LINCOS - A Storage System Providing Long-Term Integrity, Authenticity, and Confidentiality (Full Paper)[*]

Johannes Braun, Johannes Buchmann, Denise Demirel, Matthias Geihs
TU Darmstadt, Germany

Mikio Fujiwara, Shiho Moriai, Masahide Sasaki, Atsushi Waseda
NICT, Japan

ABSTRACT

The amount of digital data that requires long-term protection of integrity, authenticity, and confidentiality grows rapidly. Examples include electronic health records, genome data, and tax data. In this paper we present the secure storage system LINCOS, which provides protection of integrity, authenticity, and confidentiality in the long-term (e.g., decades, centuries). It is the first such system. It uses the long-term integrity scheme COPRIS, which is also presented here and is the first such scheme that does not leak any information about the protected data. COPRIS uses information-theoretic hiding commitments for confidentiality-preserving integrity and authenticity protection. LINCOS uses proactive secret sharing for confidential storage of secret data. We also present implementations of COPRIS and LINCOS. A special feature of our LINCOS implementation is the use of quantum key distribution and one-time pad encryption for information-theoretic private channels within the proactive secret sharing protocol. The technological platform for this is the Tokyo QKD Network, which is one of worlds most advanced networks of its kind. Our experimental evaluation establishes the feasibility of LINCOS and shows that in view of the expected progress in quantum communication technology, LINCOS is a promising solution for protecting very sensitive data in the cloud.

1. INTRODUCTION

1.1 Motivation and problem statement

Today large amounts of data are digitally stored, increasingly in cloud-based data centers, and this amount will massively grow in the future. For example, Japanese hospitals use redundant cloud storage to protect sensitive medical data from loss due to natural catastrophes [23, 24]. Also, in his state of the union address 2015, the U.S. President Barack Obama announced a Precision Medicine Initiative which will require to digitally store the health data of virtually all U.S. citizens.

Protection requirements. Digitally stored data require protection throughout their whole lifetime which may be very long. Important protection goals are *integrity*, *authenticity*, and *confidentiality*. Integrity means that illegitimate and accidental changes of the data can be discovered. Authenticity refers to the origin of the data being identifiable. Confidentiality guarantees that only authorized parties are able to access the data. For example, consider medical data. Their integrity is extremely important because changes may lead to incorrect treatment with serious health consequences. Authenticity is required for liability reasons and confidentiality protects the privacy of the involved individuals. Medical data may have to be kept as long as the respective patients are alive or even beyond this time. So the required protection period may be more than 100 years. Other examples for sensitive long-lived data are genome data, governmental secrets, and tax data.

Current cryptography is unsuitable. Unfortunately, current technology does not provide integrity, authenticity, and confidentiality protection over such a long time. The cryptographic algorithms used today for such protection, such as AES encryption and RSA signatures, fail to provide sufficient security guarantees. They are *complexity-based* which means that their security relies on the intractability of certain algorithmic problems, e.g., integer factorization. However, cryptanalytic power is steadily increasing. According to Moore's law, the computing speed doubles every 18 months. Also, there is algorithmic progress. Hence, keys chosen today will be too short in the future. For example, in their original RSA paper [32], the authors estimate the required RSA modulus size: "using 200 digits provides a margin of safety against future developments." However less than 30 years later factoring 200 decimal digit numbers became feasible [3]. This situation is very critical. Adversaries may store encrypted data now and decrypt them later when the encryption algorithm becomes broken which may happen during the lifetime of the protected data. Technologically, this appears to be quite feasible. For instance, the Utah Data Center of the NSA has an estimated capacity of 4 to 12 Exabytes (10^{18} bytes) which allows to store huge amounts of encrypted data for a long time.

The question arises whether it is possible and feasible to provide long-term protection of integrity, authenticity, and confidentiality of digital data over decades or even centuries.

There exist several partial solutions to this problem. For overviews of confidentiality and integrity/authenticity related solutions see [6] and [38], respectively. These surveys also contain the relevant references.

Confidentiality of data in transit. In 1949 Claude Shannon presented his model of *information-theoretic confidentiality* protection and proved that *one-time-pad encryption*

[*]This is the full version of a paper published at the 13th ACM Asia Conference on Computer and Communications Security (ASIACCS 2017).

(OTP) provides such protection for transmitted data (*data in transit*). However, OTP keys are as long as the protected data, can only be used once, and are required to be exchanged in an information-theoretically secure fashion. Therefore, OTP encryption has been only used for special applications such as military applications with key exchange by trusted couriers. In the past decades, other methods of such key exchange have been developed, including schemes based on the *bounded storage*, *noisy channel*, or *limited access* models, and *quantum key distribution (QKD).* Among these options, QKD is by far the most advanced, both theoretically and experimentally. For example, many countries such as Austria, China, Japan, Switzerland, and the USA are currently deploying QKD-protected backbones. For example, they use QKD to protect keys for complexity-based symmetric encryption. However, in this case no information-theoretic security is achieved.

Confidentiality of data at rest. Unfortunately, OTP encryption is unsuitable for stored data. This is because OTP requires using and protecting one-time keys that are as long as the original data. Hence, nothing is gained by using OTP. Instead, *proactive secret sharing* can be used to provide information-theoretic confidentiality protection of stored data. Proactive secret sharing decomposes the secret into n shares in such a way that a threshold number $k \leq n$ of shares is required to reconstruct the secret while any smaller number of shares reveals no information about the secret. The shares are renewed on a regular basis in order to prevent attacks of *mobile adversaries* who may be able to learn more and more shares over time. Such solutions are well suited for cloud storage systems and are already used in this context [27]. However, as in currently used secret sharing solutions communication protection is only complexity-based, they do not provide information-theoretic confidentiality.

Integrity and authenticity. There are standardized solutions for long-term integrity and authenticity protection (see [18]) which are already used in practice. They utilize timestamp chains to prolong the validity of complexity-based digital signatures thereby protecting integrity and authenticity for any length of time. However, these solutions prohibit long-term confidentiality protection. This is because they submit hashes of the protected data to timestamp authorities. As cryptographic hash functions only offer complexity-based security, they may leak information over time. This is also why the solutions in [22] and [31] do not support long-term confidentiality protection.

In summary, the problem of long-term protection of integrity, authenticity, and confidentiality of digital data is urgent and a comprehensive solution that provides such protection is not known so far.

1.2 Contribution

In this paper we present the first storage solution that can simultaneously protect integrity, authenticity, and confidentiality of digital data potentially over decades and centuries. We analyze its security and experimentally study its feasibility. As our solution uses a distributed storage system, it is suitable for cloud applications.

Confidentiality-preserving long-term integrity protection. Our first contribution is the new scheme COPRIS. It is the first long-term integrity scheme that is confidentiality preserving, i.e., it does not leak any information about the protected data to third party services. It also provides authenticity protection if the protected data is signed and the signature is protected together with the data. COPRIS is inspired by existing long-term integrity schemes, e.g., ERS [18]. The idea in COPRIS is to no longer timestamp the protected documents. Instead, information-theoretically hiding commitments to these documents are timestamped. These commitments never leak any information about the documents. Information-theoretically hiding commitments can only be computationally binding [5]. Therefore, commitments are renewed on a regular basis. Timestamps are also renewed. In this way, we obtain an *evidence record*: a sequence of timestamps of commitments. In order to verify the integrity of a document, this document and the secret decommitment values are revealed. Then the evidence record is used to recursively prove the existence of the commitments at their respective generation times. Also, the commitments are recursively opened proving eventually, that the first commitment opens to the data and therefore, these data existed at the asserted time.

Long-term integrity, authenticity, and confidentiality protection. Our second contribution is the secure storage system LINCOS. It is the first storage system that simultaneously protects integrity, authenticity, and confidentiality of stored data in the long-term. We present a security analysis and report on our implementation and thorough experimental evaluation of LINCOS.

In LINCOS, a document owner communicates with an *integrity system* and a *confidentiality system.* The integrity system is based on COPRIS which we implemented as a Java application. For any document, it maintains an evidence record that can be used to verify document integrity. Document authenticity can be enabled by an initial digital signature on the document. The confidentiality system protects the document and the decommitment values. It uses proactive secret sharing to protect the confidentiality of the data at rest. In our experiments we use four shareholders and a threshold of three. Our implementation is based on Shamir's secret sharing and we make it proactive by requiring the document owner to renew the shares. In order to achieve information-theoretic confidentiality, private channels are used. Our implementation realizes them via QKD and OTP. We choose QKD because compared to the other candidates it is the theoretically most clear cut and practically most promising solution for information-theoretic key exchange. We use the Tokyo QKD Network to implement these channels. This network is one of worlds most advanced QKD networks and allows for a reliable feasibility study. It provides private channels between the four shareholders and the document owner.

We report on an experiment that simulates protecting documents of different sizes for 100 years. The size of the evidence records generated over this time period is at most a few hundred kilobytes and is independent of the size of the protected document. Also, the time for generating and verifying integrity and authenticity is almost independent of the document size and grows from a few milliseconds to at most 10 seconds. This shows that the performance of the integrity system is very acceptable, in particular as computing speed is expected to increase considerably over the next

Instance	RSA-1024	RSA-2048	RSA-4096
Security until	2006	2040	2085

Table 1: Instances of the RSA signature scheme with estimated security time frame according to Lenstra [25].

100 years. As for confidentiality protection, the limiting factor turns out to be the speed of QKD key generation. The average key supply that we currently achieve is 40 kb/s. So transmitting 1 GB of data requires 2.3 days of prior key accumulation. This allows for proactive secret sharing of 158 GB with a share renewal period of 2 years. However, in the near future key rates of 1 Mb/s can be expected which will reduce the time for distributing a 1 GB key to 2.2 hours. Thus it will be possible to protect 4 TB with a share renewal period of 2 years. For example, 4 TB is the size of the genomes of roughly 5000 persons.

LINCOS is well suited for long-term storage systems. Availability requirements in such systems, in particular in case of natural or other catastrophes, suggest to redundantly store the data in multiple locations which are far apart from each other. In fact, redundant storage is already common practice in many scenarios (e.g., [23, 24]). LINCOS can be used in these scenarios and additionally achieves long-term integrity, authenticity, and confidentiality protection.

1.3 Organization

Our paper is organized as follows. Section 2 introduces the cryptographic components that are used in COPRIS and LINCOS and also discusses their security. Section 3 presents the long-term integrity and authenticity scheme COPRIS and its security analysis. Section 4 describes the comprehensive solution LINCOS for simultaneous integrity, authenticity, and confidentiality protection and also includes the security analysis of LINCOS. Our implementation of COPRIS and LINCOS is described in Section 5. The experimental results are discussed in Section 6. Finally, in Section 7 we draw conclusions and sketch future work.

2. CRYPTOGRAPHIC COMPONENTS

In this section we give an overview of the cryptographic components that are used in COPRIS and LINCOS. Such a cryptographic component may provide computational security or information-theoretic security. Some components, such as commitment schemes, may even have both properties for different functionalities. Computational security is based on the intractability of a computational problem, e.g., integer factorization. Information-theoretic security is based on the principles of information theory and does not require such an assumption (see [36]).

A computationally secure component is usually parametrized with a security parameter which determines the hardness of the underlying computational problem. The security parameter is chosen such that the cryptographic component remains secure for the intended usage period. Lenstra [25] presents heuristics for choosing security parameters appropriately. A variety of parameter suggestions tables can be found at [1]. As an example, Table 1 shows security estimates for the RSA signature scheme.

Timestamps. Timestamps are issued by timestamp services using *timestamp schemes* [20, 2]. A timestamp scheme involves a protocol Stamp and an algorithm Verify. In the protocol Stamp a client sends the hash $h(d)$ of a document d, which may be any data object, to a timestamp service and obtains a timestamp $T = (t, \mathsf{s}, \mathsf{V})$. Here t is the time when the timestamp was issued, s is a signature by the timestamp service on $(t, h(d))$, and V is a certificate chain required to verify s. Such a timestamp can be verified using algorithm Verify. Input to Verify is a trust anchor, the document d, the timestamp T, a timestamp time t, and a validation reference time t_{ref}. Typically, the trust anchor is the public key of the certificate authority that issued the first certificate in V. The algorithm verifies the signature s for $(t, h(d))$ using the trust anchor and the certificate chain V. It outputs 1 if the signature is valid relative to time t_{ref} and 0 otherwise. Output 1 means that the document d existed at time t. More details regarding timestamp verification can be found in [2]. The security notion for signature-based timestamp schemes is *unforgeability* which is defined analogous to signature unforgeability [17, 14].

Authenticated channels. An *authenticated channel* is a mutually authenticated connection between a sender and a receiver. In protocol Setup, the sender and receiver agree on channel parameters, for example session keys. After executing Setup, the sender uses protocol Send to send data d to the receiver. We require an authenticated channel to guarantee computationally secure mutual authentication of the sender and the receiver. For details see [4, 12].

Private channels. In addition to authenticated channels we also use *private channels*. Like authenticated channels, private channels also support the protocols Setup and Send. However, the security guarantees are stronger. In addition to computationally secure mutual authentication, private channels also provide information-theoretic confidentiality of the transmitted data [37, 36].

Commitment schemes. A *commitment scheme* allows a party to commit to some document without revealing it. Such a scheme consists of the algorithms Commit and Verify. Input to Commit is a document. Output is a pair (c, r) where c is a *commitment value* and a r is a *decommitment value*. Input to Verify is a trust anchor, a document, a commitment value, a decommitment value, and a validation reference time t_{ref}. The trust anchor is used to verify that the commitment scheme was correctly instantiated by some trusted authority. Output is 1 or 0, where 1 means that the commitment is valid relative to time t_{ref}. We require commitment schemes to be *computationally binding* and *information-theoretically hiding*. For details see [30, 16].

Proactive secret sharing. The participants of a *proactive secret sharing scheme* are a *dealer*, several *shareholders*, and a *retriever*. Such schemes allow information-theoretic confidentiality protection of data at rest and are suitable for secure cloud storage as they involve a set of data storage servers. Proactive secret sharing schemes support the protocols Setup, Share, Reshare, and Retrieve. In the Setup protocol the dealer and an initial set of shareholders agree on sharing parameters, e.g., the size of the shares. In the Share protocol, the dealer generates one share for each shareholder and sends it to the respective shareholder through a private channel. The Reshare protocol involves a current set S and a new set S' of shareholders. Initially, the shareholders in S have their individual shares. After the protocol has terminated, all the shareholders in S have deleted their shares and the shareholders in S' have new shares. Finally, the Retrieve protocol involves a retriever and a set of shareholders. The retriever obtains one share from each shareholder through a private channel. From these shares, he reconstructs the initial data of the dealer. One way of implementing protocol Reshare is to let the dealer reconstruct the secret document using protocol Retrieve and then distribute new shares to the potentially new shareholders using protocol Share. An alternative is to use resharing protocols that do not reconstruct the secret data, e.g., [21, 11, 40, 19].

We require proactive secret sharing schemes to provide information-theoretic confidentiality in the *mobile adversary model* [21]. In this model, adversaries may learn some shares over time. However, before the number of learned shares exceeds the selected threshold k, the shares are renewed via Reshare. Also, up to k shares provide no information about the shared secret.

Another useful security property of secret sharing schemes is *verifiability* [10, 30]. It prevents the dealer from distributing inconsistent shares. However, this property is not essential in our context and may be added as a further property of our storage solution later.

3. COPRIS: CONFIDENTIALITY-PRESERVING LONG-TERM INTEGRITY SCHEME

In this section we present the scheme COPRIS which ensures long-term integrity protection and is long-term confidentiality preserving, i.e., it does not leak any information about the protected data. The scheme also provides long-term authenticity if the protected data is signed and the integrity of the signature is protected together with the signed data.

3.1 Scheme

COPRIS works as follows. A document owner stores a document d at some time t. He keeps d secret and uses COPRIS to construct a *proof of integrity* PI *for d. Later he may choose to reveal d to another party. This party then uses* PI *to verify that d existed at time t.* To preserve the confidentiality of d, the proof of integrity PI is constructed in such a way that no information about d is revealed to any third party. Confidential storage of the secret data is out of the scope of COPRIS and is dealt with in LINCOS (Section 4).

We now explain the construction of the proof of integrity and its verification and we show that this construction has the desired security properties. The integrity proof is a pair (E, R), where E is an *evidence record* and R is a list of decommitment values. The evidence record is constructed interactively between the document owner and an *evidence service* which, in turn, interacts with a timestamp service. The list of decommitment values is constructed and kept secret by the document owner. The document owner may decide to reveal the decommitment values together with the document to a third party *verifier.*

In the following we describe COPRIS in detail. Figure 1 illustrates the functionality of COPRIS. Figure 2 lists the algorithms used in COPRIS.

Initial protection. The initial integrity proof is constructed as follows. The document owner runs algorithm Protect. Input is the document d and the (initially empty) list of decommitment values R. He selects a commitment scheme CS and computes a commitment $(c, r) \leftarrow \mathsf{CS.Commit}(d)$. He sets $R = (r)$ and sends the commitment value c to the evidence service. When the evidence service receives c, it runs algorithm AddEv. Input is c and the (initially empty) evidence record E. It requests a timestamp T on c from a timestamp service TS using protocol TS.Stamp at time t. The evidence record is initialized as $E = (c, T, t)$.

Timestamp renewal. Before the last timestamp becomes insecure it must be renewed. In this case, the evidence service executes algorithm RenewTs, where the input is the current evidence record E. It selects a new timestamp scheme TS and obtains a timestamp T on E at time t using protocol TS.Stamp. Then, it appends $(\perp, T, t)$ to E, where $\perp$ indicates that there is no commitment generated at timestamp renewal.

Commitment renewal. Before the last commitment created by the document owner becomes insecure, it must be renewed. The document owner runs the algorithm RenewCom. Input is the document d and the decommitment value list R. The document owner selects a new commitment scheme CS and computes $(c, r) \leftarrow \mathsf{CS.Commit}(d, R)$. He add r to the list R at position $|E|$ and sends c to the evidence service. When the evidence service receives c, it runs algorithm AddEv. Input is c and the evidence record E.

Verification. When the document owner reveals d to the verifier, he also transmits the asserted existence time t and the integrity proof (E, R). Using this information, the verifier can validate the existence of d at time t as follows. Let $R = (r_0, \ldots, r_n)$ and $E = (c_0, T_0, t_0, \ldots, c_n, T_n, t_n)$.

We describe the verification procedure. We define t_{n+1} to be the time of verification and for $i \in \{0, \ldots, n\}$ we set $E_i = (c_0, T_0, t_0, \ldots, c_i, T_i, t_i)$ and $R_i = (r_0, \ldots, r_i)$. Furthermore, for $i \in \{0, \ldots, n\}$ let TS_i the timestamp scheme associated with T_i and if $c_i \neq \perp$, let CS_i denote the commitment scheme associated with c_i. Also, let $t_{\mathrm{NRC}(i)}$ denote the time of the next recommitment after c_i, i.e., the minimum $t_{i'}$ with $i' > i$ and $c_{i'} \neq \perp$ or t_{n+1} if $i = n$. The verifier uses his trust anchor TA, which must contain the necessary certificates for the public parameters of the signatures and commitments, and for $i \in \{0, \ldots, n\}$ verifies that

if $c_i \neq \perp$:

$$\mathsf{CS}_i.\mathsf{Verify}(TA, (d, R_{i-1}), c_i, r_i; t_{\mathrm{NRC}(i)}) = 1$$

$$\mathsf{TS}_i.\mathsf{Verify}(TA, (E_{i-1}, c_i), T_i, t_i; t_{i+1}) = 1$$

else:

$$\mathsf{TS}_i.\mathsf{Verify}(TA, E_{i-1}, T_i, t_i; t_{i+1}) = 1.$$

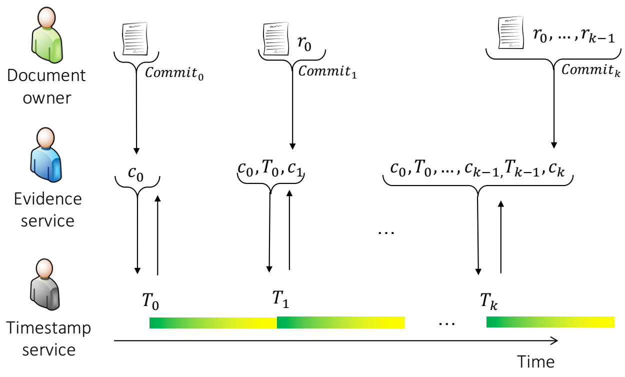

Figure 1: Functionality of COPRIS.

$\mathsf{Protect}(d, R)$:

1. Select commitment scheme CS
2. $(c, r) \leftarrow \mathsf{CS.Commit}(d)$
3. Send c to evidence service, set $R = (r)$

$\mathsf{AddEv}(c, E)$:

1. Select timestamp scheme TS
2. $(T, t) \leftarrow \mathsf{TS.Stamp}(E, c)$
3. Append (c, T, t) to E

$\mathsf{RenewTs}(E)$:

1. Select timestamp scheme TS
2. $(T, t) \leftarrow \mathsf{TS.Stamp}(E)$
3. Append $(\perp, T, t)$ to E

$\mathsf{RenewCom}(d, R)$:

1. Select commitment scheme CS
2. $(c, r) \leftarrow \mathsf{CS.Commit}(d, R)$
3. Send c to evidence service, append r to R

$\mathsf{Verify}(TA, d, t, R, E)$:

1. Let $R = (r_0, \ldots, r_n)$ and $E = (c_0, T_0, t_0, \ldots, c_n, T_n, t_n)$ such that r_i corresponds to c_i. For $i \in \{0, \ldots, n\}$, let $E_i = (c_0, T_0, t_0, \ldots, c_i, T_i, t_i)$ and $R_i = (r_0, \ldots, r_i)$, let TS_i be the timestamp scheme associated with T_i, and if $c_i \neq \perp$, let CS_i be the commitment scheme associated with c_i. Furthermore let $t_0 = t$ and t_{n+1} be the current time.
2. For $i \in \{0, \ldots, n\}$, if $c_i = \perp$, verify that $\mathsf{TS}_i.\mathsf{Verify}(TA, E_{i-1}, T_i, t_i; t_{i+1}) = 1$, and if $c_i \neq \perp$, verify that $\mathsf{CS}_i.\mathsf{Verify}(TA, (d, R_{i-1}), c_i, r_i; t_{\mathrm{NRC}(i)}) = 1$ and $\mathsf{TS}_i.\mathsf{Verify}(TA, (E_{i-1}, c_i), T_i, t_i; t_{i+1}) = 1$.

Figure 2: Algorithms used in COPRIS for initial protection, adding evidence, timestamp renewal, commitment renewal, and verification.

3.2 Security

In the following, we provide a security analysis of COPRIS. We show that COPRIS provides long-term integrity and authenticity protection and that no confidential data is leaked to the evidence and timestamp service.

We consider adversaries that may be active for an unbounded period of time while being computationally bounded per unit of time. We refer to [15, 8, 7] for more details regarding this adversary model. It reflects the indefinite lifetime of long-lived systems and of the data processed by them. The fact that adversaries have limited capabilities per unit of real time allows for the usage of computationally secure cryptographic primitives in long-lived systems.

Our security analysis is based on the results of [15, 8, 7], where it is shown that (under certain computational assumptions) extractable commitments and timestamps can be used to argue about the knowledge of an adversary at an earlier point in time based on its current knowledge and the history of interactions. Here, we use the following notation to argue about the knowledge of an adversary $\mathcal{A}$. For a data object d and a time t, we write $d \in \mathcal{K}_{\mathcal{A}}[t]$ to denote that $\mathcal{A}$ knew d at time t. We remark that adversaries do not forget knowledge once they obtained it. That is, for any data object d and any two points in time t and t', it holds that if $d \in \mathcal{K}_{\mathcal{A}}[t]$ and $t' > t$, then also $d \in \mathcal{K}_{\mathcal{A}}[t']$. We also use the convention that for verification of timestamps and commitments a trust anchor TA is provided by a PKI that certifies the verification keys of the used timestamp and commitment scheme instances and specifies the corresponding instance validity periods.

We state two lemmas that we will use for proving long-term unforgeability of COPRIS. These lemmas resemble results of [15, 8, 7] about extractable timestamps and commitments. The first lemma states that if an adversary $\mathcal{A}$ knows a timestamp (T, t) and a data object d at a time t', and (T, t) is valid for d at t', then $\mathcal{A}$ has already known d at time t (with high probability). The second lemma states that if an adversary $\mathcal{A}$ knows a commitment value c at a time t, a message d and a decommitment r are known at time $t' > t$, and r is a valid decommitment from c to d at time t', then $\mathcal{A}$ has already known the message d at the commitment time t (with high probability).

Lemma 1. *For any data object d, timestamp (T, t), and time t':*

$$(d, (T, t)) \in \mathcal{K}_{\mathcal{A}}[t'] \land \mathsf{VerTs}_{TA}(d, T, t; t') = 1 \implies d \in \mathcal{K}_{\mathcal{A}}[t] \ .$$

Lemma 2. *For any commitment value c, time t, data object d, decommitment value r, and time $t' > t$:*

$$c \in \mathcal{K}_{\mathcal{A}}[t] \land (d, r) \in \mathcal{K}_{\mathcal{A}}[t'] \land \mathsf{VerCom}_{TA}(d, c, r; t') = 1 \\ \implies d \in \mathcal{K}_{\mathcal{A}}[t] \ .$$

By long-term unforgeability of COPRIS we mean that for any adversary $\mathcal{A}$ as described above it is feasible to construct a valid integrity proof for a document d and a time t if $\mathcal{A}$ did not know d at time t.

It is essential for the security of COPRIS that the following assumptions hold.

I1. The commitment schemes used in the proof of integrity are computationally binding in their usage period.

I2. The timestamp schemes used in the proof of integrity are computationally unforgeable in their usage period.

I3. The verifier has a valid trust anchor.

Here, by *usage period* of the cryptographic schemes we mean the time interval that begins when the cryptographic scheme is chosen and ends when it is replaced by a new scheme. Also, by a valid trust anchor we mean a trust anchor that allows for the verification of all timestamps and commitments.

THEOREM 3.1. *Under Assumptions I1, I2, and I3,* COPRIS *is long-term unforgeable.*

PROOF. Assume that an adversary $\mathcal{A}$ outputs (d, t, E, R) such that (E, R) is a valid integrity proof for d and time t. We show that in this case $\mathcal{A}$ knew d at time t (with high probability).

We write $R = (r_0, \ldots, r_n)$ and $E = (c_0, T_0, t_0, \ldots, c_n, T_n, t_n)$. For $i \in \{0, \ldots, n\}$ let $E_i = (c_0, T_0, t_0, \ldots, c_i, T_i, t_i)$ and $R_i = (r_0, \ldots, r_i)$. Furthermore, we denote by CS_i the commitment scheme corresponding to c_i, for $c_i \neq \bot$, and by TS_i the timestamp scheme corresponding to T_i. Also, denote by t_{n+1} the time at which the adversary outputs (d, t, E, R). Finally, let $t_{\mathrm{NRC}(i)}$ denote the time of the next recommitment after c_i or t_{n+1} if $i = n$.

We show that for $i \in \{n, \ldots, 0\}$ the following holds:

- $\mathcal{A}$ knew (c_i, T_i, t_i) at time t_{i+1}.
- If $c_i \neq \bot$, then $\mathcal{A}$ knew (d, E_i, R_i) at time $t_{\mathrm{NRC}(i)}$.

We prove the first two statements recursively. We observe that for $i = n$ both statements are obviously true as $\mathcal{A}$ outputs (d, t, E, R) at time $t_{n+1} = t_{\mathrm{NRC}(n)}$ and $E = E_n$ contains (c_n, T_n, t_n) and $R = R_n$. Next, we prove that if the statements are true for i, then they must also be true for $i-1$. We observe that if $c_i = \bot$, then by the validity of the integrity proof we have $\mathsf{TS}_i.\mathsf{Verify}(TA, E_{i-1}, T_i, t_i; t_{i+1}) = 1$ and by assumptions I2 and I3 and Lemma 1 we have that E_{i-1} was known at time t_i, which implies that $(c_{i-1}, T_{i-1}, t_{i-1})$ was also known at time t_i. Furthermore, we observe that if $c_i \neq \bot$, then we have $\mathsf{CS}_i.\mathsf{Verify}(TA, (d, R_{i-1}), c_i, r_i; t_{\mathrm{NRC}(i)}) = 1$ and $\mathsf{TS}_i.\mathsf{Verify}(TA, (E_{i-1}, c_i), T_i, t_i; t_{i+1}) = 1$. By assumptions I2 and I3 and Lemma 1 it follows that (E_{i-1}, c_i) was known at time t_i, which implies that $(c_{i-1}, T_{i-1}, t_{i-1})$ was also known at time t_i, and by assumptions I1 and I3 and Lemma 2 it follows that (d, R_{i-1}) was known at time $t_i = t_{\mathrm{NRC}(i-1)}$. We observe that, in particular for $i = 0$, this means that $\mathcal{A}$ knew d at time $t_1 = t$ (with high probability). □

We remark that the concrete security level of the protection degrades slowly over time based on the chosen schemes and parameters. For more details, we refer to [15, 8, 7].

Next, we show that COPRIS is confidentiality preserving in the long-term, i.e., no information is leaked to the evidence and timestamp service (in an information-theoretic sense). This fact relies on the following assumption.

C1. The commitment schemes are information-theoretically hiding.

THEOREM 3.2. *Under assumption C1,* COPRIS *is information-theoretic confidentiality preserving.*

PROOF. The only data that is sent by the document owner to the evidence service and from there to the timestamp service are information-theoretically hiding commitments. By assumption C1, these commitments do not leak any information about the committed data. □

4. LINCOS: LONG-TERM INTEGRITY, AUTHENTICITY, AND CONFIDENTIALITY SYSTEM

In this section we describe our new long-term storage system LINCOS which allows for information-theoretic confidentiality and long-term integrity and authenticity protection. The situation is similar as in COPRIS. A document owner stores a document d at time t. He uses an *integrity system*, which is based on COPRIS, to construct an integrity proof PI. Additionally, he uses a *confidentiality system* for information-theoretic confidential storage of the secret document. The confidentiality system is also used for confidential storage of the secret decommitment values, which are generated during integrity proof construction.

4.1 System specification

An overview of LINCOS is shown in Figure 3. The involved parties are the *document owner*, the *evidence service*, a *timestamp service*, a set of *shareholders*, and a *verifier*. These parties are connected by private or authenticated channels as shown in Figure 3. While LINCOS is running, the respective channels are instantiated securely whenever a connection needs to be established. When referring to the evidence service, we use the same notation as in the description of COPRIS. That is, the document owner maintains a list of decommitment values R and the evidence service maintains an evidence record E. Both are initially empty.

Initial document protection. For initial protection of a document d, the document owner runs COPRIS.Protect. Input is a document d and the (initially empty) list of decommitment values R. The document owner chooses a confidentiality system involving several shareholders. The document owner uses protocol Share to distribute (d, R) among the shareholders.

Renewal of timestamps. In COPRIS, timestamps are renewed on a regular basis. For this, the evidence service uses COPRIS.RenewTs.

Renewal of commitments. In COPRIS also the commitments are renewed regularly. For this, the document owner does the following. First, he retrieves d and the sequence of decommitment values R from the confidentiality system by running protocol Retrieve. Then, he runs the algorithm COPRIS.RenewCom, thereby updating the list of decommitment values R and the evidence record E. Finally, the document owner selects a potentially new confidentiality system and runs protocol Share to distribute the document d and the updated sequence of decommitment values R among the shareholders in the confidentiality system.

Renewal of secret shares. The shares stored by the shareholders are renewed on a regular basis. This prevents a mobile adversary to take advantage of shares he may have been able to obtain in the past. In this process, the current set of shareholders of the confidentiality system may also be replaced by a new set of shareholders operated by the same confidentiality system. This resharing is done by running protocol Reshare.

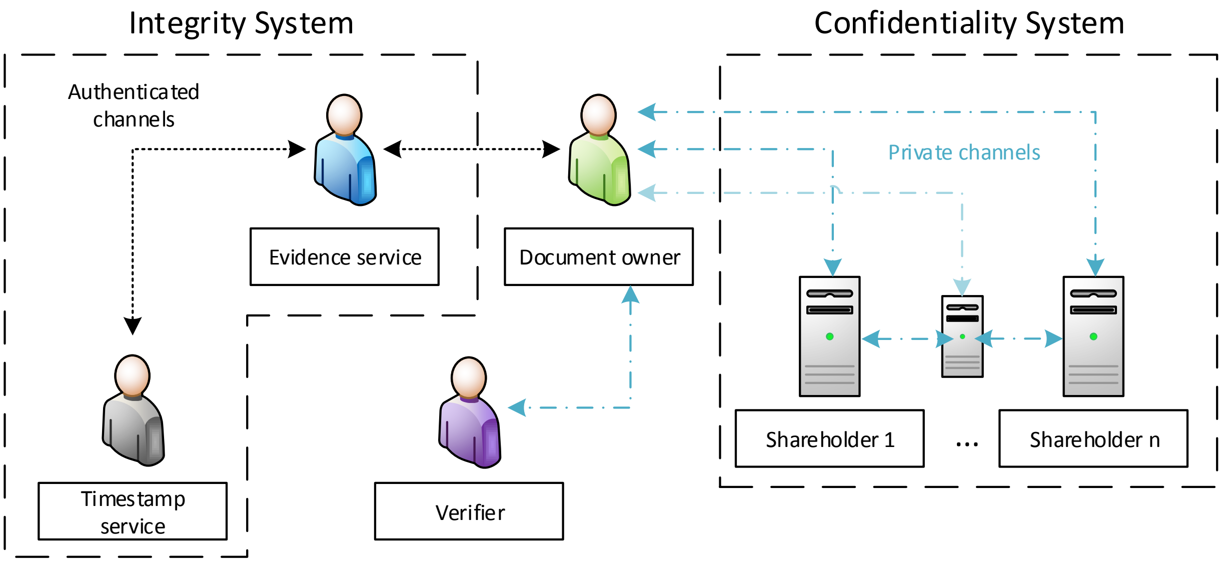

Figure 3: Overview of the long-term secure storage system LINCOS.

Verification. When the document owner decides to reveal the document d to a verifier and prove that it existed at time t, he executes the following steps. He requests the current evidence record E from the evidence service. He also retrieves the document d and the list of decommitment values R from the confidentiality system by running the protocol Retrieve. He sends the document d, time t, evidence record E, and the list of decommitment values R to the verifier through a private channel. The verifier uses the data that he received and his trust anchor TA and checks that COPRIS.Verify$(TA, d, t, E, R) = 1$. This proves that d existed at time t and has not been changed.

4.2 Security

We show that under appropriate assumptions, LINCOS provides integrity protection for an indefinite period of time and information-theoretic confidentiality protection.

Adversaries are assumed to have the capabilities described in Section 3.2. They run forever but are computationally bounded per unit of time. In addition, to analyze confidentiality, adversaries are assumed to be active and mobile. This means that adversaries may eavesdrop on channels or corrupt shareholders. A more detailed discussion of this model can be found in [29, 9, 21].

Integrity. Theorem 3.1 states that in this adversary model, LINCOS provides long-term integrity and authenticity protection if Assumptions I1, I2, and I3 from Section 3.2 are satisfied.

Confidentiality. We say that LINCOS provides information-theoretic confidentiality protection if an adversary with capabilities as described above cannot recover any information about the stored document in an information-theoretic sense.

For information-theoretic confidentiality we require Assumption C1 from Section 3.2 and the following assumptions to hold.

C2. The private channels used in LINCOS provide information-theoretic confidentiality and computational authenticity at the time of data transmission.

C3. The proactive secret sharing schemes used in LINCOS provide information-theoretic confidentiality.

C4. During their usage periods, the secret sharing services used in LINCOS prevent mobile adversaries from learning k or more shares.

THEOREM 4.1. *Under assumptions C1, C2, C3, and C4 the system* LINCOS *provides information-theoretic confidentiality protection.*

PROOF. LINCOS is based on COPRIS, which is information-theoretic confidentiality preserving under Assumption C1. Hence, an adversary cannot obtain any information about the confidential document by eavesdropping on the authenticated channel from the document owner to the evidence service or from the evidence service to the timestamp service.

Information-theoretic confidentiality of data sent through the private channels from the document owner to the shareholders or between the shareholders is guaranteed by Assumption C2. Information-theoretic confidentiality of data stored at the shareholders is guaranteed by Assumptions C3 and C4. □

5. IMPLEMENTATION

In this section we describe our implementation of the storage system LINCOS, which we presented in Section 4.

LINCOS uses COPRIS for its integrity system and proactive secret sharing combined with appropriate private channels for its confidentiality system. We describe the implementation of these two systems. One important feature of our implementation is the possibility of replacing cryptographic components. This is required because of Assumptions I1 and I2. Another feature is the realization of private channels using the Tokyo QKD Network [33].

5.1 Implementation of COPRIS

The parties involved in COPRIS are the document owner, the evidence service, and a timestamp service. As cryptographic componentes they use commitment and timestamp schemes. We implemented COPRIS in Java following the specification given in Section 3. Here we describe the implementation of the components.

Commitment scheme. As the commitment scheme, which is used by the document owner to generate commitments to documents, we use the scheme proposed by Pedersen [30]. It is computationally binding and information-theoretically hiding (Assumptions I1 and C1) and is parametrized by two prime numbers p and q. The binary length of q determines the size of the data that can be committed to. Bindingness of Pedersen's commitment scheme is based on the discrete logarithm problem in a subgroup of order q of the multiplicative group of the finite field of order p.

To allow committing to data of arbitrary length, data are first hashed, using a cryptographic hash function, and then committed to. Our implementation supports the SHA-2 hash function family which contains the hash functions SHA-224, SHA-256, SHA-384, and SHA-512. They have increasing security levels.

The hash function and the parameters of the commitment scheme are chosen such that computational bindingness is achieved during the intended usage period as required by Assumption I1. In practice, these choices can be made on the basis of trustworthy recommendations. For an overview of recommendations see [1].

Timestamp scheme. The timestamp service used by the evidence service is implemented in accordance with standard RFC 3161 [2]. Implementing it requires choosing a hash function and a digital signature scheme. We use the SHA-2 hash function family and the RSA digital signature scheme. The security of the used RSA instance depends on the bitlength of the RSA-modulus. Hash function and RSA-modulus are chosen such that they remain secure during their usage period as required by Assumption I2. Again, see [1] for an overview of recommendations on how to choose hash functions and security parameters in practice.

Authenticated channels. Authenticated channels are realized using TLS [12]. This protocol is state of the art and provides mutual authentication in a computational sense. Authenticated channels are important for the robustness of our system. They guarantee that the document owner connects with the intended evidence service. However, our security analysis does not require security properties of these channels. Therefore, we do not discuss the schemes and parameters chosen for TLS.

5.2 Secret sharing and private channels

In the following we describe the implementation of the confidentiality system in LINCOS which uses private channels and proactive secret sharing.

Private channels. LINCOS uses private channels to connect the document owner with the shareholders. By Assumption C2, these channels are required to provide information-theoretic confidentiality and computational authenticity. For establishing such private channels we use the Tokyo QKD Network [33], which is shown in Figure 4. A combination of Wegman-Carter authentication, QKD, and OTP encryption is used to achieve information-theoretic private and authenticated channels [26, 28, 34]. The network consists of three layers; the *quantum layer*, the *key management layer*, and the *application layer*. Secret sharing is run on the application layer. Parties on the application layer request and receive key material from the key management layer. The key management layer establishes an interface to the quantum layer where the raw key material is generated using QKD technology. To improve the capabilities of the network, keys are relayed on the key management layer by key management agents. In order to allow for Assumption C2 to hold, further technical protection measures are in place as explained below. In the following, we explain the functionality of the network in more detail.

Wegman-Carter authentication [39] is used to guarantee authenticity of the channels. The initial authentication is done using a preshared key. Further key material for this authentication is generated using QKD.

We explain the key exchange mechanism. On the quantum layer, nodes that are directly connected via a QKD link can generate raw key material. QKD transmitters and receivers are assumed to be located in the trusted nodes. The exact configurations of the QKD links and protocols in use are shown in Table 2 with achieved key generation rates.

Once keys have been generated in the quantum layer, they are pushed up to the key management layer and then stored and managed by the key management agents (KMAs). All the KMAs are placed in the trusted nodes. The KMAs are connected by authenticated channels, and execute key relays by key encapsulation in a hop-by-hop fashion. Thus a key pair can be shared between two terminal nodes even if they are not directly connected by a QKD link. A reliable key management server (KMS) is also located at one of the trusted nodes, gathers link information (bit error rates, key rates, amounts of accumulated keys, etc.) from the KMAs, organizes a routing table, and provisions secure paths to the KMAs. Secure key transfer is made on request from the KMA to the document owner/shareholder via a protected classical channel, e.g., a tamper resistant cable of short distance. This guarantees non-interceptable key transfer from the QKD platform to the document owner/shareholder located outside the trusted nodes. Furthermore, the trusted nodes are protected by one-way firewalls to prevent attackers from sending malicious commands from the application layer to the QKD platform. Once supplied with the keys, the document owner and the shareholders are in charge of key management. Thus the boundary of responsibility (point-of-interface) is set between the QKD platform and the application layer. For further details we refer to [33, 13].

Secret sharing. Our implementation of secret sharing is based on the secret sharing scheme proposed by Shamir [35]. The scheme provides information-theoretic confidentiality as required by Assumption C3. We use a (3,4)-threshold secret sharing, which suits the network structure of the Tokyo QKD Network. This means that the document owner dis-

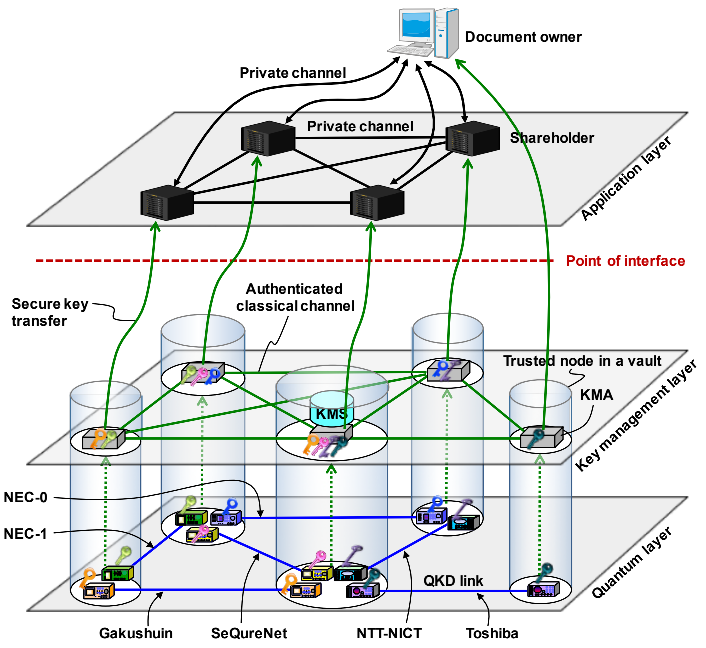

Figure 4: The secret sharing scheme supported by the Tokyo QKD Network.

Name	Protocol	Length	Key rate
		km	kb/s
NEC-0	BB84	50	200
NEC-1	BB84	22	200
Toshiba	BB84	45	300
NTT-NICT	DPS-QKD	90	10
Gakushuin	CV-QKD	2	100
SeQureNet	CV-QKD	2	10

Table 2: Specifications of the QKD links.

Security	SHA-2	RSA	Pedersen
year	instance	$\log_2(n)$	$\log_2(p)$, $\log_2(q)$
2040	SHA-224	2048	2048, 224
2065	SHA-224	3072	3072, 224
2085	SHA-256	4096	4096, 256
2103	SHA-384	5120	5120, 384
2116	SHA-384	6144	6144, 384

Table 3: Parameter selection according to Lenstra [25].

tributes shares to 4 shareholders and 3 shareholders are needed for the reconstruction of the data. To allow for sharing data of arbitrary size, these data are decomposed into parts of appropriate size. Our implementation supports the resharing protocol described in Section 2. For resharing, the document owner first retrieves and reconstructs the data and then generates and distributes new shares. As required by Assumption C4 we assume that resharing happens before the adversary corrupts more than 2 shareholders. In the future we plan to implement proactive secret sharing as suggested in [21]. It allows for taking the document owner out of the loop when resharing happens. Furthermore, it is desirable to have a system that does not involve the document owner in the commitment renewal process. However, this requires more research.

6. EXPERIMENTAL EVALUATION

In the following, we present a performance analysis of LINCOS. We estimate the storage space required by the system and investigate data transmission limits imposed by QKD. We also measure the time required for integrity verification. To do so, we run the following experiment. A document is stored and protected using LINCOS over a period of 100 years, starting in 2016 and ending in 2116. Share and timestamp renewal happen every two years. The share renewal period is to be chosen such that mobile adversaries are unable to recover more shares than permissible. Also, the typical storage hardware maintenance service interval is two years. The timestamp renewal period is chosen in accordance with typical certificate renewal periods. Such certificates are required to verify the timestamps. Finally, commitment renewal happens every ten years. This is in accordance with the heuristic security assumptions for the commitment scheme parameters.

Parameter choice for the complexity-based cryptographic components is done according to the heuristics in [25]. The corresponding expected protection periods are presented in Table 3.

6.1 Storage space

We analyze the storage space required by the various parties of LINCOS. It is analyzed as a function of the bitlength $\mathsf{size_d}$ of the protected document d.

Shareholders. Each shareholder stores one share s per document. Its size is $\mathsf{size_s} = \mathsf{size_d} + \mathsf{size_R}$. Here R is the list of decommitment values accumulated over time. Its size is independent of the document size. The size of a single de-

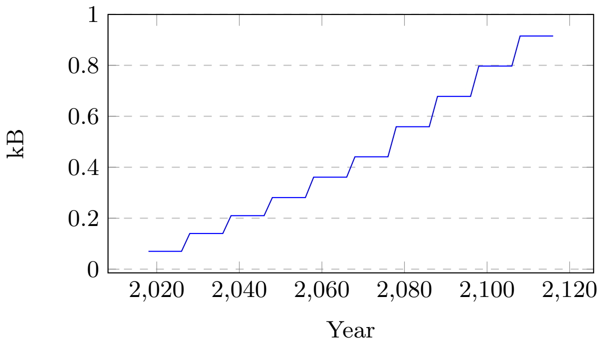

Figure 5: Accumulated size of decommitment values.

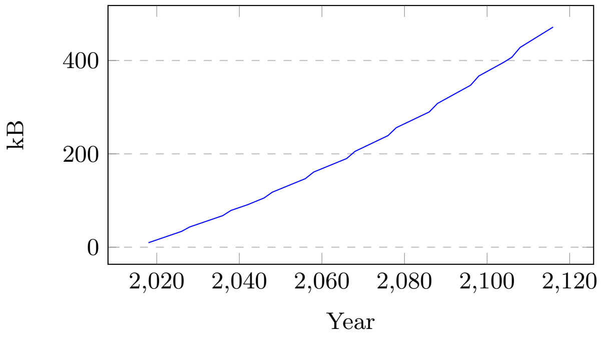

Figure 6: Size of evidence record.

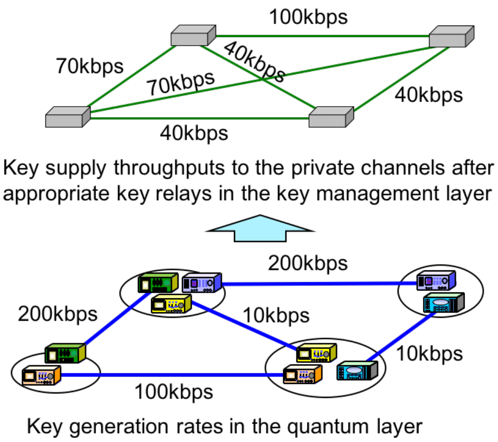
100kbps
70kbps
40kbps
70kbps
40kbps
40kbps
Key supply throughputs to the private channels after
appropriate key relays in the key management layer
200kbps
200kbps
10kbps
10kbps
100kbps
Key generation rates in the quantum layer

Figure 7: Key generation rates of the QKD links, and key supply throughputs to the private channels after appropriate key relays in the key management layer.

$\mathsf{size_s}$	**Sharing**	**Resharing**
1 kB	0.2 s	0.4 s
1 MB	3 min 20 s	6 min 40 s
1 GB	2.3 days	4.6 days

Table 4: Key exchange time for sharing and resharing with $\mathsf{keyRate_{QKD}} = 40$ kb/s.

commitment value equals the size of the parameter q of the commitment scheme. At present, a secure instantiation of the Pedersen commitment scheme requires a decommitment value size of 224 bit. The growth of $\mathsf{size_R}$ over 100 years is shown in Figure 5. The experiments show that it is at most 1 kB.

Evidence service. The evidence service stores one evidence record E per document. The size of the evidence record $\mathsf{size_E}$ is independent of the document size. It depends on the size and number of timestamps and commitments contained in the evidence record. It grows over time because a new timestamp and a new commitment are added with each renewal. The growth of $\mathsf{size_E}$ over time is shown in Figure 6. Our experiments show that the size of the evidence record accumulates over 100 years to $\mathsf{size_E} \approx 500$ kB.

6.2 Data transmission

Our system uses authenticated and private channels. Authenticated channels easily allow for data rate of 1 Gb/s, while they are used for sending only a few hundred kB of evidence data. So the cost for data transmission via authenticated channels is negligible.

Private channels are realized using OTP and QKD. The transmission rate of these channels is limited by the key generation rate of QKD. Therefore, in our analysis we focus on the QKD part.

Data rate of private channels. The QKD performance in the Tokyo QKD Network differs from link to link because fiber channel lengths as well as specifications of QKD devices are different from each other. Furthermore, some nodes are directly connected by a QKD link, others have to use key relay. The achieved secret key rates of the QKD links are summarized in Table 2. They range from 10 kb/s to 300 kb/s depending on the specification of the respective QKD link. To prevent being limited by the slowest QKD links (10 kb/s), keys are relayed between appropriate KMAs such that OTP keys can be supplied at a reasonable key supply throughput, which is denoted as $\mathsf{keyRate_{QKD}}$. Such key relaying balances the key material across the network. The resulting throughput lies between the slowest and fastest key generation rates of the QKD links. In our current configuration of the QKD platform, this key relay allows to raise the minimum throughput of key supply for each pair of four shareholders to $\mathsf{keyRate_{QKD}} = 40$ kb/s, as shown in Figure 7.

Storage and retrieval. When the document owner stores data in the confidentiality system, he sends one share to each shareholder. Likewise, when retrieving the data, the document owner receives one share per shareholder. Since $\mathsf{size_s} = \mathsf{size_d} + \mathsf{size_R}$, the time required for generating the necessary OTP key material per share transfer in a private channel is $t_s = \mathsf{size_s}/\mathsf{keyRate_{QKD}}$ seconds. Table 4 shows timings for shares of different sizes with $\mathsf{keyRate_{QKD}} = 40$ kb/s.

Share renewal. For share renewal, the document owner retrieves the current set of shares and distributes new shares to the shareholders. So the time for communicating the key material required for resharing is $2 * t_s$. Table 4 also lists timings for shares of different sizes.

The data that can be protected when resharing happens

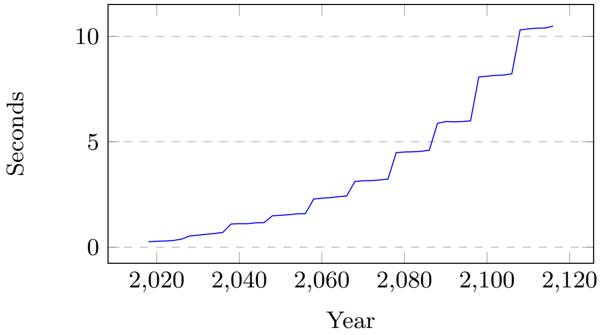

Figure 8: Performance of evidence verification.

every two years as in our experiment has maximum size $\mathsf{size_s} = 2$ years $* \mathsf{keyRate_{QKD}}/2 = 1$ year $* \mathsf{keyRate_{QKD}}$. For the current key supply throughput of 40 kb/s we obtain $\mathsf{size_s} = 158$ GB. This data size approximately corresponds to human genomic data of 195 persons. In the near future (4 to 5 years), QKD technology with key rates of 1 Mb/s over 50 km is expected to be available. Then, data of size up to 3942 GB can be protected, which is roughly 4 TB or the size of the genomes of 4926 persons. If the key supply throughput can be increased to 1 Gb/s, data of size 4 PB can be handled, which corresponds to human genomic data of 4.9 million persons. Such a QKD performance can be expected to be realizable using dense wavelength division multiplexing of 1000 quantum channels as well as fast key distillation processing. This is a challenge, but will be feasible by employing integrated photonic technologies and dedicated key distillation engines on semiconductor chips.

6.3 Evidence verification

Figure 8 shows timings for verification of an integrity proof. The timings were measured on a machine with an 2.9GHz Intel Core i5 CPU and 8GB RAM running our Java implementation of the verification algorithm. As the evidence record and the list of decommitment values grow over time, the verification time increases. Verification of evidence accumulated over 100 years takes approximately 10 seconds. It can be expected that, because computers are getting faster, in a hundred years from now integrity proof verification will only take a fraction of this time.

6.4 Summary

Our experimental evaluation shows the following situation. The long-term integrity system based on COPRIS has very good performance, in particular in view of the expected growth of computing power. So the time and space cost for time-stamping commitments instead of hash values and for renewing these commitments is negligible. As expected, information-theoretic confidentiality protection is expensive. One limiting factor is the additional space required by secret sharing. However, it does not exceed the additional storage space required by cloud storage solutions that use secret sharing for robustness reasons. The second limiting factor is QKD. It is technically complex and transmission rates are not yet fully satisfactory. But, as we have explained above, the development in this area is promising so that practical solutions can be expected in the future.

7. CONCLUSION

We have presented the system LINCOS. It is the first storage system that, under appropriate assumptions, simultaneously provides integrity, authenticity, and confidentiality protection in the long-term (e.g., decades, centuries). LINCOS builds on the newly developed long-term integrity scheme COPRIS, which is the first such scheme that preserves information-theoretic confidentiality of the protected data. We have presented an implementation and an experimental evaluation of LINCOS. One of the special features of this implementation is that it uses QKD for establishing private channels for secret sharing. With current QKD technology our LINCOS implementation allows to store and protect data of size up to approximately 150 GB. Current performance is limited by QKD key transmission rates. We have reported about predictions of future developments that will improve QKD transmission rates considerably.

There are two main directions of further research. The first concerns improvement of QKD performance. As mentioned in Section 6.2 dense wavelength division multiplexing of many quantum channels combined with fast key distillation processing is promising. This requires employing integrated photonic technologies and dedicated key distillation engines on semiconductor chips.

The second research direction concerns the performance of the commitment renewal and resharing process. Currently, in both processes the document owner is required to retrieve the document regularly. However, it is desirable to take the document owner out of the loop and let the confidentiality and integrity systems deal with these issues independently. For proactive secret sharing, we will use a more advanced resharing protocol, e.g., [19]. It allows for renewing the shares without the help of the document owner. We also aim at developing a commitment renewal protocol that does not involve the document owner.

8. ACKNOWLEDGEMENTS

This work has been co-funded by the DFG as part of project S6 within the CRC 1119 CROSSING, by the European Union's Horizon 2020 research and innovation program under Grant Agreement No 644962, and by the ImPACT Program of the Council for Science, Technology and Innovation (Cabinet Office, Government of Japan).

9. REFERENCES

[1] https://www.keylength.com.

[2] C. Adams, P. Cain, D. Pinkas, and R. Zuccherato. Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). RFC 3161 (Proposed Standard), Aug. 2001. Updated by RFC 5816.

[3] F. Bahr, M. Boehm, J. Franke, and T. Kleinjung. Factorization of rsa-200. *Public announcement on May 9th*, 2005.

[4] M. Bellare and P. Rogaway. *CRYPTO' 93*, chapter Entity Authentication and Key Distribution, pages 232–249. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.

[5] G. Brassard, C. Crépeau, D. Mayers, and L. Salvail. A brief review on the impossibility of quantum bit commitment. *arXiv preprint quant-ph/9712023*, 1997.

[6] J. Braun, J. Buchmann, C. Mullan, and A. Wiesmaier. Long term confidentiality: a survey. *Designs, Codes*

and Cryptography, 71(3):459–478, 2014.

[7] A. Buldas, M. Geihs, and J. Buchmann. Long-term secure commitments via extractable-binding commitments. In J. Pieprzyk and S. Suriadi, editors, *Information Security and Privacy*, pages 65–81, Cham, 2017. Springer International Publishing.

[8] A. Buldas, M. Geihs, and J. Buchmann. Long-term secure time-stamping using preimage-aware hash functions. In T. Okamoto, Y. Yu, M. H. Au, and Y. Li, editors, *Provable Security*, pages 251–260, Cham, 2017. Springer International Publishing.

[9] R. Canetti and A. Herzberg. Maintaining security in the presence of transient faults. In Y. G. Desmedt, editor, *Advances in Cryptology — CRYPTO '94*, pages 425–438, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[10] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *Foundations of Computer Science, 1985., 26th Annual Symposium on*, pages 383–395. IEEE, 1985.

[11] Y. Desmedt and S. Jajodia. Redistributing secret shares to new access structures and its applications. Technical report, George Mason University, 1997.

[12] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), Aug. 2008. Updated by RFCs 5746, 5878, 6176, 7465, 7507, 7568, 7627, 7685.

[13] M. Fujiwara, A. Waseda, R. Nojima, S. Moriai, W. Ogata, and M. Sasaki. Unbreakable distributed storage with quantum key distribution network and password-authenticated secret sharing. *Scientific Reports*, 6, 2016.

[14] M. Geihs, D. Demirel, and J. Buchmann. A security analysis of techniques for long-term integrity protection. In *Privacy, Security and Trust 2016*, 2016.

[15] M. Geihs, D. Demirel, and J. Buchmann. A security analysis of techniques for long-term integrity protection. In *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, pages 449–456, Dec 2016.

[16] O. Goldreich. *Foundations of Cryptography – Volume 1*, chapter Perfectly Hiding Commitment Schemes. Cambridge University Press, 2001.

[17] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, Apr. 1988.

[18] T. Gondrom, R. Brandner, and U. Pordesch. Evidence Record Syntax (ERS). RFC 4998 (Proposed Standard), Aug. 2007.

[19] V. H. Gupta and K. Gopinath. G_{its}^2 VSR: An information theoretical secure verifiable secret redistribution protocol for long-term archival storage. In *Security in Storage Workshop, 2007. SISW '07. Fourth International IEEE*, pages 22–33, Sept 2007.

[20] S. Haber and W. S. Stornetta. *CRYPTO '90*, chapter How to Time-Stamp a Digital Document, pages 437–455. Springer Berlin Heidelberg, Berlin, Heidelberg, 1991.

[21] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. *CRYPTO '95*, chapter Proactive Secret Sharing Or: How to Cope With Perpetual Leakage, pages 339–352. Springer Berlin Heidelberg, Berlin, Heidelberg, 1995.

[22] D. Hühnlein, U. Korte, L. Langer, and A. Wiesmaier. A comprehensive reference architecture for trustworthy long-term archiving of sensitive data. In *2009 3rd International Conference on New Technologies, Mobility and Security*, pages 1–5, Dec 2009.

[23] T. Kuroda, E. Kimura, Y. Matsumura, Y. Yamashita, H. Hiramatsu, and N. Kume. Simulating cloud environment for HIS backup using secret sharing. *Studies in health technology and informatics*, 192:171–174, 2012.

[24] T. Kuroda, E. Kimura, Y. Matsumura, Y. Yamashita, H. Hiramatsu, N. Kume, and A. Sato. Applying secret sharing for HIS backup exchange. In *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 4179–4182. IEEE, 2013.

[25] A. K. Lenstra. Key lengths. In *The Handbook of Information Security*. Wiley, 2004.

[26] H.-K. Lo and H. F. Chau. Unconditional security of quantum key distribution over arbitrarily long distances. *science*, 283(5410):2050–2056, 1999.

[27] T. Loruenser, A. Happe, and D. Slamanig. Archistar: Towards secure and robust cloud based data sharing. In *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 371–378, Nov 2015.

[28] D. Mayers. Unconditional security in quantum cryptography. *Journal of the ACM (JACM)*, 48(3):351–406, 2001.

[29] R. Ostrovsky and M. Yung. How to withstand mobile virus attacks (extended abstract). In *Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '91, pages 51–59, New York, NY, USA, 1991. ACM.

[30] T. P. Pedersen. *CRYPTO '91*, chapter Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing, pages 129–140. Springer Berlin Heidelberg, Berlin, Heidelberg, 1992.

[31] T. A. Ramos, N. da Silva, L. C. Lung, J. G. Kohler, and R. F. Custódio. An infrastructure for long-term archiving of authenticated and sensitive electronic documents. In *EuroPKI*, pages 193–207, 2010.

[32] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, Feb. 1978.

[33] M. Sasaki et al. Field test of quantum key distribution in the tokyo qkd network. *Opt. Express*, 19(11):10387–10409, May 2011.

[34] V. Scarani, H. Bechmann-Pasquinucci, N. J. Cerf, M. Dušek, N. Lütkenhaus, and M. Peev. The security of practical quantum key distribution. *Rev. Mod. Phys.*, 81:1301–1350, Sep 2009.

[35] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, Nov. 1979.

[36] C. E. Shannon. Communication theory of secrecy systems. *The Bell System Technical Journal*, 28(4):656–715, Oct 1949.

[37] G. Vernam. Secret signaling system, July 22 1919. US Patent 1,310,719.

[38] M. A. G. Vigil, J. A. Buchmann, D. Cabarcas, C. Weinert, and A. Wiesmaier. Integrity, authenticity, non-repudiation, and proof of existence for long-term archiving: A survey. *Computers & Security*, 50:16–32, 2015.

[39] M. N. Wegman and J. L. Carter. New hash functions and their use in authentication and set equality. *Journal of computer and system sciences*, 22(3):265–279, 1981.

[40] T. M. Wong, C. Wang, and J. M. Wing. Verifiable secret redistribution for archive systems. In *Security in Storage Workshop, 2002. Proceedings. First International IEEE*, pages 94–105. IEEE, 2002.