels D, S, and T identify scores derived from experiences given by document submitters, storage systems, and TSAs, respectively.

To compute the trust scores on a time-stamp, N_s can be set as follows. For document submitters and the storage system, N_s should be set to 1 because for every time-stamp only a single document submitter and single storage system can provide an experience. In contrast, we propose to use $N_s = r > 0$ for TSAs because r TSAs are selected randomly by the reputation system to provide an experience on a time-stamp (where r is also a public input parameter). For the trust score on a TSA, N_T can be defined as follows. Assume that the TSA has issued around $k > 0$ time-stamps. Then, N_T is given by multiplying the values of N_s by k.

B. Trust Evaluation

Assume a retriever requests a document from the storage system and wants to decide whether this document is a forgery or not. Then, the storage system verifies whether the corresponding time-stamp sequence $s_1, \ldots, s_k$ is valid, i.e., the time-stamps contain valid signatures. Afterwards, it sends the time-stamp sequence to LoT. LoT computes the trust scores on these time-stamps and involved TSAs from the experiences that have been collected so far. Next, LoT signs and returns the resulting trust scores to the storage system. The storage system then returns the requested document and trust scores to the retriever. Finally, the retriever uses the trust scores to decide whether to trust the document.

It may happen that the document has a valid time-stamp sequence, but the decision mechanism suggests that the document should not be trusted. To solve this issue, we propose a trust renewal procedure. Note that the retriever can also perform the above verification of the time-stamp sequences by him- or herself. In this case, he or she obtains the data needed from the storage system. Moreover, the retriever can obtain the experiences LoT published on the public board and run the trust evaluation. Therefore, the retriever neither has to trust the storage system nor our proposed system, LoT.

Next, we detail the trust evaluation of time-stamps and TSAs. After that, we present our decision mechanism and a procedure to renew trust.

Trust evaluation of time-stamp sequences. The trust opinion (score) o_s on a sequence of time-stamps s_i ($i = 1, \ldots, k$) is computed as follows. Assume that weights $w_D, w_S, w_T \in [0, 1]$ represent how much retrievers rely on the collected experiences from submitters, the storage system, and TSAs, respectively. These values can either be provided by retrievers or be public parameters. Initially, the trust scores $o^D_{s_i}$, $o^S_{s_i}$, and $o^T_{s_i}$ on each of the time-stamps s_i are calculated as described before. Then, the overall trust score o_{s_i} is calculated by aggregating the trust scores $o^D_{s_i}$, $o^S_{s_i}$, and $o^T_{s_i}$ on each time-stamp s_i according to the formula $o_{s_i} = o^D_{s_i} \widehat{\oplus}_w o^S_{s_i} \widehat{\oplus}_w o^T_{s_i}$, where $\widehat{\oplus}_w$ refers to the *Weighted Fusion* (W.FUSION) operator as defined in CertainLogic [16], an extension of CertainTrust. Essentially, the weighted fusion operation provides a weighted average over the scores, averaging and combining trust scores into an overall, fused score. This score uses the assigned weights w_D, w_S, and w_T and the certainty values, so that a trust score with a higher weight and/or a higher certainty value has a higher impact on the overall score. Finally, to gauge the trustworthiness of an entire time-stamp sequence, the trust score o_s on the time-stamp sequence is computed from the combined trust opinions $o_{s_1}, \ldots, o_{s_k}$ by calculating $o_s = o_{s_1} \wedge \ldots \wedge o_{s_k}$. Since the overall score should represent how much one can trust that **all** time-stamps contain the correct date and time, the AND operator $\wedge$ as defined in CertainLogic [17] is used. The CertainLogic $\wedge$ operator functions like a probabilistic AND operator over trust scores, considering not only the probabilistic trust value t but also the certainty value c.

Trust evaluation of TSAs. Assume there is a set of l TSAs T_j, where $j = 1, \ldots, l$, $0 < l \leq k$, and each of the TSAs signs at least one time-stamp contained in the time-stamp sequence $s_1, \ldots s_k$. Also, assume weights, $w_D, w_S, w_T \in [0, 1]$, representing how much retrievers rely on the collected experiences from submitters, storage system, and TSAs, respectively. Thus, the computation of o_T is analogous to calculating the trust score on the time-stamp sequence. More precisely, for every TSA T_j, the overall trust score o_{T_j} is calculated using the CertainLogic weighted fusion operator, the individual scores $o^D_{T_j}$, $o^S_{T_j}$, and $o^T_{T_j}$, and the assigned weights w_D, w_S, and w_T. Finally, the trust score o_T on TSAs is computed using the CertainLogic AND operator $\wedge$ and the combined trust scores $o_{T_1}, \ldots, o_{T_l}$, such that $o_T = o_{T_1} \wedge \ldots \wedge o_{T_l}$. The score o_T should tell how much we can trust that **all** involved TSAs generate time-stamps containing the correct date and time.

Decision mechanism. We propose a threshold-based decision mechanism which works as follows. The retriever defines trust thresholds α_s and α_T for the sequence of time-stamps and the TSAs, respectively. These thresholds are in the form of CertainTrust opinions. The values assigned as trust thresholds depend on the time-stamped document. For example, in a hospital using time-stamped electronic health records, thresholds should be as high as possible when the misuse of records are life threatening for patients (e.g., containing the wrong blood type). Other medical records, say, containing blood pressures, might be less harmful and to them a lower threshold can be assigned than to the life threatening ones.

The retriever provides α_s, α_T, o_s, and o_T as input for the decision mechanism. The trust decision is made as follows: if and only if $o_s \geq \alpha_s$ and $o_T \geq \alpha_T$ the mechanism outputs true, i.e., the document is trusted not to be a forgery. Otherwise,

 ISBN: 978-1-61208-493-0

false, i.e., the document may be a forgery.

Note that there is a good reason for only the trust opinion o_T on TSAs and the trust threshold α_T for TSAs being not sufficient to make trust decisions. It is because the trust opinion o_s on a time-stamp sequence and the trust threshold α_s for time-stamp sequences can prevent retrievers from accepting, for example, a sequence that contains a wrong time-stamp accidentally issued by a trustworthy TSA.

C. Resetting trust

Even though the time-stamps for a document have valid signatures, the document may not be trusted. We discuss two such situations where this problem may arise. Moreover, we propose a solution to this problem.

Assume there is a document and the corresponding time-stamp sequence in the storage system. The first situation can happen when one or more TSAs, issuing the time-stamp sequence, have bad reputation. The second situation can happen when the time-stamp sequence grows large in size. More precisely, the more time-stamps a sequence contains, the higher the chance that one of these time-stamps have been wrongly issued.

In order to address both situations, we propose the following solution. Assume that a document d has a time-stamp sequence $s_1, \ldots, s_k$ and can no longer be trusted by retrievers. Moreover, assume that there is an expert which is able to check whether the content of d is indeed correct. For example, if d is a document describing a patient's disease, the expert could be a physician that examines the patient to confirm that d is correct. Note that this is a necessary requirement to use the resetting procedure. If there is no expert to check the correctness, then this cannot be done. If there is an expert available he or she first verifies that the content of document d is correct and that it has been legitimately signed, say, by a doctor. If d fails the verification, then the expert alerts the storage system and the procedure ends. Otherwise, the expert re-signs d, generating d'. Then, he or she submits d' to the storage system. The storage system obtains a time-stamp s_{k+1} on d' together with d and $s_1, \ldots, s_k$. Afterwards, the storage system stores d' and s_{k+1} together with d and $s_1, \ldots, s_{k+1}$. Our proposed solution is illustrated in Fig. 4. Next, the storage system requests trust scores on the certificates of the signatures from CA-TMS and collects and stores experiences on s_{k+1} from the reputation system.

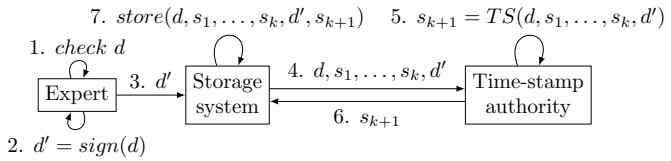

Figure 4. The resetting trust procedure.

Future retrievers use document d' instead of document d. Note that d' is protected by the time-stamps $s_{k+1}, s_{k+2}, \ldots$. The document d and time-stamps $s_1, s_2, \ldots$ can be checked for audit purposes. For example, auditors can check that d and $s_1, s_2, \ldots$ existed before d' and that d and d' are consistent.

V. Evaluation

In this section, we demonstrate the applicability and efficacy of our proposed trust system, LoT. First, we explain an application scenario related to EHRs. Second, we describe the implementation of a demo EHR application. Finally, we simulate the EHR scenario in order to demonstrate the efficacy of our proposed system.

A. Application scenario: Electronic Health Record

Health care institutions (e.g., hospitals) keep their patients' health information in the form of EHRs. An EHR is a container of documents, where each document provides specific information about the patient's health (e.g., the diagnosis of a disease). Additionally, the container includes time-stamp sequences for the documents proving the correctness of the information. The involved parties are storage systems, TSAs, the trust system, and physicians.

Storage systems, TSAs, and the trust system are the same as presented in Section IV. The storage systems can be either hosted by the health care institutions or outsourced to a global service, such as a private health care cloud. The trust system should be hosted by organizations other than the health care institutions and TSAs to prevent collusion. Physicians play the roles of document submitters and retrievers. They are submitters when they update a patient's EHR by adding new documents and are retrievers when they obtain documents from the storage system to learn the health conditions of patients. The EHR software allows to submit and retrieve EHRs and verifies the retrieved documents by checking their time-stamp sequences and trust opinions.

However, issues on performance and trust may occur in this scenario. Performance can be an issue if a retrieved document has a long time-stamp sequence (e.g., it has been stored for decades) and the device running the EHR software has low computing power (e.g., outdated tablet computers). To address this issue, the verification of time-stamp sequences can be pre-computed by the storage systems. Moreover, the computations on trust opinions for each time-stamp and TSA can be done by the reputation-based trust system. Note that in any case the EHR software can perform these verifications by itself, which we recommend for life-critical medical data. Trust issues can happen, for example, when the involved TSAs build bad reputation or time-stamp sequences grow largely (see Section IV-C). Therefore, the EHR software must also show when the retrieved document should not be trusted. Finally, note that physicians may be unable to provide their experiences to the trust system because they might not know how to verify the date and time contained in a time-stamp. Therefore, the EHR software performs such verification automatically when a new document is added to the storage system.

Assumptions. The majority of the involved parties is expected to provide reliable experiences by checking the date and time contained in time-stamps properly. Note that physicians and storage systems are interested in correct time-stamps to avoid being liable for forgeries. In contrast, a TSA wants to build a good reputation but may have no reasons to provide experiences on other TSAs' time-stamps. A possible solution is that the reputation-based trust system only publishes a TSA' reputation if this TSA provides experiences on request.

 ISBN: 978-1-61208-493-0

B. Electronic Health Records Software

We developed a demonstration software to show that physicians can easily use the reputation-based trust system. The demonstrator allows for creating EHRs for new patients, adding documents to existing EHRs, consulting EHRs, and renewing the trust in documents. Due to space limitation, we only describe how physicians consult EHRs and renew trust.

To consult an EHR, physicians use the interface depicted in Fig. 5 as follows. First, they enter the patient's name and birthday. Next, the software retrieves the documents from the patient's EHR found in the storage system. Then, the software obtains the required trust scores from the trust system and uses the decision mechanism to check whether the documents can be trusted. Finally, the software presents the corresponding documents in a table.

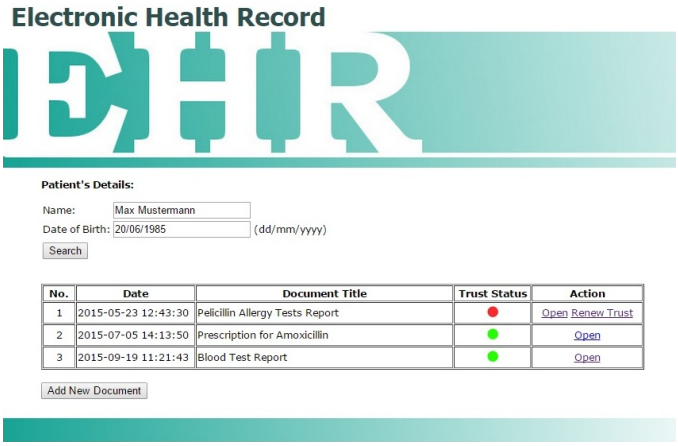

Figure 5. Consulting documents in an EHR.

The table shows for each document the date and time it was initially time-stamped, the document's title, and the trust status. A trust status marked with green means that the time-stamps are correct and that the document has passed the decision mechanism. In this case, physicians can trust the document and open it. In contrast, the red trust status alerts physicians that the document could be a forgery and that its trust should be renewed.

To renew trust in a document, the physician uses the interface illustrated in Fig. 6 as follows. First, the software shows the details and the content of the document. Next, the physician checks that the document contains correct information (e.g., he or she interviews the patient or request new exams). The document can be changed by the physician if necessary. Finally, he or she signs the document and the software submits the document to the storage system.

C. Simulation

To demonstrate the efficacy of the proposed reputation-based trust system in the long term, we simulate the storage of EHRs for 100 years as follows. Initially, submitters (physicians) send $10,000$ documents to a storage system and the initialization procedure is executed for each document. Next, because time-stamps are usually valid for up to five years, the renewal procedure is executed 19 times for each submitted document in order to guarantee its authenticity for 100 years.

Figure 6. Renewing the trust in a document.

These procedures request time-stamps from a pool of 60 TSAs. From these TSAs, one third issues time-stamps containing correct date and time with a probability of 0.90. The other two thirds issue correct time-stamps with probabilities of 0.85 and 0.80. After a TSA issued a time-stamp, we mimic the collection of experiences from the participants. The submitter, the storage system, and three TSAs check the date and time contained in this time-stamp. The submitter and the storage system do this properly with probabilities p equal to 0.50 and 0.90, respectively. TSAs check it properly with $p = 0.80$. That is, with p they submit *TRUE* to the trust system if the time-stamp contains the correct date and time and *FALSE* otherwise. Furthermore, with probability $1 - p$ they submit *TRUE* if the time-stamp contains wrong date and time and *FALSE* otherwise.

Next, we compute the probability that retrievers obtain a submitted document that has a time-stamp sequence containing at least one time-stamp with wrong date and time. We calculate this probability after executing the initialization procedure, i.e., when the time-stamp sequences of all submitted documents contain only one time-stamp. Furthermore, we compute the same probability every time after running the renewal procedure, i.e., for time-stamp sequences of 2–20 time-stamps.

We compare the calculated probabilities in three scenarios. In Scenario 1, neither our reputation system nor our decision mechanism is available. Therefore, TSAs are selected randomly from the pool and retrievers trust all documents in the storage system. In Scenario 2, only the reputation system is available. TSAs are selected randomly from the set of TSAs having the 10% highest trust opinions. Note that this leads to the *exploration versus exploitation problem* [18] which is also known for other scenarios, such as Amazon or eBay. How to deal with this requires more research and is out of scope for this work. To bootstrap the reputation-based trust system, $10,000$ time-stamps are issued by the TSAs in the pool, the experiences on these time-stamp are collected, and their reputation scores are computed. In addition to the trust system, in Scenario 3 the decision mechanism helps retrievers to decide whether to trust the documents. We use the trust

 ISBN: 978-1-61208-493-0

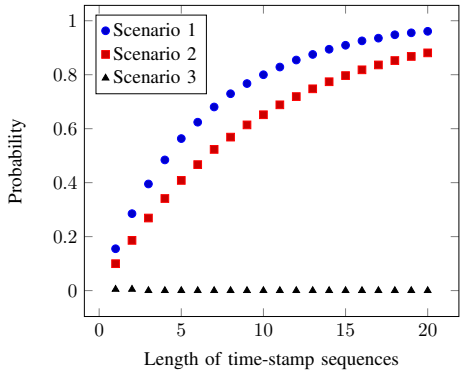

Figure 7. The probabilities that retrievers trust a document that could be a forgery.

opinion $(t = 0.60, c = 0.60)$ as trust threshold for time-stamp sequences and TSAs.

Fig. 7 shows that the probability of retrievers obtaining a forged document grows exponentially for Scenario 1 and 2. However, Scenario 2 is safer for retrievers because the used TSAs are more likely to provide correct time-stamps. By contrast, in Scenario 3 this probability is almost negligible. So it is prevented that a retriever accepts a forged document with high confidence.

However, note that if the trust opinions on time-stamp sequences and TSAs for a document are under the threshold, this does not automatically mean that the document must be a forgery. Therefore, the decision mechanism can produce false negatives, i.e., time-stamp sequences that are indeed correct may not be trusted by retrievers. In this case, if an expert is available the trust can be reset. However, one may want to do this as few as possible and in some scenarios there is even no expert allowing to reset the trust. In this sense, the simulation showed another advantage of using our proposed trust system. More precisely, when selecting preferably trustworthy TSAs, this reduces the chance that a forged time-stamp is generated. Consequently, the trustworthiness of time-stamp sequences decays more slowly.

VI. Conclusion & Future Work

Digital archiving systems store documents in the long term. A serious threat in these systems is that attackers may explore security breaches to tamper with stored documents without being noticed. To address this, a standardized archiving system is available that uses a public key infrastructure, where trusted time-stamp authorities (TSAs) date and sign stored documents periodically. However, in practice TSAs may not be fully trustworthy and could collude with attackers. To cope with this issue, trust models and reputation systems could be used to identify trustworthy TSAs. However, none of the existing systems are designed to run in the long term and to provide trust scores for TSAs.

In this work, we proposed a reputation-based trust system for long-term archiving called *Long-term evaluation of Trust* (LoT). It provides trust scores for TSAs and their time-stamps. These scores are derived from experiences collected from other participants of the system. We demonstrated the applicability of LoT in the use case of electronic health records (EHRs). We described how physicians could use LoT to avoid forged EHRs that, for example, could mislead physicians into prescribing wrong treatments. We presented a demonstration software for physicians and simulated how LoT can reduce the probability of malicious time-stamp sequences generated to forge electronic documents.

This is the first work that shows how to apply reputation systems and trust models to time-stamping-based long-term archiving schemes. Our solution allows to extend the standardized schemes by trust evaluations of archived documents, increasing the chance of detecting maliciously modified or generated documents. This is an important contribution for the practicability of long-term storage, since detecting such misbehavior increases the overall security of archiving systems. Therefore, this improvement allows archiving systems to be used also for the use cases where very sensitive data is processed, such as electronic health records.

Future work. We plan to analyze how to adapt LoT to archiving schemes where multiple TSAs sign a time-stamp together. Moreover, further security analysis on the storage system (e.g., with respect to non-repudiation) is desired. Moreover, we are working on efficient approaches to provide confidentiality together with authenticity for archived documents. Furthermore, it would be interesting to analyze colluding attacks [19][20], e.g., Ballot stuffing, Bad-mouthing, Self-promoting, Slandering, and Sybil attacks, against our proposed reputation-based trust system, LoT. In these types of attacks, attackers can influence the trust scores by sending fake positive or negative experiences to the participants of the system. Therefore, we plan to analyze the attack-resistant trust methods, e.g., similar to [21], in the context of LoT.

Acknowledgments

This work has been co-funded by the DFG as part of projects “Scalable Trust Infrastructures” and “Long-Term Secure Archiving” within the CRC 1119 CROSSING. In addition, it has received funding from the European Union’s Horizon 2020 research and innovation program under Grant Agreement No 644962.

References

[1] H. M. Gladney, Preserving digital information. Springer, 2007.

[2] C. Jee, “Nhs promises real-time digital health and care records by 2020,” http://www.computerworlduk.com/news/data/nhs-promises-real-time-digital-health-care-records-by-2020-3585822/ [retrieved: June, 2016].

[3] A. J. Blazic, S. Saljic, and T. Gondrom, “Extensible markup language evidence record syntax (xmlers),” RFC 6283, Internet Engineering Task Force, Jul. 2011, https://tools.ietf.org/html/rfc6283 [retrieved: June, 2016].

[4] N. Leavitt, “Internet security under attack: The undermining of digital certificates,” IEEE Computer, vol. 44, no. 12, 2011, pp. 17–20.

[5] D. Demirel and J. Lancrenon, “How to securely prolong the computational bindingness of pedersen commitments,” IACR Cryptology ePrint Archive, vol. 2015, 2015, p. 584, http://eprint.iacr.org/2015/584 [retrieved: June, 2016].

[6] D. Lekkas and D. Gritzalis, “Long-term verifiability of the electronic healthcare records’ authenticity,” Journal of Medical Informatics, vol. 76, no. 5-6, 2007, pp. 442–448.

[7] M. Vigil, D. Cabarcas, J. Buchmann, and J. Huang, “Assessing trust in the long-term protection of documents,” in ISCC 2013, 2013, pp. 185–191.

 ISBN: 978-1-61208-493-0

[8] M. Vigil, J. Buchmann, D. Cabarcas, C. Weinert, and A. Wiesmaier, "Integrity, authenticity, non-repudiation, and proof of existence for long-term archiving: A survey," Computers & Security, vol. 50, no. 0, 2015, pp. 16–32.

[9] T. Grandison and M. Sloman, "A survey of trust in internet applications," IEEE Communications Surveys and Tutorials, vol. 3, no. 4, 2000, pp. 2–16.

[10] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," Decision Support Systems, vol. 43(2), 2007, pp. 618–644.

[11] S. M. Habib, S. Hauke, S. Ries, and M. Mühlhäuser, "Trust as a facilitator in cloud computing: a survey," Journal of Cloud Computing: Advances, Systems and Applications, vol. 1, no. 19, 2012, p. 19.

[12] J. Braun, F. Volk, J. Classen, J. Buchmann, and M. Mühlhäuser, "CA trust management for the web PKI," IOS Press: JCS, 2014, Jun. 2014.

[13] R. Canetti, L. Cheung, D. Kaynar, N. Lynch, and O. Pereira, "Modeling computational security in long-lived systems," in CONCUR 2008, 2008, pp. 114–130.

[14] S. Ries, "Extending bayesian trust models regarding context-dependence and user friendly representation," in Proceedings of the ACM SAC. New York, NY, USA: ACM, 2009, pp. 1294–1301.

[15] A. Jøsang, "A logic for uncertain probabilities," INT J UNCERTAIN FUZZ, vol. 9, no. 3, 2001, pp. 279–212.

[16] S. M. Habib, S. Ries, S. Hauke, and M. Mühlhäuser, "Fusion of opinions under uncertainty and conflict – application to trust assessment for cloud marketplaces," in 11th IEEE TrustCom 2012, June 2012, pp. 109 –118.

[17] S. Ries, S. Habib, M. Mühlhäuser, and V. Varadharajan, "Certainlogic: A logic for modeling trust and uncertainty," in TRUST, vol. 6740, 2011, pp. 254–261.

[18] W. T. L. Teacy, G. Chalkiadakis, A. Rogers, and N. R. Jennings, "Sequential decision making with untrustworthy service providers," in Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2, ser. AAMAS '08. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 755–762.

[19] K. Hoffman, D. Zage, and C. Nita-Rotaru, "A survey of attack and defense techniques for reputation systems," ACM Comput. Surv., vol. 42, no. 1, Dec. 2009, pp. 1:1–1:31.

[20] A. Jøsang and J. Golbeck, "Challenges for robust of trust and reputation systems," in Proceedings of the 5th International Workshop on Security and Trust Management (STM 2009), 2009.

[21] S. Ries and E. Aitenbichler, "Limiting sybil attacks on bayesian trust models in open soa environments," in Proceedings of the The First International Symposium on Cyber-Physical Intelligence (CPI-09), 2009.

 ISBN: 978-1-61208-493-0