A QKD-based digital archiving solution providing everlasting confidentiality and integrity

Johannes Braun Johannes Buchmann, Denise Demirel, Matthias Geihs
Technische Universität Darmstadt, Germany
{jbraun,buchmann,ddemirel,mgeihs}@cdc.informatik.tu-darmstadt.de

Mikio Fujiwara, Shiho Moriai, Masahide Sasaki, Atsushi Waseda
National Institute of Information and Communications Technology, Japan
{fujiwara,shiho.moriai,psasaki,a-waseda}@nict.go.jp

ABSTRACT

With increasing digitization, the amount of archived data that requires long-term protection of confidentiality and integrity increases rapidly. Examples include electronic health records, genome data bases, and tax data. In this paper we present the first archiving solution that provides everlasting confidentiality and, at the same time, maintains a proof that the data were not changed since they were archived. For confidentiality protection, our solution combines quantum key distribution (QKD) for data in transit and proactive secret sharing for data at rest. Proofs of existence are generated using sequences of timestamped unconditionally hiding commitments. In addition to a security and performance analysis, we present a proof-of-concept implementation and an experimental evaluation. It uses the QKD platform of the National Institute of Information and Communication Technology of Japan.

1. INTRODUCTION

1.1 Motivation and problem statement

With increasing digitization, the amount of archived data increases rapidly. Such data require *integrity* protection and, in many cases also their *confidentiality* must be guaranteed. Confidentiality means that only authorized parties can access the data. Integrity protection provides a *proof of exixtence* which allows to verify that the data in the present form existed at some earlier time, for example, when the data were archived. In fact much of the archived data is kept for a *very long time*. For such data *long-term protection* is necessary. For example, electronic health records and genome data bases may have to be kept as long as the respective persons are alive or even beyond this date. So the archiving and protection period may be more than 100 years.

In this paper we address the problem of providing long-term confidentiality and integrity protection of archived data. More precisely, we aim at everlasting confidentiality protection and maintaining a proof that the data existed at archiving time. We consider solving this problem to be of major importance now and even more in the future.

However, this problem appears to be very challenging. Cryptographic techniques used today for providing integrity and confidentiality do not allow for long-term protection. Keys chosen today will be too short in the future. New attacks using quantum computers may become real (see [4]) thereby compromising many of the current cryptographic schemes. There are some partial solutions to the long-term protection problem. But as explained in Section 1.3 they do not allow for a comprehensive solution of the problem. For example, the current long-term integrity solutions compromise the confidentiality of the archived data.

1.2 Contribution

In this paper we present a solution to the problem simultaneously protecting confidentiality and integrity of archived data in the long-term. To the best of our knowledge this is the first such solution. With respect to integrity protection, our new idea is to use chains of timestamped unconditionally hiding commitments. Timestamping such commitments rather than the protected documents as in the known schemes (see [35]) makes our integrity solution compatible with confidentiality protection. In regard to confidentiality protection we combine proactive secret sharing with quantum key distribution (QKD). The known proactive secret sharing solutions assume the existence of private channels without discussing their realization. Our idea is to realize private channels using QKD techniques. So we bring together techniques that have so far been studied in separate communities.

Our contribution does not stay on the conceptual level. Instead, we present a full fledged implementation and experimental evaluation. The quantum key distribution part of this implementation uses the very advanced Tokyo QKD Network [29]. The cryptographic tools are provided by TU Darmstadt. Using our implementation we are able to provide experimental data. They demonstrate that our solution is promising.

In short, our solution works as follows. In regard to confidentiality we distinguish between the protection of data in transit and data at rest. For example, data are in transit when they are sent to the archive and they are at rest in the archive. Confidentiality protection of data in transit is achieved by a combination of one-time pad encryption [34] and quantum key distribution [14, 30]; both are known to be unconditionally secure (see information-theoretic security [32]). Confidentiality of data at rest is achieved by proactive secret sharing as described in [21, 19]. The idea is to decompose the data to be protected into shares that are distributed to a number n of *shareholders*. From a threshold number t, ($t \leq n$) of these shares the secret data can be reconstructed. However, less than t shares do no yield any information about the secret. The scheme is called proactive as it provides protection against *mobile adversaries* which may be able to learn shares over time. This protection is achieved by running a resharing protocol between the shareholders.

To maintain a proof of existence we use sequences of timestamped cryptographic commitments. Timestamping commitments rather than the archived documents allows for maintaining confidentiality. In fact, the commitments are unconditionally hiding which allows for everlasting confidentiality protection as desired. Initially, such a commitment to the archived data is timestamped. Unfortunately, unconditionally hiding commitments can only be computationally binding. Hence, before a commitment looses its bindingness, a new commitment and a new timestamp are generated. When the data are revealed, the verification of the proof of existence is performed as follows. The timestamp sequence is recursively verified, proving that the commitments existed at their respective generation times. Also, the commitments are recursively opened proving eventually, that the first commitment opens to the archived data and therefor, these data existed at archiving time.

1.3 Related work

Confidentiality and integrity protection can be achieved by encryption and digital signatures, respectively (see [7]). However, the encryption and signature schemes used today are complexity-based. This means that their security is based on the intractability of some computational problem such as integer factorization. Such schemes do not provide long-term protection. Cryptographic keys chosen today will be too short in the future. New attacks such as quantum computer attacks may become real (see [4]) thereby compromising many of the current cryptographic techniques. Researchers have proposed solutions that address the challenges of long-term confidentiality and integrity protection individually. On the one hand, there are long-term confidentiality protection solutions (see [6]). They refer to *data in transit* and *data at rest*. For example, data are in transit when they are sent to an archive and they are at rest when they are stored in the archive. Everlasting confidentiality protection of data in transit can be achieved by QKD-based techniques (see [33, 24]). The confidentiality of data at rest can be protected in the long-term by proactive secret sharing, e.g. [21, 10, 36, 18, 19]. On the other hand, long-term integrity protection is for example possible with timestamp chains as described in [35] and [13, 12, 20, 17, 5]. Unfortunately, the known schemes for long-term integrity protection cannot be combined with long-term confidentiality protection since hashes of the protected data are required which may in the long-term reveal information about the data, for example to the timestamp authority which learns the hash of the data. This is why the solutions in [22] and [27] do not provide long-term confidentiality protection.

1.4 Organization

Our paper is organized as follows. Section 2 introduces the cryptographic components that are being used in our framework and discusses their security. Section 3 presents our new framework. In Section 4 we discuss the security of our framework. Our implementation is described in Section 5. A theoretical performance analysis is provided in Section 6 while the experimental results are shown in Section 7. Finally, in Section 8 we draw conclusions and sketch future work.

2. CRYPTOGRAPHIC COMPONENTS

In this section we present the cryptographic components that are used in our framework which is shown in Figure 1 and explained in Section 3.

Each cryptographic component may consist of a collection of algorithms and/or protocols. The cryptographic components may provide *computational* or *unconditional* security. Computational security is based on the intractability of some computational problem such as integer factorization. Such components do not provide long-term protection. If the components are unconditionally secure, then their security is everlasting.

Before a component can be used it is *instantiated.* We explain what we mean by this conceptually. Then we give examples. Unconditionally secure components are instantiated by choosing an implementation of this component. For computationally secure components, first an *expiration date* is chosen until which the component is expected to be secure. Next, an *implementation* of the component is chosen. For example, the RSA signature scheme is an implementation of a digital signature scheme. Then, *parameters* are selected that allow for the implementation to be secure until the expiration date. When we say that a cryptographic component is used we mean that an *instance* of this component is used which consists of an expiration date and implementations and parameters for each cryptographic algorithm or protocol of the component. For each cryptographic component we present the corresponding security guarantees.

For example, we use *timestamp schemes.* A timestamp scheme consists of the algorithms Stamp and Verify. Algorithm Stamp receives as input some data d and returns a timestamp T on d and the current time t. Input of algorithm Verify is data d, a time t, a timestamp T, and a trust anchor TA. Typically, the trust anchor is the public key of some root certification authority. It must be known to a verifier. Output is either 0 or 1 where 1 means that the timestamp proves the existence of d at time t and 0 stands for the timestamp being invalid. The security notion for timestamp schemes is *unforgeability* which is defined analogous to signature unforgeability (see [16]).

This is a rather abstract description. In real applications we need to explicitly say which timestamp scheme is being used. For example, we can use the timestamp scheme from [2] based on the RSA signature scheme. The parameters of such a scheme include a cryptographic hash function and an RSA-modulus. These parameters are chosen such that they allow for the scheme to be secure until the chosen expiration date. In Section 3 we will denote the corresponding instance by TSI.

Let us consider another example of a cryptographic component which involves cryptographic protocols: an *authenticated channel.* It consists of the protocols Setup and Send. When we describe protocols we start by presenting the parties involved in the protocol. In the case of the authenticated channel, they are the *sender* and the *receiver.* As for all protocols, we next specify *initial* and *final state* of the protocols. The initial state of the Setup protocol is trivial: sender and receiver use the same instance of the protocol. This is an obvious assumption and we will not always mention it. After an execution of Setup both sender and receiver have the same channel parameters. An example for such a channel parameter is a session key. Now we turn to the Send protocol. Initially, the sender has some data d. After a protocol run, the receiver also has these data. The channel guarantees mutual authentication in a computational sense. For details see [3]. Instances of authenticated channels will be described in Section 5.

In addition to authenticated channels we also use *private channels.* Private channels also involve a sender and a receiver and support the same protocols as authenticated channels: Setup and Send. However the security guarantees are stronger. In addition to computationally secure mutual authentication private channels also support unconditionally secure confidentiality (see [34, 32]).

Next, our framework uses *commitment schemes.* A commitment schemes is composed of the algorithms Commit and Verify. Input to Commit is some data. Output is a *commitment value* and a *decommitment value.* Correspondingly, input to Verify are data, a commitment value, and a decommitment value. Output is 1 or 0. The purpose of such a scheme is to allow committing to some data without revealing the data. We require the commitment schemes to be *computationally binding* and *unconditionally hiding.* For details see [26, 15].

Finally, we use *proactive secret sharing schemes* in which a *dealer*, several *shareholders*, and a *retriever* participate. Such schemes allow unconditional confidentiality of data at rest. When this component is initialized, a first set of shareholders is selected. However, the set of shareholders may change over time. Proactive secret sharing schemes are composed of the protocols Setup, Share, Reshare, and Retrieve. In the Setup protocol, dealer and shareholders agree on sharing parameters, e.g., 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. In the Reshare protocol a current set S and a new set S' of shareholders are involved. They are mutually connected by private channels. Initially, the shareholders in S have their individual shares. After the protocol has terminated, all 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 is able to reconstruct the initial data of the dealer.

We require that the proactive secret sharing schemes that we use are unconditionally hiding in the mobile adversary model as introduced in [21]. In this model, adversaries may be able to learn some shares over time. Proactive secret sharing prevents the success of such adversaries by renewing the shares on a regular basis.

3. FRAMEWORK

The purpose of the framework which is described in this section and depicted in Figure 1 is everlasting confidentiality protection of a document d that belongs to some *document owner* and maintaining a proof that d existed at archiving time t_0. In our framework, the cryptographic components from Section 2 are used. In principle, our framework works as follows.

For integrity protection, an *evidence service* maintains *evidence records* which are essential for the proof of existence. This evidence service is not a trusted third party. It just simplifies the work for the document owner. The evidence service uses timestamped commitments that are issued on a regular basis by the document owner. Confidentiality of the document and the decommitment values are protected by proactive secret sharing. Confidentiality with respect to the evidence service is provided by the commitments being unconditionally hiding.

When the document is revealed to some third party by the document owner, he reconstructs the document and the decommitment values of all commitments generated so far using the most recent shares. He requests an evidence record from the evidence service. The proof of existence consists of the evidence record and the decommitment values. The document owner transmits the document and the proof of existence to the third party. The third party can verify the proof of existence using the information it has received.

In the sequel, we present a more technical description of the framework. The involved parties are the document owner, the evidence service, and later a third party to which the document is revealed. The document owner may be replaced by an *archivist* who manages document protection on behalf of several document owners.

Initialization.

First, the framework is initialized. The initialization includes selecting a commitment scheme CSI_0, a trusted timestamp authority which uses a timestamp scheme TSI_0, and a proactive secret sharing scheme PSS together with an initial set S of shareholders. Also, mutual private channels are established between the document owner and the initial shareholders. Furthermore, an authenticated channel is set up between the document owner and the evidence service. This channel is updated whenever its security is about to expire. For all the cryptographic components that only provide computational security, expiration dates are chosen and parameters that provide security of the components until the expiration dates.

Document protection.

Protection of the document d is managed by the document owner (or the archivist on behalf of the document owner). Document protection is initialized by computing a commitment value c_0 and a decommitment value r_0 for d using the Commit algorithm of the commitment scheme CSI_0. The commitment value c_0 is sent to the evidence service. To protect the confidentiality of the document, the document owner (who acts as the dealer) uses the Share protocol of the proactive secret sharing scheme PSS to share (d, r_0) among the current shareholders.

At certain times, the protection is renewed. This either happens shortly before the regular expiration of the current commitment scheme or when this component is in danger of becoming insecure. In this case the commitment scheme is replaced by a new instance CSI_i.

The renewal works as follows. The document owner runs the Retrieve protocol of PSS with the current shareholders and obtains d and the sequence of decommitment values $y_{i-1} = (r_0, \ldots, r_{i-1})$. The document owner computes a new commitment value c_i and a decommitment value r_i for (d, y_{i-1}) using the Commit algorithm of the new commitment scheme instance. He sets $y_i = (r_0, \ldots, r_{i-1}, r_i)$. Then he applies the Share protocol of the proactive secret sharing scheme to distribute (d, y_i) among shareholders.

Constructing evidence records.

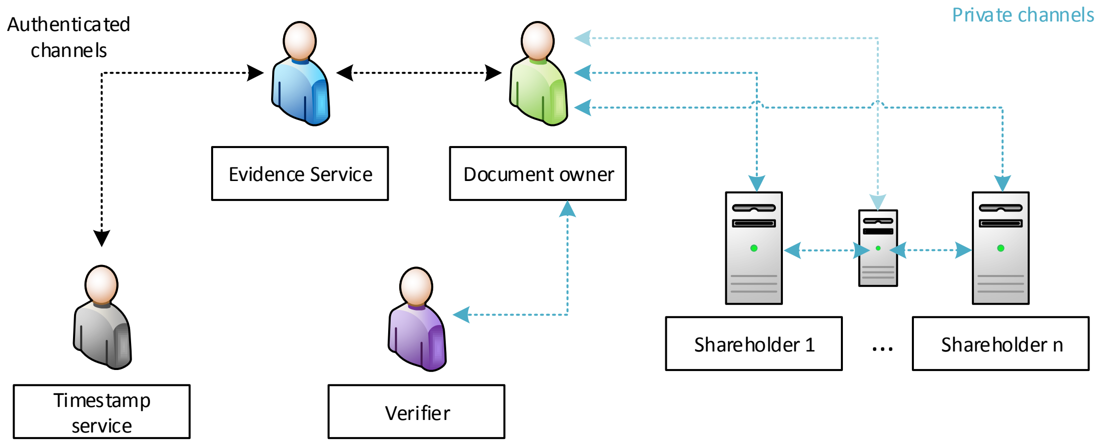

Figure 1: The archiving framework.

The evidence service creates *evidence records* that are an essential part of the proof of existence and do not reveal any information about the archived document.

To compute the initial evidence record, the evidence service obtains the commitment value c_0 from the document owner. Then she requests a timestamp on c_0 from the trusted timestamp authority. The timestamp authority applies the Stamp function of the current timestamp scheme and returns the timestamp T_0 together with the current time t_0. The initial evidence record is $e_0 = (c_0, T_0, t_0)$.

The evidence record is renewed when the commitment scheme or the timestamp scheme are about to expire or are expected to become insecure. This renewal is done as follows. Suppose that $i-1$ renewals have been made, $i > 0$. Then $e_{i-1} = (c_0, T_0, t_0, \ldots, c_{i-1}, T_{i-1}, t_{i-1})$ is the current evidence record. If the commitment scheme was updated, then c_i is the new commitment value received from the document owner. Otherwise, the evidence service uses the old commitment value and sets $c_i \leftarrow c_{i-1}$. Then the evidence service requests a timestamp on (e_{i-1}, c_i) from the timestamp authority. It uses a new timestamp scheme TSI_i which may coincide with the old one if it is still secure for some time. She receives (T_i, t_i). The new evidence record is $e_1 = (c_0, T_0, t_0, \ldots, c_i, T_i, t_i)$.

Resharing.

To provide protection against mobile adversaries, the shares are renewed on a regular basis. The shareholders agree when this should happen. This prevents a mobile adversary to take advantage of shares he may have been able to obtain. The current set of shareholders may also be replaced by a new set of shareholders. Resharing is done by running the Reshare protocol on the current shares.

Revealing the document.

At any point in time t, the document owner may reveal the document to a *verifier* and to prove that it existed at time t_0. To do this, the document owner executes the following steps. He requests the current evidence record $e_k = (c_0, T_0, t_0, \ldots, c_k, T_k, t_k)$ from the evidence service. He also retrieves the document d and the list of decommitment values $y_k = (r_0, \ldots, r_k)$ by running the Retrieve protocol for the proactive secret sharing scheme. He sends the current proof of existence (d, e_k, y_k) together with t_0 to the intended verifier. To protect the confidentiality of the document these data are sent through a private channel.

The verifier to whom the document has been revealed, does the following to verify the proof of existence. Assume that the verification happens at time t and that T_k is the most recent timestamp and let $t_{k+1} = t$.

For $j = k, k-1, \ldots, 0$ the verifier executes the following steps. She checks that TSI_j was secure in time interval $[t_j, t_{j+1}]$ and that T_j is valid. Next, the verifier checks that CSI_j was secure in time interval $[t_j, t_{j+1}]$ and that c_j opens with r_j to (d, y_{j-1}). For $j = 0$ this means that c_0 opens with r_0 to d. Since y_{-1} is the empty sequence, this completes the proof that d existed at time t_0. A more formal proof will be given in Section 4. We note that for the initial timestamp verification, the verifier must know beforehand which timestamp and which commitment scheme were used with which parameters and she must know the trust anchor required to verify T_k. If $j < k$ this information is included in timestamp T_{j+1}.

4. SECURITY ANALYSIS

In this section we analyze the security of our solution. We first describe the adversary model. Afterwards, we present the security analysis.

We consider an adversary who is active and mobile. Active adversaries may deviate from the protocol or may corrupt parties to act on their behalf. Mobile adversaries may corrupt different parties at different stages of the protocol execution which do not have to be selected at the beginning of an attack [25, 9, 21]. With respect to computation power we consider adversaries that may be active for an unbounded period of time, but can only do a bounded amount of work per unit of real time (see [8] for a more precise version of this adversary model). The adversary model reflects the indefinite lifetime of long-lived systems and of the data processed by the system. Assume, for instance, the document is sent in encrypted form to the evidence service. When the encryption scheme becomes insecure decades later, attackers can decrypt ciphertexts computed with this scheme and violate confidentiality. But since realistic attackers do not have unbounded computing power, the model limits this power per unit of real time.

First we analyze the security of the proof of existence for a document d.

To be able to demonstrate its security, we need to make a few assumptions. First, we need to assume that timestamp authorities are trustworthy to include the correct time and verification information in their timestamps.

The next assumptions are in fact checked during the verification process. First, the trust anchor used by verifiers of the proof of existence must be correct. Second, when they are used, the cryptographic components must be secure in the sense of Section 2. By this we mean the following. We use cryptographic components that only have complexity-based security: timestamp schemes and commitment schemes. These schemes are introduced at some point in time and then replaced by new ones before they become insecure. We assume that their security is guaranteed between these two points in time. This is also checked in the verification.

THEOREM 4.1. *Under the above assumptions, a proof of existence demonstrates the existence of d in its current form at time t_0.*

PROOF. Suppose that at some point in time a proof of existence is generated. Let T_k be the most recent timestamp. Then the input to the verification algorithm is the initial time t_0 and the current proof of existence (d, e_k, y_k), containing the document d, the current evidence record $e_k = (c_0, T_0, t_0, \dots, c_k, T_k, t_k)$, and the list of decommitment values $y_k = (r_0, \dots, r_k)$.

We prove by induction on $i = k, k-1, \dots, 0$ that a successful ith step of the verification implies that (e_{i-1}, c_i) and (d, y_{i-1}) existed at time t_i. For $i = 0$, this demonstrates our assertion since y_{-1} is the empty sequence and so d existed at time t_0.

First, let $i = k$. Initially, the verifier has successfully checked that TSI_k was secure in time interval $[t_k, t]$ where t is the current time and that T_k is valid. Since the trust anchor is assumed to be correct for this verification and the timestamp authority is assumed to include the correct timestamp, this implies that (e_{k-1}, c_k) existed at time t_k. Next, the verifier has successfully checked that CSI_k was secure in time interval $[t_k, t]$ and that r_k opens c_k to (d, y_{k-1}). The validity of T_k, the existence of r_k at time t, and the existence of c_k at time t_k show that (d, y_{k-1}) existed at time t_k.

Now let $i \le k$ and assume that (e_{i-1}, c_i) and (d, y_{i-1}) existed at time t_i, $0 \le i \le k$.

If $i = 0$, then as shown above the proof of existence is complete.

If $i > 0$, then we need to show that (e_{i-2}, c_{i-1}) and (d, y_{i-2}) existed at time t_{i-1}. The verification has shown that TSI_{i-1} was secure in time interval $[t_{i-1}, t_i]$ and that T_{i-1} is valid. Since the timestamp authority included the correct verification information, it follows that (e_{i-2}, c_{i-1}) existed at time t_{i-1}. Next, the verifier has checked that CSI_{i-1} was secure in time interval $[t_{i-1}, t_i]$ and that c_{i-1} with r_{i-1} opens to (d, y_{i-2}). Note that although c_{i-1} might not be secure anymore at the time of verification t, it still cannot be opened to another value because r_i was fixed at time t_i. Hence we have shown that (e_{i-2}, c_{i-1}) and (d, y_{i-2}) existed at time t_{i-1}. □

Now we show that our framework provides everlasting confidentiality. For this we require that the proactive secret sharing scheme provides unconditional security in the sense of information theory. In particular, we require that if k is the number of shareholders needed to reconstruct the data shared, then no more than $k-1$ shareholders collaborate or are corrupted by the adversary at any point in time. Also, after renewing the shares the non-corrupted shareholders delete their old share. We also require that the private QKD-channels provide unconditional security.

THEOREM 4.2. *Under the above assumptions, our framework provides unconditional confidentiality of d.*

PROOF. Data in transit: During document protection, the document owner sends commitment values to the evidence service and shares to the shareholders. During resharing the shareholders exchange information about the shares and during revealing the document the shareholders return the shares to the document owner who forwards the document to the verifier. The shares of the document and the decommitment values are sent through private channels that guarantee unconditional confidentiality and mutual authentication of sender and receiver. Authenticated channels, which do not guarantee confidentiality, are only used to send commitments which are unconditionally hiding. Hence, this does not affect the unconditional confidentiality of the document.

Data at rest: At the shareholders, the unconditional confidentiality of the archived document is guaranteed by the unconditional secrecy of the proactive secret sharing scheme in the mobile adversary model. At the evidence service, unconditional confidentiality of the archived document is provided because only unconditionally hiding commitments are stored by the evidence service. □

5. IMPLEMENTATION

In this section we describe our implementation of the framework described in Section 3. We implemented the complete framework including QKD except that we only implemented plain secret sharing without resharing. This is sufficient to show the feasibility and obtain a first impression of the performance of our solution. Proactive secret sharing will be added in the next version of the implementation. The current Tokyo QKD Network allows for four shareholders.

In order to be able to simulate long-term protection, our implementation allows for replacing the computationally secure cryptographic components by more secure ones

The implementation is done in Java and C. The system components are hosted on servers at TU Darmstadt and NICT and communicate over web service interfaces. In the following we explain the implementation of the cryptographic components and, in particular, of the QKD-based private channels.

5.1 Implementation of cryptographic components

As commitment scheme which is used by the document owner to generate commitments to documents we use Pedersen commitments [26]. They are unconditionally hiding and computationally binding as required by the framework. The scheme is parametrized by two prime numbers p_c and q_c. The binary length of q_c determines the length of the values that can be committed to. The bindingness of the Pedersen commitment scheme is based on the discrete logarithm problem. The hardness of this problem, and thus the expiration date of a Pedersen instance, depends on the sizes of the parameters p_c and q_c.

To allow committing to data of arbitrary length, data are first hashed and then committed to. Our implementation supports the SHA-2 hash function family which consists of the hash functions SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, and SHA-512/256 for different levels of security. These hash functions are chosen, such that (1) the size of the output fits the maximum data length the used commitment scheme instance can commit to and (2) the expected time the hash function remains secure exceeds the expiration date of the commitment scheme instance.

The timestamp scheme used by the timestamp authority from which the evidence service requests timestamps is instantiated in accordance with the RFC 3161 [2] standard. It ensures unforgeability as required by the framework, given the used signature scheme does provide unforgeability. According to this standard, in addition to time information, the timestamps also include verification information such as public key certificate chains and revocation information. Furthermore, they contain the signature and hash instances used. In our implementation, we also add information about the used commitment scheme instance which is needed in the verification of the proofs of existence.

Our implementation of the timestamp scheme is based on the hash-then-sign paradigm. This means that the data to be timestamped, including the verification information, are hashed together with the current time. Then the hash value is signed with the timestamp authority's private key. As signature scheme our implementation uses RSA [28]. The parameter determining the validity period of an RSA instance is the bitlength of the RSA-modulus n. The supported hash functions with respect to the timestamp scheme are again the instances of the SHA-2 family.

We use Shamir's secret sharing scheme [31]. It provides unconditional secrecy as required by our framework. More precisely, we use (3,4)-threshold secret sharing with a total of four shareholders. Note that shared data have to be split into pieces that fit to the size of the finite field with which Shamir's secret sharing scheme is instantiated. For reconstruction of the data, each piece is reconstructed separately, and then the pieces are concatenated to obtain the original data.

Wherever authenticated channels are used in our framework, these are realized using TLS version 1.2 [11] in the currently most secure configuration. This protocol is state of the art and provides mutual authentication in a computational sense as required by the framework.

Finally, private channels are instantiated using unconditionally secure one-time pad (OTP) encryption which ensures everlasting confidentiality of the data transmission [34, 32]. The OTP encrypted data is sent via classical network links. In our implementation, the required random keys (also referred to as OTP keys) are exchanged using QKD. These are as long as the plaintext and may only be known to the respective sender and receiver. We use the Tokyo QKD Network to exchange OTP encryption keys. The network connects the four shareholders and the document owner. We provide a detailed description of the QKD platform in the following section.

5.2 Private channels using QKD

The private channels connecting the document owner and the shareholders are implemented in a metropolitan field network testbed, called the Tokyo QKD Network [29]. It consists of three layers; *quantum layer*, *key management layer*, and *application layer* as depicted in Figure 2. The document owner, the shareholders, and the private channels are on the application layer, supplied with information theoretically secure keys from the key management layer. The stack of the quantum layer and the key management layer is called the *QKD platform*, and satisfies the much higher security requirement compared to the application layer. In particular, the nodes of the QKD platform are physically protected from attackers. They are residing in a vault, referred to as "trusted nodes". In our experiment, there are five nodes. In the quantum layer, each point-to-point QKD link generates OTP keys. Each QKD link consists of a transmitter and a receiver that are connected with two channels, a quantum channel and an authenticated (classical) channel. The former is an optical fiber cable connecting a pair of authenticated encoder and decoder for transmitting random numbers via quantum signals. The latter is for realizing communications for distilling OTP keys from the random numbers. Any eavesdropping attempt on the quantum channel can be detected as the increase of the bit error rate, and such insecure key generation sessions will be discarded [14, 30]. On the other hand, a QKD link itself is vulnerable to Denial of Service attacks (key generation stops when tapped), and hence a rerouting mechanism is implemented in the network. In our experiment, there are six QKD links. The exact configurations of the QKD links and protocols in use are given in Table 1 with typical key generation rates.

Keys generated in the quantum layer are pushed up to servers, called key management agents (KMAs), in the key management layer, then stored and managed. All the KMAs and the QKD transmitters and receivers are located 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 securely even if they are not directly connected by a QKD link. The KMA resizes the key strings into appropriate key files, saves them with necessary metadata such as key IDs and transmitter/receiver IDs, and authenticates the relayed key file as well as other KMAs in the relay route. A key management server (KMS) is also located at one of the trusted nodes, and gathers link information (bit error rates, key rates, amounts of accumulated keys, etc.) from the KMAs, and 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, i.e a tamper resistant metal cable of short distance. This guarantees non-interceptable key transfer from the QKD platform to the document owner/shareholder located outside the vaults. Furthermore, the trusted nodes are protected by one-way firewalls to prevent attackers from sending mali-

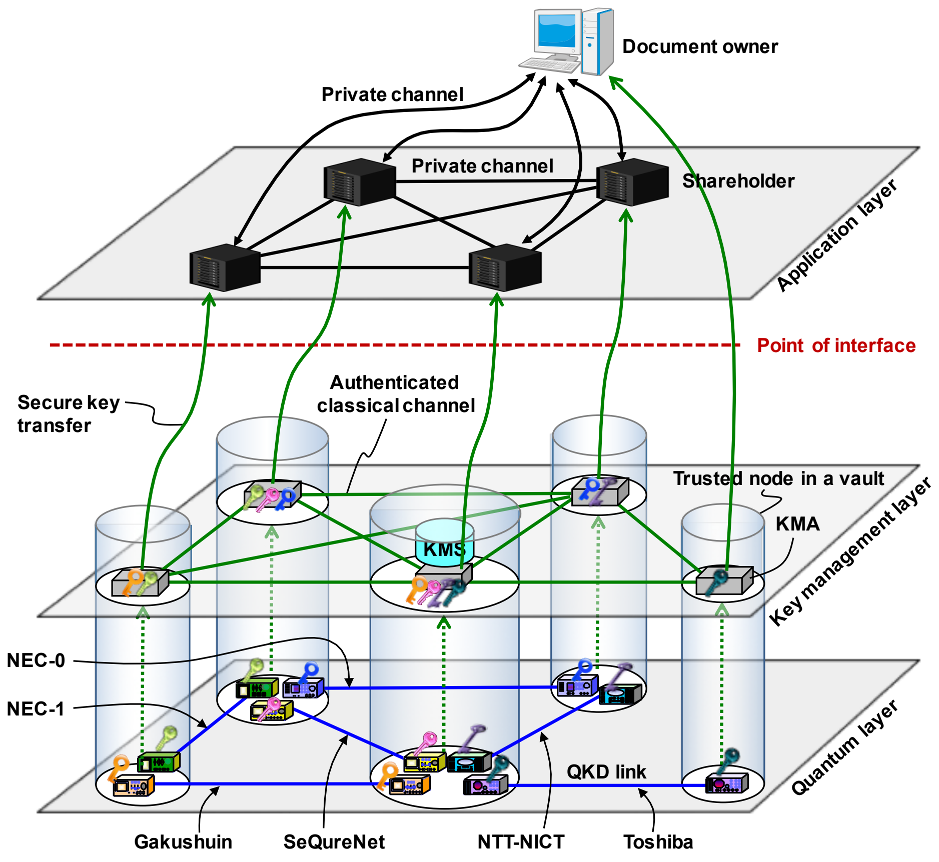

Figure 2: Visualization of the Tokyo QKD Network.

Name	Protocol	Length (km)	Key rate (bit/s)	Loss (dB)
NEC-0	BB84 with decoy	50 (spooled fiber NICT premises)	200k	10
NEC-1	BB84 with decoy	22 (field installed 95% areal line)	200k	13
Toshiba	BB84 with decoy	45 (field installed 55% areal line)	300k	14.5
NTT-NICT	DPS-QKD with decoy	90 (field installed 50% areal line)	10k	28.6
Gakushuin	CV-QKD with decoy	2 (NICT premises)	100k	2
SeQureNet	CV-QKD with decoy	2 (NICT premises)	10k	2

Table 1: Specifications of the QKD links.

cious 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.

6. PERFORMANCE ANALYSIS

Our archiving solution achieves everlasting confidentiality and integrity. Yet, this achievement comes at the cost of (1) considerable storage space requirements due to the use of secret sharing along with the additional data stored for the proof of existence and (2) relatively low data transmission rates due to the use of QKD in comparison to communication speeds achievable over classical network links.

In this section we present an analysis of (1) and (2). We estimate the storage space required by our framework. We also investigate the limits that are caused by using QKD. In our implementation, the QKD links are used to generate and communicate OTP keys before the actual communication starts. Therefore the bandwidth of the QKD channels does not limit the speed of the actual communication but determines the minimum time after which a new communication is possible.

6.1 Storage space requirements

The amount of stored data is analyzed per document d stored. Let $\mathsf{size_d}$ describe the size of d. Once d has been added to the system, data is stored by the shareholders and at the evidence service.

Shareholders.

Each shareholder stores a share s per document. The size of the share is $\mathsf{size_s} = \mathsf{size_d} + \mathsf{size_{meta}}$, where $\mathsf{size_{meta}}$ is the size of the decommitment values accumulated over time. In our solution, $\mathsf{size_{meta}}$ is independent of the document size. The size of a single decommitment value equals the size of the security parameter q_c of the commitment scheme. Currently, a secure instance of the Pedersen commitment scheme results in a decommitment size of 224 bit. The growth of $\mathsf{size_{meta}}$ over time is experimentally analyzed in Section 7 (cf. Figure 3) for a time span of 100 years. The experiments show, that the decommitment values accumulate over this time span to a total of $\mathsf{size_{meta}} \approx 1$ kilobyte.

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. $\mathsf{size_e}$ depends on the size and number of timestamps and commitments that it contains. It grows over time since a new timestamp and a new commitment are added with each renewal. The growth of $\mathsf{size_e}$ over time is experimentally analyzed in Section 7 (cf. Figure 4) for a time span of 100 years. The experiments show, that the size of the evidence record accumulates over 100 years to $\mathsf{size_e} \approx 500$ kilobytes.

Total amount of data stored per document.

Let n be the number of shareholders. Then for each document, n shares have to be stored plus the evidence record. Therefore, the system in total contains data of size: $\mathsf{size_{total}} = n * \mathsf{size_s} + \mathsf{size_e}$. For documents of reasonable size this quantity is dominated by n times the document size.

$\mathsf{size_s}$	**preparation phase storage & retrieval**	**preparation phase share renewal**
1KB	27 milliseconds	53 milliseconds
1MB	4.45 minutes	8.88 minutes
1GB	7.41 hours	14.81 hours

Table 2: Key exchange times for for sharing and resharing $\mathsf{keyRate_{QKD}}$= 300 kilobit/sec.

6.2 Data transmission

We discuss the data transmissions via private channels. Since we use QKD for these channels, the corresponding costs are the limiting factor in our framework. The communication costs for the authenticated (classical) channels are negligible due to available bandwidths. For example, state-of-the-art optical communication channels allow for a throughput of 10 terabyte/sec, while only a few hundred kilobytes of evidence data are sent through these channels.

Data transmission via private channels includes the transfer of shares from the document owner to the shareholders, the retrieval of these shares from shareholders, as well as data transmission among shareholders during share renewal. The communication performance is limited by the previous QKD key exchange rate.

In the following let $\mathsf{keyRate_{QKD}}$ be the key rate of the QKD link between two parties and let $\mathsf{size_s}$ be the size of the shares for document d. Furthermore, let n be the number of shareholders. The QKD best performance of our implementation is 300kbits/sec.

Transfer of data to and retrieval from shareholders.

When the document owner stores data in the archiving system, he sends one share to each shareholder. The same holds for data retrieval but with inverse sending direction. Remember that $\mathsf{size_s} = \mathsf{size_d} + \mathsf{size_{meta}}$. The time required for the initial key distribution between the document owner and one shareholder is $t_s = \mathsf{size_s}/\mathsf{keyRate_{QKD}}$ seconds. Given a QKD network, where the document owner is connected to each shareholder with a separate QKD link, key distribution can be parallelized. Then, t_s is the time required for key exchange prior to communicating one share. If there is not an individual QKD connection between the document owner and each shareholder, key distribution may have to be serialized in the worst case. Then the key exchange time grows to $n * t_s$. In Table 2 we give example values for parallelized key distribution.

Share renewal.

During share renewal, each shareholder sends one share to all other shareholders. If any two shareholders are connected the preparation time for this communication is $2 * t_s$. If these direct connections do not exist, the preparation time for share renewal may be at most $n * (n-1) * t_s$. In Table 2 we also give example values for parallelized key distribution in this case.

Our analysis shows that everlasting confidentiality comes at the cost requiring a distributed storage system with total storage space of a multiple of the original data size. This is due to the use of secret sharing where the total storage space requirements is proportional to the number of shareholders. Therefore, the number of shareholders must be chosen carefully in order to balance security guarantees and storage space requirements. The size of the data stored for the proof of existence is independent of the document size.

Regarding transmission rates, our analysis shows that the realization of an archiving solution is feasible with current QKD technology. Yet, it also shows that a considerable time is required to distribute keys via the QKD links. Parallelization of key distribution is essential to achieve key distribution rates that are optimal for the available QKD technology.

7. EXPERIMENTS

In Section 6 we have presented an analysis of the bottlenecks of our archiving solution: storage space and QKD key exchange time.

In this section, we present experimental results regarding our implementation. They refer to the private QKD channels and to our integrity solution.

QKD performance.

A major part of our experiments was the realization of secret sharing with QKD links between the respective parties. To the best of our knowledge, our implementation happens to be the first QKD-based secret sharing implementation. Our experiments show that such an implementation is possible but there is still room for optimization. The achieved secret key rates of the QKD platform are shown in Table 1 and range from 10 kilobits/sec to 300 kilobits/sec depending on the specification of the respective QKD link. To prevent being limited by the slowest QKD links (10 kilobits/sec), keys are relayed between appropriate KMAs such that OTP keys can be supplied at a reasonable key supply throughput. In our experiment, this allows to raise the minimum throughput of key supply for each private channel to 40 kilobits/sec.

Scenario for the integrity proof evaluation.

We studied how space for the evidence records and verification time for our integrity solution grow over time. We choose the following scenario. We consider an archive that is operated for 100 years starting in 2016 and ending in 2116. In 2016, a document is initially protected by secret sharing as described in Section 3. Also, the initial evidence record is generated. Then both the commitments and the evidence records are renewed as described in our framework.

We choose secure parameters according to the heuristic of Lenstra [23, 1]. The relevant parameters are presented in Table 3 and Table 4. The tables show, which parameter choices are assumed to be secure until which date.

Timestamps are renewed by the evidence service every 2 years. This is a recommended renewal period for digital certificates. We choose this period because timestamps are backed by digital certificates. The commitment renewal period is set to 10 years. This is a conservative choice considering the proposed parameter lifetimes in Table 4, but was chosen to also cover unforeseen developments that may require commitment renewals. The transition to new parameter lengths is always done at the beginning of a new validity period.

Results.

Security	Scheme	Parameters
until 2040	SHA-2 RSA	224 2048
until 2065	SHA-2 RSA	224 3072
until 2085	SHA-2 RSA	256 4096
until 2103	SHA-2 RSA	384 5120
until 2118	SHA-2 RSA	384 6144

Table 3: Timestamping scheme parameters.

Security	Scheme	Parameters
until 2040	SHA-2 Pedersen	224 (2048, 224)
until 2065	SHA-2 Pedersen	224 (3072, 224)
until 2085	SHA-2 Pedersen	256 (4096, 256)
until 2103	SHA-2 Pedersen	384 (5120, 384)
until 2116	SHA-2 Pedersen	384 (6144, 384)

Table 4: Commitment scheme parameters.

In the following, we present the experimental results for our integrity protection solution.

Figure 3 shows the size of the decommitment values that need to be stored by the shareholders in addition to the document d. This value grows over time, as decommitment values are accumulated. Furthermore, the sizes of new decommitment values added to the existing ones grow with the size of the commitment scheme parameters. The total size of the data grows very slowly from 0.07 kilobytes in 2016 to 0.91 kilobytes in 2116. Also recall that the size of the decommitment values is independent of the size of the archived document. In relation to the size of large data such as genome data, these additional data are negligible.

Figure 4 shows the growth of the evidence records that are stored by the evidence service. As new commitments and timestamps are added to the evidence records, they grow over time. In fact, the evidence size grows almost linearly in the number of timestamp renewals. A small superlinear factor is introduced by increasing the parameter sizes from time to time. Notably, the size of the evidence does not depend on the size of the document that is stored, but only on the configuration of the commitment scheme and the timestamping scheme. In our experiments the evidence record grows from 9.59 kilobytes to 471.61 kilobytes within 100 years. Compared to genome data, for example, this is still acceptable.

Finally, in Figure 5 we present timings for the verification of the proofs of existence. They were measured on a machine with an Intel Core i5 2.9Ghz CPU and 8GB of RAM running our Java implementation of the verification algorithm. As new evidence is added to the evidence record over time, the time required for the verification of the evidence record increases. The graph shows that adjusting parameters over time influences the verification performance

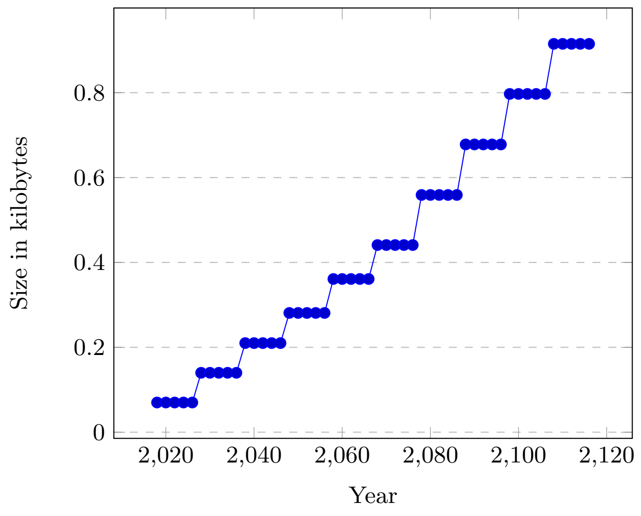

Figure 3: Size of decommitment values stored by the shareholders.

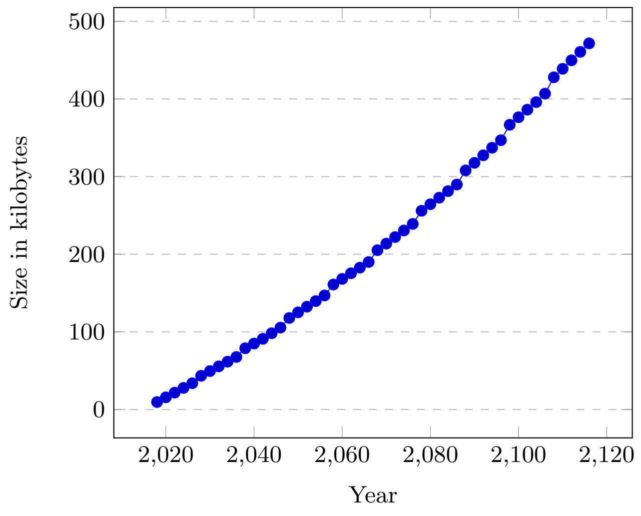

Figure 4: Size of evidence records.

significantly. Nevertheless, the performance of our implementation remains acceptable as verification of the proofs of existence after 100 years takes only approximately 10 seconds.

In summary, the experiments show that the size of all data that need to be stored in addition to an archived document is independent of the document size and grows very slowly. Furthermore, even with today's technology the proof of existence to a document can still be efficiently verified after 100 years of protection.

8. CONCLUSION

We have presented the first digital archiving framework that simultaneously provides everlasting confidentiality of the archived documents and maintains a proof that the documents existed at archiving time and were not changed since. Our solution has three main ingredients: QKD-based private channels which allow for unconditional confidentiality of data in transit, proactive secret sharing which provides

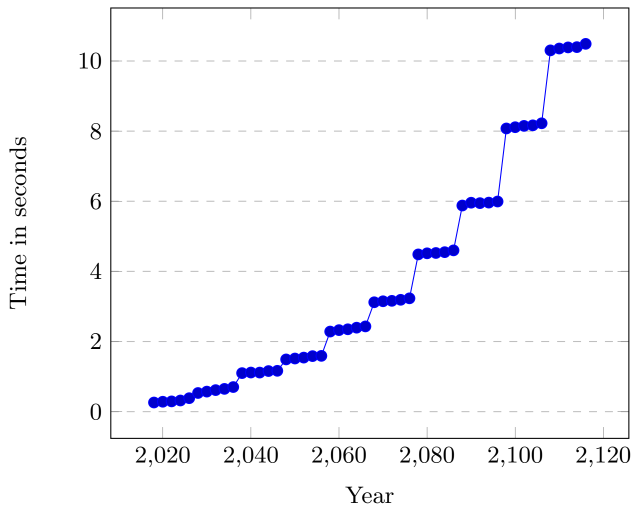

Figure 5: Time required for proof of existence verification.

unconditional confidentiality of data at rest, and chains of timestamped unconditionally hiding commitments which enable long-term proofs of existence without compromising confidentiality. We have also presented a proof of concept implementation of our framework which combines the technically very advanced NICT QKD platform with state of the art implementations of the cryptographic components. Experimental results show that our solution is promising. However, there are still a number of issues that we will address in the future.

The first challenge is to optimize the performance of our solution. For example, we plan to modify the QKD platform such that it does not require serialization of private channel communications but supports multiple private channel communications in parallel. The next step is to include proactive secret sharing and to extend the experiments accordingly. Another challenge will be to also include the possibility of carrying out computations on the shared data. This is in principle possible since secret sharing schemes typically have certain homomorphic properties. In regard to our integrity solution, it is our goal to come up with a solution that does not require document reconstruction when the proof of existence is renewed. Also, our integrity solution relies on cryptographic insecurities being predictable. We will also study the question how an unexpected break of cryptography can be dealt with.

9. REFERENCES

[1] Cryptographic key length recommendation. https://www.keylength.com. Accessed: May 2016.

[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] M. Bellare and P. Rogaway. *Advances in Cryptology — CRYPTO' 93: 13th Annual International Cryptology Conference Santa Barbara, California, USA August 22–26, 1993 Proceedings*, chapter Entity Authentication and Key Distribution, pages 232–249. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.

[4] D. J. Bernstein, J. A. Buchmann, and E. Dahmen. *Post-Quantum Cryptography*. Springer Berlin Heidelberg, 2009.

[5] A. J. Blazic, S. Saljic, and T. Gondrom. Extensible Markup Language Evidence Record Syntax (XMLERS). RFC 6283 (Proposed Standard), July 2011.

[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] J. A. Buchmann. *Introduction to Cryptography*. Springer New York, 2004.

[8] R. Canetti, L. Cheung, D. K. Kaynar, N. A. Lynch, and O. Pereira. Modeling computational security in long-lived systems. In *CONCUR 2008 - Concurrency Theory, 19th International Conference, CONCUR 2008, Toronto, Canada, August 19-22, 2008. Proceedings*, pages 114–130, 2008.

[9] R. Canetti and A. Herzberg. *Advances in Cryptology — CRYPTO '94: 14th Annual International Cryptology Conference Santa Barbara, California, USA August 21–25, 1994 Proceedings*, chapter Maintaining Security in the Presence of Transient Faults, pages 425–438. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.

[10] Y. Desmedt and S. Jajodia. Redistributing secret shares to new access structures and its applications, 1997. Technical Report ISSE TR-97-01, George Mason University.

[11] 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.

[12] ETSI. *CMS Advanced Electronic Signatures (CAdES)*. Number TS 101 733. 1.7.4 edition, Jul. 2010.

[13] ETSI. *XML Advanced Electronic Signatures (XAdES)*. Number TS 101 903. 1.8.3 edition, jan 2010.

[14] N. Gisin, G. Ribordy, W. Tittel, and H. Zbinden. Quantum cryptography. *Rev. Mod. Phys.*, 74:145–195, Mar 2002.

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

[16] 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.

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

[18] V. H. Gupta and K. Gopinath. An extended verifiable secret redistribution protocol for archival systems. In *ARES*, pages 100–107, 2006.

[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 P. Kamat. A content integrity service for long-term digital archives. In *Archiving 2006*, pages 159–164. Society for Imaging Science and Technology, 2006.

[21] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. *Advances in Cryptology — CRYPTO '95: 15th Annual International Cryptology Conference Santa Barbara, California, USA, August 27–31, 1995 Proceedings*, 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] A. K. Lenstra. Key lengths. In *The Handbook of Information Security*. Wiley, 2004.

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

[25] 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.

[26] T. P. Pedersen. *Advances in Cryptology — CRYPTO '91: Proceedings*, chapter Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing, pages 129–140. Springer Berlin Heidelberg, Berlin, Heidelberg, 1992.

[27] 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.

[28] 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.

[29] M. Sasaki, M. Fujiwara, H. Ishizuka, W. Klaus, K. Wakui, M. Takeoka, S. Miki, T. Yamashita, Z. Wang, A. Tanaka, K. Yoshino, Y. Nambu, S. Takahashi, A. Tajima, A. Tomita, T. Domeki, T. Hasegawa, Y. Sakai, H. Kobayashi, T. Asai, K. Shimizu, T. Tokura, T. Tsurumaru, M. Matsui, T. Honjo, K. Tamaki, H. Takesue, Y. Tokura, J. F. Dynes, A. R. Dixon, A. W. Sharpe, Z. L. Yuan, A. J. Shields, S. Uchikoga, M. Legré, S. Robyr, P. Trinkler, L. Monat, J.-B. Page, G. Ribordy, A. Poppe, A. Allacher, O. Maurhart, T. Länger, M. Peev, and A. Zeilinger. Field test of quantum key distribution in the Tokyo QKD Network. *Opt. Express*, 19(11):10387–10409, May 2011.

[30] 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.

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

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

[33] P. W. Shor and J. Preskill. Simple proof of security of the BB84 quantum key distribution protocol. *Physical*

review letters, 85(2):441, 2000.

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

[35] 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.

[36] T. M. Wong, C. Wang, and J. M. Wing. Verifiable secret redistribution for archive system. In *IEEE Security in Storage Workshop*, pages 94–106, 2002.