Witold MAĆKÓW, **Jerzy PEJAŚ**
WEST POMERANIAN UNIVERSITY OF TECHNOLOGY, FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY,
49 Żołnierska St., 71-200 Szczecin

Secure Archive for Long-Term Electronic Document Storage with Provable Authenticity

PhD eng. Witold MAĆKÓW

Witold Maćków (1974) received M.Sc. title in 1998 and PhD title in 2007 from Faculty of Computer Science and Information Technology, Technical University of Szczecin (at present West Pomeranian University of Technology, Szczecin). His scientific interests include public key infrastructure application, long term archive systems, authenticated data structures, threshold secret sharing and still image steganography.

e-mail: wmackow@wi.zut.edu.pl

PhD eng. Jerzy PEJAŚ

He received M.Sc. degree in Computer Science and Engineering from the Wrocław University of Technology and PhD degree in Control Systems from Gdansk University of Technology. Main subjects of interest: information and computer network security, methods of secure electronic signatures as well as new trends in applied cryptography. Employed as Assistant Professor at the Faculty of Computer Science and Information Technology, West Pomeranian University of Technology in Szczecin.

e-mail: jpejas@wi.zut.edu.pl

Abstract

In this paper there is presented an archive system architecture designed for retaining encrypted electronic documents and ensuring their availability and provable authenticity over long periods. The general approach focuses on eliminating single points of trust and single points of failure. The elimination of single points of trust for electronically stored documents allows increasing their legal effectiveness and admissibility as an evidence in legal proceedings. On the other hand, the information splitting among many system nodes protects against information loss and allows recovering it in the event of failure. In order to achieve these features, there were used many mechanisms, including secret sharing, methods for shares distribution and redistribution, archive timestamps and methods of their renewal in the case when the shares, encryption keys and hash values were also renewed.

Keywords: archive system, fault tolerant system, secret sharing, long-term preservation, integrity and authenticity, evidence, data recovering.

Bezpieczne archiwum do długotrwałego przechowywania dokumentów elektronicznych z dowodliwą autentycznością

Streszczenie

W tym artykule przedstawiono architekturę systemu archiwalnego przeznaczonego do przechowywania zaszyfrowanych dokumentów elektronicznych i zapewnienia ich dostępności oraz dowodliwej autentyczności przez długi okres czasu. Ogólne podejście polega na wyeliminowaniu pojedynczych punktów zaufania i pojedynczych punktów awarii. Wyeliminowanie pojedynczych punktów zaufania do dokumentów przechowywanych elektronicznie pozwala na zwiększenie ich wartości prawnej oraz na dopuszczanie ich jako dowodu w postępowaniu sądowym. Z drugiej strony, podział informacji pomiędzy wiele węzłów systemu chroni przed utratą informacji i pozwala na ich odtworzenie w przypadku awarii. W celu osiągnięcia tych funkcji zastosowano wiele mechanizmów, w tym schematy podziału sekretów, metody dystrybucji udziałów i ich redystrybucji, znaczniki czasu archiwum oraz metody odnawiania w przypadku, gdy niezbędne jest również odnowienie udziałów, kluczy szyfrowania i wartości skrótów kryptograficznych.

Słowa kluczowe: system archiwalny, system odporny na błędy, podział sekretu, ochrona długoterminowa, integralność i autentyczność, poświadczenie, odtwarzanie danych.

1. Introduction

Electronic documents do not have the same longevity properties as physical documents. The proper electronic document preservation raises many problems that still remain unsolved. In this paper the authors propose a multi-server based architecture of a long-term archive system. The primary goal of an archive system is to preserve the long-term availability, confidentiality and evidences validity of the existence of documents, or assertions of agreements that were originally asserted with digital signatures, even in the case of storage server failures and compromises.

Long-term archive services may support a range of applications, including wills, land records, medical data, criminal case files, personnel files and contracts. They may be used by any type of entity, e.g. organizations, citizens, notaries [1].

T. M. Wong, et.al. [2] presents an archival storage system that consists of two components: clients and the group of servers. The specially prepared pieces of the electronic document are stored on each server (shareholder) belonging to the group. These pieces, representing the shares of each server, can be used to restore the electronic document to its original form. The document splitting and its recovering are based on verifiable secret redistribution protocol for secrets distributed with the Shamir's threshold sharing scheme [3]. The main goal of this protocol is to:

- preserve a minimum level of fault-tolerance over the long term period,
- allow producing new shares when shareholders become unavailable (permanently or temporarily) or new shareholders join the system.

It is true that due to the above mentioned solution the electronic document remains confidential for a long period. However, the direct usage of the protocol proposed in [2] for the electronic document splitting has many disadvantages. Among them, the most important are as follows:

- shares creation for large-size electronic documents is time consuming,
- the shares redistribution process needs even more time, particularly when the redistribution concerns all shares,
- there is the possibility of modification of chosen shares of the electronic document; such a modification can be made by a document owner in cooperation with a group of servers for example,
- due to the above mentioned possibility, and used share redistribution mechanism as well, the document legal effectiveness and admissibility as evidence in legal proceedings can be denied after its recovering; the parties of legal action can call into question, for example, the weakness of secret sharing scheme used many years ago.

The impossibility to guarantee a non-repudiable (the case of authenticity of the origin and integrity) proof of the electronic document existence throughout the whole storage period is another important disadvantage of the solution presented in [2]. This is particularly essential for signed electronic documents (see e.g. [5]). Furthermore, the creation of document shares, its distribution to distinct remote locations, the share redistributions, etc., can prevent a verifier from correct verification of a signature or signatures validity related to the document to be recovered. Such a situation may arise as a result of very long archival period (even 30 years or more), during which hash and public key algorithms can become weak or certificates can become invalid.

To avoid the above listed problems, two essential requirements have been stated in the draft version of RFC [6]:
- a long-term archive service must be capable to provide evidence that can be used to demonstrate the integrity of data for which it is responsible from the time it received the data until the expiration of the archival period of the data,
- a long-term archive service must provide means for accepting encrypted data in such a manner that future preservation activities apply to the original, unencrypted data.

It is very difficult to meet these two requirements when the threshold sharing scheme is used for archival data only. However, the secret sharing scheme is very useful when it is used together with redistribution and validity verification of recovered shares mechanisms (like those presented below).

2. Our contributions

In this paper we describe how to modify the prior verifiable secret redistribution protocol for archive system [2, 3, 4, 8-11]. We explain also how the publication service (provided by the user or shareholders) can make highly available the current, authenticable validity status of each shares issued by one of a large number of shareholders. Moreover, every shareholder can check not only whether she or he is in possession of proper share, but whether this share will allow to restore original value of secret (or secrets) in access structure $\Gamma_P^{(m,n)}$ as well (see Section 4.1). We present also our new verifiable secret redistribution protocol for the redistribution of secrets from this access structure to any other access structure $\Gamma_{P'}^{(m',n')}$. Our protocol is similar to the one presented in [2], but it contains another mechanism of verification. This mechanism allows every shareholder to carry out the proof of correctness for shares restored in such a recreated access structure (without revealing of secrets).

According to our proposal - an archive document is not split into shares. Instead of this, the document is encrypted twice, and then shares are generated for encrypting keys. It allows sending a document to an archive in a secure way and to renew keys if cipher algorithms used for document encryption or secret sharing become weak or if the validity period of the time stamping authority (TSA) certificate expires or is revoked.

3. New proposal of long-term archive system

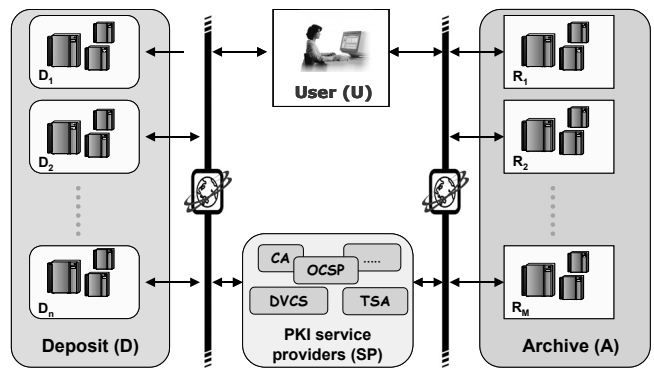
Archive **A** consists of *M* repositories $\mathbf{R_i}$ (see Fig.1). The number of repositories may vary. Each repository is an independent archive element, which may cooperate with any PKI (Public Key Infrastructure, for more see [14]) service provider **SP**. User **U** is a client of an archive **A**, requesting to store digitally signed document *C* in an archive. The processing of such a request is rather standard and fulfilled in the following way:
- the user verifies digital signature on a document, time stamps this signature and transforms it into archival form *ES-A* (e.g. [6, 7]);
- the user selects two different symmetric encryption algorithms $E_{k_1}^1(\ldots)$ and $E_{k_2}^2(\ldots)$ and then generates proper keys for these algorithms - k_1 and k_2 respectively;
- the digital document and its signature (archival form) are doubly encrypted together $ED = E_{k_2}^2\left(E_{k_1}^1(C, ES-A)\right)$;
- the obtained cryptogram *ED* is transmitted to each repository.

Each repository receives encrypted documents and creates archive time stamps for them [1]. That archive timestamps directly protect the integrity of the bit-stream and freezes the bit structure at the time of archiving.

Keys k_1 and k_2 are divided into *n* shares according with the Shamir's threshold sharing scheme [3]. The generated shares are sent to the depository **D**. The depository consists of *n* nodes, further called shareholders $\mathbf{D_i}$. Every share is stored by a separated shareholder.

Shareholders are responsible for a confidential storage of deposits. Additionally, they are obligated to redistribute deposits in the following cases:
- revealing of any share stored in nodes $\mathbf{D_i}$;
- losing of shares because of a damage of any node;
- increasing of nodes.

The process of share redistribution may be initialized by user **U**, for example when more resistant algorithms or keys should be used. In such a case user **U** receives encrypted document from archive and required number of shares from depository. User reconstructs encryption keys using received shares and then applies them for document decryption. The whole commission process, necessary to store digitally signed document **C** in an archive, should be repeated. Additionally user may subject decrypted document to the procedure of extending of validity period of digital signature (e.g. [5]).



Rys. 1. Ogólna architektura prezentowanego archiwum
Fig. 1. General architecture of the presented archive

4. Publicly verifiable secret sharing protocol

In the following section a publicly verifiable secret sharing protocol will be introduced. This protocol is used for distribution and redistribution of shares. It consists of two basic protocols: a distribution protocol, where the secret is distributed by the user **U** among shareholders, and a reconstruction protocol, where the secret is recovered by pooling the shares of the qualified subset of the shareholders.

Basic threshold secret sharing schemes solve the problem under the assumption that all players in the scheme are honest. A bit more sophisticated solutions avoid this assumption - Feldman [8], Pedersen [12], Stadler [13], Schoenmakers [10] design two new classes of secret sharing schemes: verifiable secret sharing (VSS) schemes and publicly verifiable secret sharing (PVSS) schemes.

Our new secret sharing scheme belongs to the group of PVSS schemes and similarly to the scheme proposed in [11] has two advantages: shareholders can verify not only their own share, but also verify whether other shareholders received the correct shares, shareholders simply release their shares in the reconstruction protocol, subsequently the released shares may be verified by anybody against the output of the distribution protocol.

4.1. Assumptions for the PVSS scheme

Let G_q denote a group of prime order *q* and let *g*, $h \in Z_p$ (*p* mean large prime such that *p-1* is divided by prime *q*) denote independently selected elements of order *q*.

Additionally, we denote a secure hash function by $H(\ldots)$. Let us assume also that user **U** knows secrets k_1, $k_2 \in G_q$ and wants to distribute them among *n* parties indexed from a set $P = \{1, 2, \ldots, n\}$ ($|P| = n$), such that any *m* of the shareholders can find k_1 and k_2 if necessary, but less than *m* shareholders get no information about

k_1 and k_2. We say that authorized subset $B \in PS(P)$[1], $|B|=m$ forms access structure $\Gamma_P^{(m,n)}$ of some (m, n) threshold scheme.

4.2. Distribution protocol

To distribute k_1 and k_2 to the access structure $\Gamma_P^{(m,n)}$, the user **U** selects two random polynomials $f_1(x)$ and $f_2(x)$ of m-1 degree with coefficients in G_q:

$$f_1(x) = \sum_{j=0}^{m-1} \alpha_j x^j \quad (1)$$

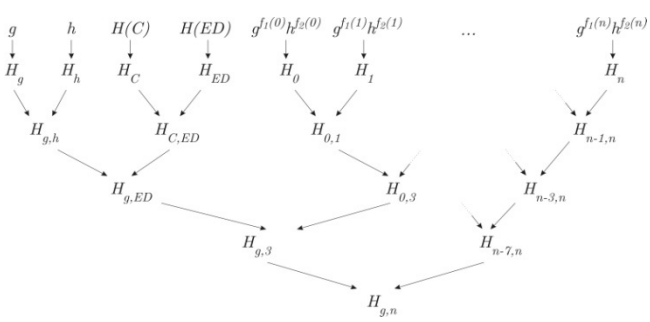
$$f_2(x) = \sum_{j=0}^{m-1} \beta_j x^j \quad (2)$$

and sets $\alpha_0 = k_1$, $\beta_0 = k_2$.

Polynomials $f_1(x)$ and $f_2(x)$ are used for share generation, i.e. the values $f_1(i)$ and $f_2(i)$ for every $i \in P$. These shares are sent to the corresponding $i \in P$ over private channels. The user keeps polynomials $f_1(x)$ and $f_2(x)$ secret, but for every $i \in P$ creates the evidences over Z_p for keys $f_1(0)=k_1$, $f_2(0)=k_2$ and all shares $f_1(i)$ and $f_2(i)$ ($i=1, ..., n$):

$$g^{f_1(0)}h^{f_2(0)}, g^{f_1(1)}h^{f_2(1)}, \ldots, g^{f_1(n)}h^{f_2(n)} \quad (3)$$

Next, the user **U** creates the proof of share validity and their relations to the digital document C on the base of generated evidences (this can be made in parallel with share generations, see **Algorithm 1**). Authenticated dictionaries may be used for this purpose, for example Merkle tree scheme (see Fig.2). Leaves of such a tree contain hash values for concatenation of $g^{f_1(i)}h^{f_2(i)}$ (for $i=0, \ldots, n$) with contents of a plain document and contents of an encrypted document. Root hash is digitally signed by a user, and the signature is time stamped by TSA. Initial archive timestamp T_{U0}, together with the user's **U** signature, gives a function of authenticity proof for key shares, and they are designated as $Sig_U(H_{g,n}, T_{U0})$.



Rys. 2. Publicznie weryfikowalne poświadczenia dla schematu podziału progowego Shamira (m,n) opartego na drzewie Merkla

Fig. 2. Publicly verifiable evidences for Shamir's threshold sharing scheme (m,n), based on Merkle tree

Algorithm 1: Generation of shares and their publicly verifiable evidences

Input: in=[$\Gamma_P^{(m,n)}$, k_1, k_2, H(ED), H(C), p, q, $f_1(x)$, $f_2(x)$]

1. $\Gamma_P^{(m,n)}$ - access structure, k_1, $k_2 \in G_q$ - keys
2. H(ED) - hash value of double cryptogram
3. H(C) - hash value of signed document **C**
4. p and q - two large primes such that q|(p-1)
5. two random polynomials $f_1(x)$ and $f_2(x)$ each of m-1 degree with coefficients in G_q, *i.e.* $f_1(x) = \sum_{j=0}^{m-1} \alpha_j x^j \pmod q$ and $f_2(x) = \sum_{j=0}^{m-1} \beta_j x^j \pmod q$ *where* $\alpha_0 = k_1$, $\beta_0 = k_2$

Output: out=[f_1, f_2, evd, sig]

1. f_1, f_2 - vectors of shares
2. evd - a vector of verifiable evidences
3. sig=$Sig_U(H_{g,n}, T_{U0})$ - time stamped signature of the authenticated dictionary root hash

```
1 begin
2   forall i ← 1, 2, 3, ... , n do
3     f1[i] ← = α0 + α1x + … + αm-1x^m-1 mod q
4     f2[i] ← = β0 + β1x + … + βm-1x^m-1 mod q
5     evd[i] ← g^f1(i) h^f2(i) mod p
6   endforall
7   Hg,n ← rootOfMerkleTree( f1, f2, evd, g, h,
        H(C), H(ED))
8   sig ← SignAndTimestamp(Hg,n)
9 end
```

After building and signing the tree of proof (see procedure *rootOfMerkleTree* and *SignAndTimestamp* of **Algorithm 1**, respectively), a user **U** sends $Sig_U(H_{g,n}, T_{U0})$ together with values of leaves of this tree to all $i \in P$ over the broadcast channel. This tree is additionally published and may be accessed by any shareholder.

Receiving data from a user **U**, every shareholder P_i verifies his share according to the following procedure:

- for shares $f_1(i)$ and $f_2(i)$ he calculates value $(g)^{f_1(i)}(h)^{f_2(i)}$ and compares it with the one received from user **U**; if values are different he notifies user **U** about it;
- for calculated value $(g)^{f_1(i)}(h)^{f_2(i)}$ he recovers the Merkle hash tree (of Fig.2), calculates root hash and verifies its authenticity,
- he chooses any authorized subset $B \in PS(P)$ and verifies that:

$$g^{f_1(0)}h^{f_2(0)} \equiv \prod_{i \in B} \left(g^{f_1(i)}h^{f_2(i)}\right)^{\lambda_i} \text{ where } \lambda_i = \prod_{j \in B, j \neq i} \frac{j}{j-i} \quad (4)$$

If the conditions hold, a share holder broadcasts a "commit" message; otherwise, broadcasts an "abort" message.

The last test may be repeated for others authorized subsets B, until a shareholder assures he possesses the share really related to the secrets k_1 and k_2.

Remarks. The correctness of equation (4) can be simply proved using the following calculations:

$$\prod_{i \in B} \left(g^{f_1(i)}h^{f_2(i)}\right)^{\lambda_i} = (g)^{\sum_{i \in B} f_1(i)\lambda_i}(h)^{\sum_{i \in B} f_2(i)\lambda_i} = g^{f_1(0)}h^{f_2(0)} = g^{k_1}h^{k_2} \quad (5)$$

4.3. Redistribution protocol

Secrets stored in depository **D** nodes are exposed to attacks and as a result several faults might occur:

- secrets can be revealed,
- shares can gradually be corrupted/compromised,
- hardware can be failure or damaged.

The goal of the redistribution protocol is to create the ability to renew secret shares. This process does not require the intermediate reconstruction of the secret; we adapt Desmedt and Jajodia protocol [2, 9].

Shares redistribution involves modification of an access structure $\Gamma_P^{(m,n)}$, assuming that threshold value m and number of shares n may undergo a change. Let $\Gamma_{P'}^{(m',n')}$ denote so modified access structure. Then the share redistribution protocol involves proper conversion from n shares of secrets k_1 and k_2 of access structure $\Gamma_P^{(m,n)}$ into n' shares of access structure $\Gamma_{P'}^{(m',n')}$. Let us choose any authorized subset $B \in PS(P)$. Every shareholder belonging to that subset selects two random polynomials $f_{i_s 1}(x)$

[1] $PS(X)$ means a power set of X and is the set of all subsets of X

and $f_{i_s 2}(x)$ (for every $i_s \in B$) of m'-1 degree with coefficients in G_q:

$$f_{i_s 1}(x) = \sum_{j=0}^{m'-1} \alpha_{i_s j} x^j \tag{6}$$

$$f_{i_s 2}(x) = \sum_{j=0}^{m'-1} \beta_{i_s j} x^j \tag{7}$$

and sets $\alpha_{i_s 0} = f_1(i_s)$, $\beta_{i_s 0} = f_2(i_s)$, where $f_1(i_s)$ and $f_2(i_s)$ – shares of the shareholder number i_s from the access structure $\Gamma_P^{(m,n)}$ (see Section 4.2).

Then every shareholder $i_s \in B$ acts similarly as in the case of secret distribution (Subsection 4.2) and performs steps according to **Algorithm 2**. The shareholder $i_s \in B$ generates shares $f_{i_s 1}(j)$ and $f_{i_s 2}(j)$ for every $j \in P'$ and sends them to the corresponding shareholder $j \in P'$ over private channels. Then the shareholder $i_s \in B$ for every $j \in P'$ creates the evidences to $f_{i_s 1}(0) = f_1(i_s)$ and $f_{i_s 2}(0) = f_2(i_s)$, and all shares $f_{i_s 1}(j)$ and $f_{i_s 2}(j)$ ($j=1, \ldots, n'$):

$$g^{f_{i_s 1}(0)} h^{f_{i_s 2}(0)}, g^{f_{i_s 1}(1)} h^{f_{i_s 2}(1)}, \ldots, g^{f_{i_s 1}(n')} h^{f_{i_s 2}(n')} \tag{8}$$

In the same manner as in Subsection 4.2, the above mentioned evidences are used by the shareholder $i_s \in B$ to construct the authenticated dictionary and to create the proof of authenticity for newly generated shares: $Sig_{i_s}\left(H_{g,n}^{i_s}, T_{i_s 0}\right)$. Next, the evidence and the authenticated dictionary components are published and sent to all other (n'-1) shareholders.

Algorithm 2: Shares redistribution of shareholder number $i \in B$ with publicly verifiable evidences

Input: in=[$\Gamma_{P'}^{(m',n')}$, k_1, k_2, H(ED), H(C), p, q, $f_1(x)$, $f_2(x)$]

1. $\Gamma_{P'}^{(m',n')}$ – a new access structure,
2. $f_1[i]$, $f_2[i]$ – shares of shareholder number $i \in B$
3. H(ED) – hash value of double cryptogram
4. H(C) – hash value of signed document **C**
5. p and q – two large primes such that q|(p-1)
6. two random polynomials $f_{i1}(i)$ and $f_{i2}(i)$ each of m′-1 degree with coefficients in G_q, *i.e.*
   $f_{i1}(i) = \sum_{j=0}^{m'-1} \alpha_{ij} i^j \pmod q$ and $f_{i2}(i) = \sum_{j=0}^{m'-1} \beta_{ij} i^j \pmod q$
   where α_{i0} = $f_1[i]$, β_{i0} = $f_2[i]$

Output: out=[f_{i1}, f_{i2}, evd, sig]
1. f_1, f_2 – vectors of shares
2. evd_i – a vector of verifiable evidences
3. $sig_i = Sig_U(H_{g,n'}^i, T_{i0})$ – time stamped digital signature of the root hash

```
1 begin
2   [fi1,fi2,evdi, sigi] = call Algorithm 1 with
       [Γ_P'^(m',n'), f1[i], f2[i], H(ED), H(C), p, q,
        fi1(i), fi2(i)]
3   forall j=0, 1, ..., n′ do
4     if i ≠ j then
5       sendSecure(fi1[j], fi2[j], j)
        /* over the private channel */
6       sendAndPublish(evdi, sigi, j)
        /* over the broadcast channel */
7     endif
8   endforall
9 end
```

After the completion of all shares and proofs every shareholder $i \in P'$ checks their authenticity (see Subsection 4.2), and then reconstructs his share using an equation (9) below.

$$\begin{aligned} s_{i1} &= \sum_{i_s \in B} \lambda_{i_s} f_{i_s 1}(i) \\ s_{i2} &= \sum_{i_s \in B} \lambda_{i_s} f_{i_s 2}(i) \end{aligned} \quad , \quad \lambda_{i_s} = \prod_{j \in B, j \neq i_s} \frac{j}{j - i_s} \tag{9}$$

Moreover, every shareholder $i \in P'$ can check for any authorized subsets $B' \in PS(P')$, $|B'| = m'$, $i \in B'$, whether the reconstructed share enables recovering primary keys k_1 and k_2:

$$g^{k_1} h^{k_2} \equiv \prod_{j_s \in B'} \left(g^{s_{j_s 1}} h^{s_{j_s 2}} \right)^{\lambda_{j_s}} \quad , \quad \lambda_{j_s} = \prod_{x \in B', x \neq j_s} \frac{x}{x - j_s} \tag{10}$$

The values $g^{s_{j_s}} h^{s_{j_s 2}}$ (for $j_s \neq i$) are calculated according to equation (10) on the basis of values publicly obtainable and defined in equation (7).

$$g^{s_{j_s 1}} h^{s_{j_s 2}} \equiv \prod_{i_s \in B} \left(g^{f_{i_s 1}(j_s)} h^{f_{i_s 2}(j_s)} \right)^{\lambda_{i_s}} \quad , \quad \lambda_{i_s} = \prod_{j \in B, j \neq i_s} \frac{j}{j - i_s} \tag{11}$$

The reconstructed and verified shares are sent in the form $g^{s_{j1}} h^{s_{j2}}$ to the user **U** (for every $j \in B'$). The user **U** checks whether (10) is fulfilled. If so, then the authenticated directory is updated by linking it to the previously created directory (that type of linkage is presented in [15, 16], for example). An updated evidence of authenticity for every share $Sig_U\left(H_{g,n}^t, T_{Ut}\right)$, where t denotes the sequence number of the proof, is sent over the broadcast channels to all other shareholders. Every shareholder, using the proofs received, can check an authenticity of the share owned once again (it is obvious that the step should be made after an authenticated directory has been updated using an equation (10)).

5. Retaining archive-stored document authenticity

Unlike paper documents, authenticity of signed and/or encrypted electronic documents (signed or encrypted) can become worse due to the time elapsed. Particularly, the reasons are as follows:

- encryption algorithms and cryptographic keys used for an electronic signature creation can become weak and not sufficiently resistant in the case of different attacks,
- information necessary for an electronic signature verification and validation is not accessible (i.e. the lack of access to CA certificates, CRLs, electronic documents, etc.).

After many years facts stated above could make verifying the formerly created electronic signatures impossible, which could result in the lack of an electronic document authenticity.

It is required, in order to ensure the document authenticity, to transform all electronic signatures to the archival form before any encryption of the document by the user **U**. An obtained archival form [6, 7] includes all necessary information required for the signature verification and the so called "initial time stamp" for an archival form, which links an archival form hash value to the trusted time.

When an encrypted document is sent to the archive (Fig. 1), every node (from M entities) creates an initial time stamp value for it. Time stamps for every encrypted document are created independently. A time stamp can become invalid when a public key algorithm or a hash function respectively:

- will be or could be broken, or
- public key certificate of a time stamping authority will be overdue soon or is revoked.

Both stated above circumstances require returning the trust to existing time stamps, and it could be done by means of new time

stamping of an encrypted document form. Two types of time stamp renewal for encrypted documents are used[2] [17, 18]:

- time stamp renewal: the new time stamp for an encrypted document is created, related to the previous one; subsequent time stamps obtained in this renewal procedure form the chain of time stamps for a data object or a group of data objects;
- hash value renewal: the new time stamp for an archival form is created; it is related to previous time stamps and to data objects stamped with initial time stamp for an archival form as well; the new set of linked time stamps is created; one or more linked time stamp sets form the sequence of encrypted form time stamps.

Any subsequent renewal of time stamps or hash values can be performed independently from any stored encrypted documents or the group of those documents. The last approach requires only one renewed time stamp creation (it is also the case of renewed hash values). Time stamps and hash values renewal should be performed by the user **U** as well. It can be arranged periodically or when encryption keys are renewed.

6. Evidences renewal

Evidences stored by the user **U**, at every deposit server **D** and in every archive **A**, have to provide proofs that can be used to demonstrate the integrity of data which it is responsible for; it is obligatory from the time the data are received until the expiration of the archival period.

The following threads should be considered in the context of stored evidences security:

- each time stamp token is electronically signed by a time stamping authority itself; when the verification of that signature is performed after TSA's signing certificate validity period, then the result is always negative,
- hash functions, symmetric cryptographic algorithms or threshold sharing schemes used can be become weak,
- any deposit servers or archive servers can fail.

The renewal of the timestamps is the mean enabling to avoid those threads; this should be performed by the user **U**, and every depository and repository. There are four distinguished cases when the evidences renewal is necessary, i.e.: time stamps renewal, hash values renewal, secret shares renewal, encryption keys and encrypted documents renewal.

6.1. Time stamps renewal

If the expiration time of the timestamp authority (TSA) signing certificate falls before the end of the electronic document required storage time, the timestamp applied to the invoice must be re-time stamped before the expiration of the certificate used to verification of the previous timestamp, and so on. It means, that a new timestamp is generated, which covers the timestamp of the old one.

6.2. Hash values renewal

A new timestamp which covers all the old timestamps as well as the electronic documents or other evidences is generated. This procedure is required always when the hash algorithm used to build the new hash trees in the archive timestamp loses its security properties.

6.3. Secret shares renewal

Shares renewal is carried out in the case when the number of deposit nodes (shareholders) changes (i.e.: due to the failures), threshold sharing scheme parameters are revealed (i.e.: random coefficients of the Shamir's threshold secret sharing scheme polynomials) or the scheme is cryptographically broken. For first two cases it is enough to apply a secret redistribution protocol proposed in Subsection 3.3. The third case requires choosing another stronger threshold scheme, generating new shares and distributing them according with a new used scheme. Generally, such an attempt could be difficult to realize on the basis of the proposed shares distribution and redistribution schemes.

6.4. Encryption keys and encrypted documents renewal

At any moment the user **U** or another part of the system (e.g.: any archive node) can consider an encryption algorithm too weak to ensure confidentiality of electronic documents copies, or due to another reason will enforce the key replacement for the same algorithm. In both cases it is necessary to generate a new encryption key, to encrypt an electronic document and to refresh shares (see Section 3). The renewal of encryption keys and encrypted documents should be performed by the user directly and requires to get the document copy from the repository, to collect shares from the depository, to recover appropriate keys and to decrypt the copy. A decrypted document is subjected to the procedure of an electronic signature validity renewal (see Section 4 and also [5, 6, 7]), then it is re-encrypted with the new key and sent to every archive repository. Encryption keys are shared and transferred to selected number of depository nodes according to the distribution protocol defined in Subsection 3.2. The proof of authenticity $Sig_U\left(H^t_{g,n}, T_{Ut}\right)$ is created for a new cryptogram, the same is done in the case of new shares; that proof has to be linked to previous shares validity evidences.

7. Summary

The architecture for the storage of encrypted electronic documents is presented. Documents are stored in M different archive nodes (repositories). Encryption keys, after their splitting into shares according to the Shamir's threshold sharing scheme (m, n), are stored in n different depositories. In the case of any archive node failure, or when the additional system node is required, the system allows getting any copy of an encrypted document from active archive nodes and recovering the lost copy or creating the new one. The double encryption mechanism, proposed by the authors, enables avoiding cryptographic channels creation at the link between selected archive nodes.

Shares of encryption keys stored in depository nodes are damage resistant as well. In the case of any failure (in maximum n-m nodes) the system is able to recover lost shares; for that purpose the share redistribution protocol presented in Subsection 3.3 is used. The same protocol can be used when new nodes are added to the depository.

Distribution and redistribution of secret shares is provided according to the new verifiable threshold sharing scheme. Proofs of authenticity are created on the base of authenticated directories. They are created by the users and depository nodes, and then published. The contents of such a directory includes the complete trusted history of the share storage, their redistribution history, and the history of encryption key renewals.

The fundamental target of the proponed archive is to ensure the authenticity of stored documents. Time stamps renewal techniques presented in Section 4 enable verifying the authenticity of encrypted documents; moreover, it is possible to verify and validate electronic signatures associated with those documents even many years after their placement in the archive. This feature can be particularly useful in the case of documents used in court proceedings.

[2] Those two types of renewal are used in the case of an electronic signature archival form as well.

8. References

[1] Wallace C., Brandner R. and Pordesch U.: Long-term Archive Service Requirements, Internet Draft, draft-ietf-ltans-reqs-10.txt, 2006.

[2] Wong T. M., Wang C. and Wing J. M.: Verifiable Secret Redistribution for Archive Systems, Proceedings of the First International IEEE Security in Storage Workshop, 2002.

[3] Shamir A.: How to share a secret. Communication of the ACM 22, 1979, pp. 612–613.

[4] Masinter L. and Welch M.: A System for Long-Term Document Preservation, Proceedings of Archiving 2006 Conference, Ottawa, Canada, May 23, 2006 , Volume 3, pp. 61-68.

[5] CWA 15579 E-invoices and digital signatures, July 2006.

[6] ETSI TS 101 903 XML Advanced Electronic Signatures (XAdES), v1.3.2, March 2006.

[7] ETSI TS 101 733 Electronic Signatures and Infrastructures (ESI); CMS Advanced Electronic Signatures (CAdES), v1.7.3, January 2007.

[8] Feldman P.: A practical scheme for non-interactive verifiable secret sharing, in Proc. of 28th IEEE Ann. Symp. on Foundations of Computer Science, IEEE, 1987, pp. 427–437.

[9] Desmedt Y. and Jajodia S.: Redistributing secret shares to new access structures and its applica-tions, Technical Report ISSE TR-97-01, George Mason University, Fairfax, VA, 1997.

[10] Schoenmakers B.: A Simple Publicly Verifiable Secret Sharing Scheme and its Application to Electronic Voting, in Advances in Cryptology - CRYPTO'99, Vol. 1666 of Lecture Notes in Computer Science, Springer-Verlag, 1999, pp. 148-164.

[11] Tang C., Pei D., Liu Z. and He Y.: Non-Interactive and Information-Theoretic Secure Publicly Verifiable Secret Sharing, Cryptology ePrint Archive, Report 2004/201.

[12] Pedersen T.P.: Non-interactive and information-theoretic secure verifiable secret sharing, Advances in Cryptology-CRYPTO'91, 1992, pp.129-140.

[13] Stadler M.: Publicly Veriable Secret Sharing, Advances in Cryptology-EUROCRYPT'96, 1996, pp.190-199.

[14] Adams C. and Lloyd S.: Understanding PKI (second edition), Addison-Wesley, 2003.

[15] Haber S. and Stornetta W. S.: How to time-stamp a digital document, Journal of Cryptology, vol.3, no.2, 1991, pp. 99-111.

[16] Ensuring Record Integrity with AbsoluteProofSM, Technical Whitepaper, Surety, LLC, 2003.

[17] RFC 4998: Evidence Record Syntax (ERS), T. Gondrom, R. Brandner, U. Pordesch, August 2007.

[18] RFC 6283: Extensible Markup Language Evidence Record Syntax (XMLERS), A. Jerman Blazic, S. Saljic, T. Gondrom, July 2011.

otrzymano / received: 27.04.2011
przyjęto do druku / accepted: 06.06.2011 *artykuł recenzowany*

INFORMACJE