An Infrastructure for Long-Term Archiving of Authenticated and Sensitive Electronic Documents

Thiago Acórdi Ramos, Nelson da Silva, Lau Cheuk Lung, Jonathan Gehard Kohler, and Ricardo Felipe Custódio

Computer Security Laboratory
Computer Science Graduate Program
Federal University of Santa Catarina
P.O. Box 476, 88040-900 – Florianópolis, SC, Brazil
{thg,nelson,lau.lung,jonathan,custodio}@inf.ufsc.br
http://www.labsec.ufsc.br

Abstract. Secure archiving of authenticated and sensitive documents is becoming a necessity due to the dematerialization of paper based documents and processes. Huhnlein et al. combined the Open Archival Information System (OAIS) Reference Model, Evidence Record Syntax (ERS) with Secret Sharing and proposed the Comprehensive Reference Architecture for Trustworthy Long-Term Archiving of Sensitive Data. However, their proposal requires the secret reconstruction and redistribution whenever there are changes in the structure of the servers. In addition, there are some unhandled problems (e.g. compromise of the servers) and open issues (e.g. specification of a protocol) in their proposal. In this article we propose the use of a modified version of Gupta's and Gopinath's protocol G_{its}^2 Verifiable Secret Redistribution (VSR), among other mechanisms, in order to improve the reference architecture, making it suitable for long-term archiving.

Keywords: long-term archiving; secrecy; confidentiality; authenticity; secret sharing.

1 Introduction

The long-term storage of paper based documents requires constant maintenance of the components involved: substratum (paper) and ink. Things are not different when it comes to electronic documents. It is necessary for the document to be constantly preserved, considering its digital nature and its unique characteristics.

With the advancement of Internet, mobility and the equipments we are increasingly more connected, performing tasks from anywhere in the digital environment. Public-key Infrastructure (PKI), digital certificates and signatures contributed to the dematerialization of processes, which were previously only possible in paper, in the electronic environment.

Thus, a treatment similar to the paper based documents one is necessary for electronic documents, considering its digital nature and preservation challenges.

However, there is still no appropriate solution to the long-term digital preservation problem, due to the several challenges that must be faced. The letter from United Nations Education, Scientific and Cultural Organization (UNESCO)[1] reinforces the importance of preserving the digital heritage and the migration to this environment, motivating us to work in this research field, aiming the trustworthiness of electronic documents.

The work of Huhnlein et al. [17] propose a comprehensive reference architecture for trustworthy long-term archiving of sensitive data employing the OAIS Reference Model, Evidence Record Syntax (ERS) and Secret Sharing, being an important work due to its attempt of dealing with authenticity and confidentiality of documents, based on this technology, in the same infrastructure. We found some problems and gaps in their proposal and, in this sense, we propose improvements to this reference architecture which we present in this article, incorporating a modified version of Gupta's and Gopinath's [12] protocol and other mechanisms.

We assume there is a private point-to-point connection between clients and servers, and among pairs of servers, a reliable broadcast channel which connects all the servers, and that connections between clients and servers have an unconditional secrecy because traditional methods (e.g. security socket layer) can be eavesdropped and the information will be revealed when the algorithms employed are broken. We require that servers can reliably erase the shares from a previous period in order to prevent an adversary to obtain some information about the secret.

The rest of this work is organized as follows: in section 2, we list the related works. In the next section 3, we provide an overview of the comprehensive reference architecture. In Section 3.1, we address the problems and gaps found in this architecture. An improved reference architecture is proposed in section 4. In section 5, we discuss the management and maintenance of the infrastructure, emphasizing archived objects properties handling, and briefly evaluate our proposal. Finally, we bring the concluding remarks and point some future work in section 6.

2 Related Works

In general, when it comes to the aspects of long-term archiving, there are several publications exploring and discussing strategies for obsolescence and preservation of the basic properties of storage (e.g. conservation of bits), especially projects from archives and libraries [2,4]. Moreover, there is the request for comments (RFC) 4810 [29] which lists a set of long-term archive service requirements, bringing rationales and considerations that must be weighted in such systems. Finally, the Open Archival Information System (OAIS) Reference Model, an International Organization for Standardization (ISO) standard (ISO 14721:2003 [18]),

[1] `http://portal.unesco.org/ci/en/files/13367/10700115911Charter_en.pdf/Charter_en.pdf`

makes recommendations on what an archive should provide for the long-term preservation of digital information, establishing a common framework of terms and concepts that comprise an OAIS. Some archiving system's projects already implemented are based on the OAIS[2] model.

It is know that digital signatures and certificates have a limited lifetime and need constant maintenance, which can be accomplished by time-stamps [14]. RFC 3126 [26] established the electronic signature formats for long-term electronic signatures which, aftermost, were later extended by technical specifications CMS Advanced Electronic Signatures (CAdES)[9] – also published by the Internet Engineering Task Force (IETF) in RFC 5126 [25] and XML Advanced Electronic Signatures (XAdES)[10] from the European Telecommunications Standards Institute (ETSI). Though, these formats are not effective for large volumes of data since the maintenance of these objects is individual – an archive time-stamp for each signature. In order to cover this gap there was created the Evidence Record Syntax (ERS) [6], which is based on Merkle Trees [21], and consists of a tree of hashes of the objects to be preserved. Thus, preserving the integrity and authenticity provides greater scalability and facilitates the operation by the archives.

Initial works on confidentiality were using encryption to provide secrecy for archived objects [20,8,1,15,19]. However, encryption algorithms are conditionally secure and become weaker in the course of time (increase of computing power and advances in cryptanalysis techniques) and, hence, they are not suitable for long-term. An alternative technique that provides unconditional security is the proposal of Shamir [27]. A hybrid approach was proposed by Wang et al. [30], using secret sharing for the recovery of symmetric keys, but the same obsolescence problem can be found. Some proposed works [22,17,28], for example, use the secret sharing technique, which provides some degree of availability – depending on the threshold adopted. Though, they adopted share renewal mechanisms, except for POTSHARDS[28], in which the secrets are reconstructed for a later redistribution, which leaves them exposed in the reconstruction site (e.g. a central recovery server). POTSHARDS [28] does not have a share renewal mechanism, which is a future work of their proposal.

Herzberg et al. proposed a scheme called proactive secret share [16] in which the lifetime of the secret is divided in shorter periods to avoid mobile adversaries. Their proposal basically employs a sum of polynomial, in one of them the free term is equal to zero, which avoids the reconstruction of the secret. Their proposal has the limitation that the access structure must be maintained, i.e., it needs the same number of available servers before and after a renewal. Another drawback is that Nikov and Nikova [23] investigated the security of proactive secret sharing schemes and found specific weaknesses when a mobile adversary is considered. These weaknesses, which can compromise the secret, can be applied to Herzberg et al. model.

Another approach to the renewal problem is the Verifiable Secret Redistribution (VSR)[31] protocol which combines the secret renovation protocol

[2] `http://www.oclc.org/research/activities/past/rlg/oaisactivities.htm`

without reconstruction from Desmedt and Jajodia [7] with Verifiable Secret Sharing (VSS). This protocol prevents the share renewal to execute a secret sharing technique for each shares held by the servers and allows a different access structure after redistribution. An extension to the VSR (xVSR) was proposed by Gupta and Gopinath [13], because all recipients (in this case, the servers) must be honest, relaxing this requirement only to the needs of the majority. In these protocols the approach of Feldman [11] was used as the VSS technique. Another proposal (G^2_{its} VSR) from Gupta and Gopinath [12] has revised their protocol and has incorporated Pedersen's VSS approach [24], thus providing information theoretic secrecy.

3 Overview of Comprehensive Reference Architecture

In this section we reproduce an overview of Huhnlein et al.'s [17] work, in which we found problems and gaps that are discussed in section 3.1, in order to solve them and incorporate them into our proposal.

Based on the results of projects ArchiSig [5] and ArchiSafe [32], in joint with the Evidence Record Syntax (ERS) [6], The German Federal Office for Information Security (BSI) has developed a technical directive that regulates the trustworthy long-term archiving for federal agencies in Germany, providing integrity and authenticity of archived data. Huhnlein et al. [17] extended this architecture, combining the different requirements involved, to provide confidentiality and availability as well. Their architecture supports operations for submission, retrieval, evidence request, complementary data request (such as metadata) and deletion of archived objects. The identifier of the archived objects is called Archive Token. Figure 1 illustrates the modules and operations of the proposed infrastructure.

The Archive Gateway Module receives requests from applications and encapsulates the objects to be archived in XML Archival Information Package (XAIP) – if the object is not in that package, it controls the processes and formats using standardized XML Schemas and performs access control. in the case the archival package is already signed in the application layer, the Crypto Module is invoked in order to verify the signatures found. Finally, this module requests evidence services from Evidence Module, and stores the archival package in Storage Module.

The Evidence Module generates hashes of all documents and the resulted hashes are concatenated into hash trees [21]. A time-stamp is generated for the hash resulted from the root of the tree, consolidating the validity of all documents involved. Before algorithms and cryptographic parameters become weak or be compromised, the time-stamp is renewed. When required, this module uses these trees to generate Evidence Records (ER) in concordance with the ERS [6] standard.

The Crypto Module supports cryptographic operations required by the infrastructure, such as hash functions, time-stamping and, optionally, signature generation and verification. The Crypto Interface must be compatible with standard interfaces, ensuring interoperability among others Crypto Modules.

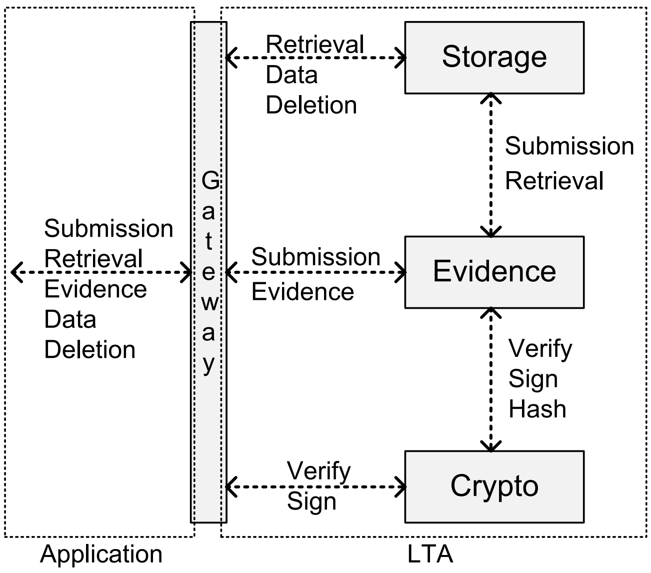

Fig. 1. Overview of modules and operations of the Comprehensive Reference Architecture

The Storage Module (eSafe), in addition to search, deletion and bit-exact reproduction services, should provide a distributed architecture that ensures a (k, n) threshold confidentiality. The Shamir's [27] secret sharing scheme was adopted.

Following the ERS [6] standard, the infrastructure continuously executes the process of Archive Time-Stamp Management, divided into Time-Stamp Renewal and Hash Tree Renewal. The former consists in renewing the time-stamp of the root's tree before events that would invalidate this time-stamp happen – such as the expiration and revocation of certificates and obsolescence of cryptographic algorithms or parameters, happen. The latter occurs when the hash algorithm is considered insecure, requiring the replacement of this algorithm for a secure one.

The Share Renewal mechanism which is described consists in a renewal of the shares whenever a server is added or removed from the system. When a server is added, there is no need to modify the existing shares. However, when a server is removed, the shares must be recalculated to invalidate the share to the removed server. This renovation has impact in Archive Time-stamp Management, depending on the configuration of shared mode that was adopted. There are two modes: inside and outside shared mode.

In Inside Shared Mode the hash is calculated on the data archive and only in eSafe Module the secret share is applied. The size of the tree depends on the number of stored objects. Considering a (k, n) threshold, where n is the total of shares and k is the minimum needed for the reconstruction of the secret, k shares are accessed to a Hash Tree Renewal.

In Outside Shared Mode, the mechanism of secret share is applied at the application layer and the system reads the shares as several documents being the hash tree built from the hash value of each share. The size of the tree depends on the number of shares. For the Hash Tree Renewal it is necessary to access all the n shares.

3.1 Discussion about Reference Architecture

In our analysis of the comprehensive reference architecture from Huhnlein et al. [17] there were detected some problems and gaps that must be solved and filled so that the architecture meets its goals.

Our first concern is related to the suitability of encryption algorithms in the course of time. Although it is true that encryption techniques are designed to be secure for a long time, its suitability degrade over time. Long-term means more comprehensive periods of time than the lifetime designed for encryption techniques, such as indefinitely. As an example of this king of documents, there are the military and government secret files, which needs secrecy against nation-state enemies. Even historical documents leakage can be harmful because the adversaries can learn secrets about the nation. Other examples could be medical records, lawsuits, industrial secrets and so on.

The secret sharing mechanisms are unconditionally secure [27] compared to the security of encryption techniques that provide only conditional security. However, the Share Renewal approach requires that archived object are reconstructed when there is a removal of the servers – to invalidate its shares and for the Hash Tree Renewal – whether the Inside Shared Mode was used, leading to the same problem of the encryption mechanisms. There is a central server for recovery that can be compromised by an adversary who, thus, obtains confidential information [31].

Huhnlein et al. mentioned a variant of the sharing mode, generating shares in the application layer and using the object hash to build the tree. Nevertheless, this approach only postpones the problem, because at some moment it will be necessary to obtain the hash of the objects for the hash tree renewal.

In the proposal, some mechanisms of monitoring and maintenance of the integrity of the shares in the eSafe Module were not mentioned. An attacker that maliciously alters or deletes $(n - k + 1)$ shares will cause the loss of this information.

Also in relation to shares, there are no mechanisms of defense against active and mobile adversaries. The former may compromise the shares before they reach the servers, during a (re)distribution, and this modification will not be detected. The latter can continuously obtain shares in the archive lifetime until she obtains k shares and reconstruct the archived object, since she make it before any changes in the configuration of servers happen.

The hash tree receives a time-stamp at its root that guarantees the long-term maintenance of the objects. However, this tree is not authenticated and it is possible for an attacker to replace it or tamper it, since she has access to the Time-Stamp Authority (TSA) used by the system, e.g. one of a third party.

Malicious Long-Term Archives (LTAs) are not considered in the Reference Architecture proposed. In real systems there is a valid assumption that an intruder can compromise and maliciously control the behavior of the components, especially in long-term systems in which the attacker has an undetermined period to operate.

Another obstacle for long-term archiving is the management of identifications (IDs) of archived objects by the clients, using the Archive Token. This approach is possible if there is a way to recover the IDs, since the clients do not have infrastructure to store information for a long time, even if it were stored on cryptographic tokens.

Finally, a protocol for distribution and renewal of shares was not chosen in the proposed architecture, leaving a significant gap in the system's definition. The interaction between the components of the system, the clients and the servers must be set and secure in order to fulfill the requirements of archival systems.

4 A New Approach of Reference Architecture

The main problem of Comprehensive Reference Architecture [17] is the Share Renewal mechanism, which exposes the secret in a risky way on a recovery central server, being attractive to attackers. Thus, the advantage of using threshold's cryptography is reduced to the security against intrusions on the central server.

Our proposal is the incorporation of a modified version of the protocol (G^2_{its} VSR) [12] in the Comprehensive Reference Architecture (CRA) – an essential mechanism which was not considered on CRA, thus, solving the Share Renewal problems and mobile adversaries in addition to tolerate faulty LTAs. Afterwards, we will discuss in details each of these problems along with our proposed solutions.

The secret sharing applied is the proposal of Shamir [27]. To share a secret S between n recipients, being k of them required for reconstruct the secret and no information about the secret can be obtained from $k-1$ shares, choose a polynomial f of order $k-1$, let the independent term be $f(0) = a_0 = S$ and choose random values to the other coefficients. The shares will be the values obtained from the evaluation of f into i points, for $i = 1...n$, such that $y_i = f(x_i)$. In order to reconstruct the secret it is only necessary to apply an interpolation formula with at least k shares and its identifying indices (i), and, then, evaluate $f(0) = S$.

The secrets being shared are electronic documents, which are split into blocks and have the secret sharing mechanism applied into all of them as follows: divide the file into blocks of the same size, fix values of x_i while varies the polynomial for each block. Subsume in a package called Share all the values of $y_i = f(x_i)$ for each x_i. At the end, distribute these n shares among the n recipients [17].

In order to avoid the Share Renewal a secret redistribution is applied, without having to reconstruct it. A client C has a secret S and divides it into n pieces, storing them on n servers, so that with k or more pieces it is possible to reconstruct S, with $k \leq n$, building an (k, n) access structure. In the course of time, the n secrets are redistributed in n' pieces being that at least k' parts are needed to reconstruct the same secret S, with $k' \leq n'$, forming an (k', n') access structure. These access structures can be of different sizes, depending on the number of servers available when the redistribution process is performed. In some moment the client can reconstruct the secret [31]. Figure 2 illustrates this protocol.

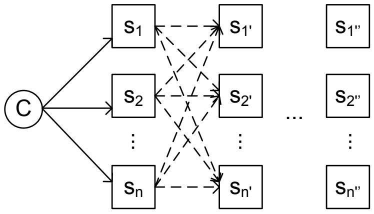

Fig. 2. Overview of VSR protocol

The modification on the protocol G^2_{its} VSR [12] relates to the use of secret sharing. They calculate the sub-shares $\hat{s}_{ij}$ for then obtain a new share s'_j using Lagrange interpolation. In our approach we simply keep the sub-shares $(\hat{s}_{ij})$ without the computation of a new share (s'_j). When all the servers receive their new shares and all the information is verified, they should erase their shares of the previous period, like PSS scheme [16]. The security against mobile adversaries relies on this erasure and if this procedure is not executed, an adversary can obtain some information or even compromise the secret.

The redistribution process uses the VSS schemes so that members can verify the integrity and validity of the shares without knowing the secret. In the G^2_{its} VSR protocol, as stated earlier, the approach of Pedersen [24] is applied, thus obtaining information theoretic secrecy. The execution of the chosen protocol is almost identical to xVSR [13], except that twice the information is sent by VSS protocol.

From Wong et al. [31], k non-faulty servers are required, and $k-1$ faulty servers can be tolerated, thus, the constraint is that $k + (k-1) \leq n$, or $k \leq (\frac{n+1}{2})$.

The use of protocol G^2_{its} VSR [12] solves the Share Renewal problem, since the redistribution can be applied when there is a structural change, either adding or removing a server from infrastructure. Mobile adversaries can be mitigated by applying a secret redistribution before an attacker can obtain k shares and reconstruct the secret S.

Yet, in the proposal of the xVSR protocol, predecessor of G^2_{its} VSR, malicious LTAs are tolerated in order to cover more realistic situations, requiring only the majority be trustworthy, instead of all recipients (servers) [13]. With this, a protocol for the archive is defined – an existing gap in the Comprehensive Reference Architecture. The verifiable secret share [24] scheme adopted in the protocol makes the verification of shares by the servers possible and also provides information theoretic secrecy.

And how is the Merkle [21] tree, or the maintenance of integrity and authenticity, regarding shares? Suppose you share the secret S with a $(k = 3, n = 4)$ threshold, being the shares (s_1, s_2, s_3, s_4), and that a hash tree was created from the hashes of these shares (h_1, h_2, h_3, h_4), as illustrated in Fig. 3. Any combination of 3 shares or more can prove that they belong to the archived object and, consequently, the integrity and authenticity of this object are established.

Now consider that there was a redistribution of shares with a $(k = 2, n = 3)$ threshold, forming the subset $(s_{11}, s_{12}, s_{13}, s_{21}, s_{22}, s_{23}, s_{31}, s_{32}, s_{33}, s_{41}, s_{42}, s_{43})$. Any combination of 2 sub-shares of 3 shares (e.g. $s_{11}, s_{12}, s_{22}, s_{23}, s_{41}, s_{43}$) or more sub-shares can recover the primary shares (in this case, s_1, s_2, s_4), returning to the previous case. No changes in the tree must be done when there is a redistribution of shares. We reached this result with our modification on protocol G^2_{its} VSR.

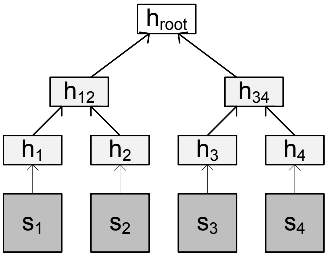

Fig. 3. Merkle tree built from the hash of shares

As previously mentioned in section 3.1, the trees generated in the comprehensive reference architecture are not signed by de system. In our proposal, we added this requirement in order to enhance the security in relation to the authenticity of the trees. In order to do so, we suggested using a distributed signature algorithm, in which a private key S_k is shared between the participating servers by threshold techniques so that, even with possible outages of any server – be them malicious or not, the signature can be performed, provided that it has the necessary quorum.

Concerning the monitoring of the integrity of the shares on the servers, each one of them actively monitors the integrity of its shares through stored hashes [28], in order to reduce the likelihood of tampering in shares, be it intentional or not. In addition, each server must use techniques of mirroring, redundant and/or erasure codes, both local and distributed.

Lastly, a remaining problem of the architecture is the management of the identification of archived objects by the clients instead of by the system. This approach is interesting so that a centralized index of this objects is not required, which is a point to be attacked in the system – such as the recovery central server previously discussed. However, clients do not have infrastructure to store such information for such a long time, e.g. loss of a cryptographic token or formatting of the operational system may occur. One possible way to address this problem is by using the techniques of global name-spaces and approximate pointers of POTSHARDS [28].

5 Discussion and Evaluation of the Proposal

The infrastructure management comprehends the maintenance of the properties that a long-term archiving service must indefinitely preserve. Properties such as integrity, authenticity, confidentiality and availability, need a constant treatment since archiving systems greatest enemy is time.

By using Merkle trees [21] it is possible to manage and maintain the integrity and authenticity of several archived objects at once. However, a maintenance of the tree itself is necessary so that these properties can be maintained.

The time-stamp on the root of the trees must be actively renewed before that the cryptographic algorithms and parameters become weak or the certificate of Time-stamp Authority (TSA) or Certification Authority (CA) belonging to the certification path TSA are revoked or expired [6].

Another necessary maintenance on the trees is related to the hash algorithm adopted. Whenever an algorithm is considered weak it must be replaced by a secure one. However, in order to do so, it is necessary to have access to all the n shares of each archived object and, thus, it becomes possible to reconstruct the trees. These maintenance processes are described in ERS and are adopted by several papers [5,32,17,3].

The integrity and authenticity of the shares are grounded on Merkle trees [21] and ERS [6] which make them attractive to attackers. The replication of these trees must be kept to avoid a single point of failure. Also, in this sense, distinct algorithms must be applied, for example, identical trees applying distinct and secure algorithms, thus avoiding an abrupt crack that compromises the system. The algorithms applied in the infrastructure should be monitored regarding their security and, as soon as they become insecure, they should be replaced by others which are considered secure.

Merkle trees guarantees the authenticity, and partly, the integrity of the shares that compose an archived object. This happens partly because the integrity of the shares requires monitoring and maintenance of the Storage Module, as previously described in section 3.1. Using an active monitoring, redundancy into servers and among them, it is possible to manage the integrity of archived objects.

The shares shall be distributed by distinct LTAs in different geographic locations in order to balance and distribute the load by the infrastructure without a LTA, otherwise a minority may reconstruct the secret, thus achieving the desired confidentiality and availability. Thus, an attacker needs that at least k LTAs maliciously cooperate to reconstruct the secret – an unlikely situation.

The use of threshold cryptography and techniques for redundancy provides the desired confidentiality and a certain degree of availability to the infrastructure. Approaches such as the encryption or the share renewal have points of exposure of confidential information, being a target to attacks. The adopted protocol [12] solves this issue and also provides verification of the integrity of shares by the servers.

Our proposal is just on earlier stages and, thus, more informations and details are not available at this moment. We describe succinctly the amount of disk space required (storage blow-up) and the number of secret sharing operations that are executed at each period, excluding the amount from VSS.

When a client sends a document to the servers this is split into blocks and a (k, n) secret sharing operation for each block of the file is applied. Because secret sharing requires that the shares are at least the same size as the information to reach unconditional security, one secret sharing operation will be executed for each of i blocks (i operations) and the storage blow-up (b) will be $b = i.n.s$, being s the share's size of a block.

At a redistribution, each share is treated as a file and will be split into i' blocks and a (k', n') secret sharing operation will be applied in each of these blocks. Thus, one secret sharing operation will be executed for each i' blocks of n shares ($n.i'$ operations) and the new storage blow-up (b') will be $b' = (n.i'.n'.s')$, being s' the share's size of a block. However, the erasure of the previous shares after the correct execution of the redistribution protocol will free b bytes of disk space. We applied different variables in formulas because the parameters can be changed as wished.

As an example, suppose we have a file with the size of 9.888 bytes, a prime modulus of $s = 48$ bytes and $n = 3$ servers. So, we have $i = 206$ blocks and the resulting blow-up will be $b = 29.664$ bytes after the distribution. Table 1 and the Fig. 4 shows the storage blow-up after redistributions (b'), considering that we keep the same configuration.

Table 1. Blow-up (b') after six rounds of redistributions

rounds	1	2	3	4	5	6
b' (in KB)	57.94	202.78	579.38	1767.09	5272.32	15845.9

Another evaluation is that the size of modulus have a significant impact on the storage blow-up. Table 2 shows the blow-up (b') when we choose different modulus, considering that we keep the same configuration.

Table 2. Blow-up after six rounds of redistribution with different modulus

modsize (bytes)	b'_1	b'_2	b'_3	b'_4	b'_5	b'_6
6	57.9	202.66	579.02	1766.02	5269.11	15836.29
12	57.94	202.78	579.38	1767.09	5272.32	15845.9
24	57.94	202.78	579.38	1767.09	5272.32	15845.9
48	57.94	202.78	579.38	1767.09	5272.32	15845.9
50	58.01	203.03	580.08	1769.24	5278.71	15865.14

We can infer from these informations that the storage blow-up increases quickly, as expected, and the size of modulus has not a great impact on that.

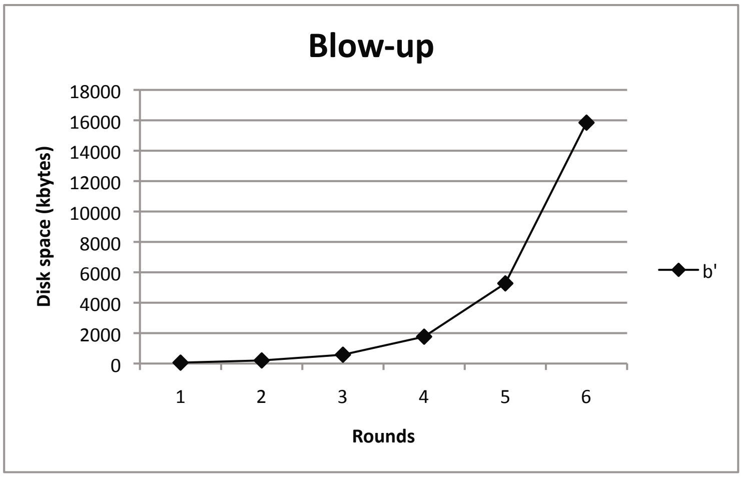

Fig. 4. Blow-up (b') after six rounds of redistributions

Table 3. Secret sharing operations per round (b'_i) with different sizes of modulus (in bytes)

modsize (bytes)	b'_1	b'_2	b'_3	b'_4	b'_5	b'_6
6	4944	14832	44496	133488	400464	1201392
12	2472	7416	22248	66744	200232	600696
24	1236	3708	11124	33372	100116	300348
48	618	1854	5562	16686	50058	150174
50	594	1782	5346	16038	48114	144342

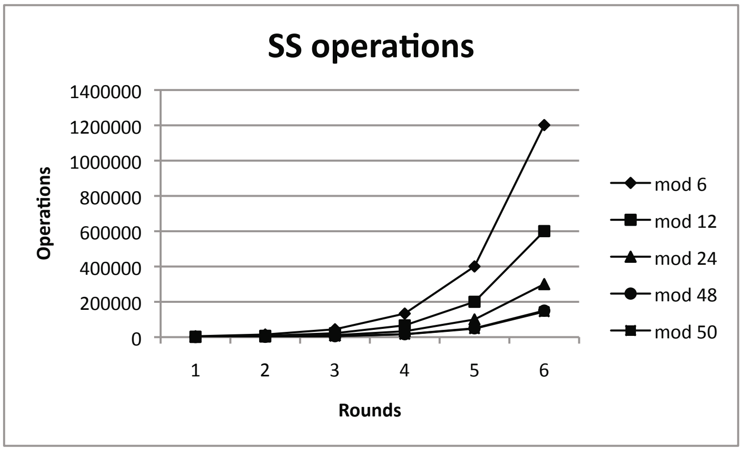

Fig. 5. Secret sharing operations per round with different modulus

However, reducing the modulus' size increases the number os blocks and, consequently, the amount of secret sharing (SS) operations that need to be executed ($n.i'$ operations). Table 3 and Fig. 5 shows this information.

The blow-up shown is considered for the overall system (we can divide it by n to get the blow-up per server) and the costs of storage are decreasing over time, thus, the storage blow-up itself is amortized.

We aim to evolve our proposal and more precisely evaluate it, which will be a subject for future works.

6 Concluding Remarks and Future Works

We presented a new proposal for a reference architecture, based on the Comprehensive Reference Architecture from Huhnlein et al. [17], with the incorporation of a modified version of the verifiable secret redistribution protocol from Gupta and Gopinath [12]. With the adoption of other techniques and mechanisms it is possible to provide a satisfactory approach to the maintaining problem of long-term archiving, especially the integrity, authenticity and confidentially of archived objects.

The appliance of threshold techniques, such as in the approach of Shamir [27], together with the secret redistribution protocol without the need of reconstruction from Desmedt and Jajodia [7], included in protocol G^2_{its} VSR [12], provides information theoretic secrecy to the architecture for long-term archived objects, without the need of recalculation of the threshold. Thus, a certain degree of availability is obtained, depending on the values of k and n adopted, since the objects are divided into n pieces but can be reconstructed with at least k pieces, in a (k, n) threshold.

Other techniques of redundancy are necessary, such as Internet connections, routers replication, load balancing, among others, which are beyond the scope of this work but must be also equally deployed for the provision of the availability in a comprehensive and appropriate way to a system of this size and importance. Defense mechanisms against denial of service (DoS) and intrusion attacks are also essential to the infrastructure and must be incorporated into the solution.

Problems related to general aspects of long-term archiving were also not addressed because the subject is extensive, but must be observed and aggregated to the solution, such as different media technology (e.g. hard drive, solid state drive, optical media, magnetic tape and microfilm), distinct operating systems and software, emulation, notarization and migration strategies, metadata (which are required for most archiving systems, including the OAIS reference model), among others subjects.

As future works we suggest the implementation and evaluation in relation to different subjects of the proposal, tests and refinements regarding the appropriated period for share redistribution, experiments with different techniques and values of k and n at threshold cryptography, incorporating mechanisms against DoS and intrusion, a formalization of the system and its protocols, the extension of the proposal detailing each component and a survey and selection of the most appropriate strategies in issues of general aspects of archiving.

Acknowledgements

We would like to thank the Brazilian Chamber of Electronic Commerce[3] for supporting our research. This work will make part of a master thesis at Computer Science Graduate Program (PPGCC) of the Federal University of Santa Catarina (UFSC).

References

1. Adya, A., Bolosky, W.J., Castro, M., Cermak, G., Chaiken, R., Douceur, J.R., Howell, J., Lorch, J.R., Theimer, M., Wattenhofer, R.P.: Farsite: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment. Operating Systems Design and Implementation (2002)
2. Beagrie, N., Jones, M.: Preservation Management of Digital Materials: The Handbook. Digital Preservation Coalition (2002)
3. Blazic, A.: Long Term Trusted Archive Services. In: First International Conference on the Digital Society (ICDS 2007), pp. 29–29 (2007)
4. Borghoff, U., Rödig, P., Scheffczyk, J., Schmitz, L.: Long-term Preservation of Digital Documents: Principles and Practices (2006)
5. Brandner, R., Pordesch, U.: Long-term conservation of provability of electronically signed documents. Beitrag zu ISSE, pp. 2–5 (2002)
6. Brandner, R., Pordesch, U., Gondrom, T.: Evidence Record Syntax (ERS). Internet Engineering Task Force (IETF) Networking Group, Request for Comments 4998 (2007)
7. Desmedt, Y., Jajodia, S.: Redistributing Secret Shares to New Access Structures and its Applications (1997)
8. Druschel, P., Rowstron, A.: PAST: a Large-scale, Persistent Peer-to-Peer Storage Utility. In: Proceedings of the Eighth Workshop on Hot Topics in Operating Systems, 2001, pp. 75–80 (2001)
9. European Telecommunications Standards Institute: Electronic Signatures and Infrastructures (ESI); CMS Advanced Electronic Signatures (CAdES) (November 2009)
10. European Telecommunications Standards Institute: Electronic Signatures and Infrastructures (ESI); XML Advanced Electronic Signatures (XAdES) (June 2009)
11. Feldman, P.: A Practical Scheme for Non-Interactive Verifiable Secret Sharing. In: 28th Annual Symposium on Foundations of Computer Science (sfcs 1987), pp. 427–438 (October 1987)
12. Gupta, V.H., Gopinath, K.: G_{its}^2 VSR: An Information Theoretical Secure Verifiable Secret Redistribution Protocol for Long-Term Archival Storage. In: Fourth International IEEE Security in Storage Workshop, pp. 22–33 (2007)
13. Gupta, V., Gopinath, K.: An Extended Verifiable Secret Redistribution Protocol for Archival Systems. IEEE, Los Alamitos (2006)
14. Haber, S., Stornetta, W.: How to Time-Stamp a Digital Document. Journal of Cryptology 3(2), 99–111 (1991)
15. Haeberlen, A., Mislove, A., Druschel, P.: Glacier: Highly Durable, Decentralized Storage Despite Massive Correlated Failures. In: Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation, vol. 2 (2005)

[3] `http://www.camara-e.net`

16. Herzberg, A., Krawczyk, H., Yung, M.: Proactive Secret Sharing Or: How to Cope With Perpetual Leakage. IBM TJ. Watson Research Center, 1–22 (1995)
17. Huhnlein, D., Korte, U., Langer, L., Wiesmaier, A.: A Comprehensive Reference Architecture for Trustworthy Long-Term Archiving of Sensitive Data. In: 2009 3rd International Conference on New Technologies, Mobility and Security, pp. 1–5 (December 2009)
18. ISO: ISO/IEC 14721:2003: Space Data and Information Transfer Systems — Open Archival Information System — Reference Model. International Standardization Organization, Geneva, Switzerland (2003)
19. Kotla, R., Alvisi, L., Dahlin, M.: SafeStore: a Durable and Practical Storage System. In: 2007 USENIX Annual Technical Conference on Proceedings of the USENIX Annual Technical Conference (2007)
20. Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C., Zhao, B.: OceanStore: an Architecture for Global-Scale Persistent Storage. ACM SIGPLAN Notices 35(11) (2000)
21. Merkle, R.C.: Protocols for public key cryptosystems. In: IEEE Symposium on Security and Privacy, vol. 0, p. 122 (1980)
22. Miyamoto, T., Doi, S., Nogawa, H., Kumagai, S.: Autonomous Distributed Secret Sharing Storage System. Systems and Computers in Japan 37(6), 55–63 (2006)
23. Nikov, V., Nikova, S.: On proactive secret sharing schemes. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 308–325. Springer, Heidelberg (2004)
24. Pedersen, T.P.: Non-interactive and Information-Theoretic Secure Verifiable Secret Sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
25. Pinkas, D., Pope, N., Ross, J.: CMS Advanced Electronic Signatures (CAdES). Internet Engineering Task Force (IETF) Networking Group, Request for Comments 5126 (2008)
26. Pinkas, D., Ross, J., Pope, N.: Electronic Signature Formats for Long Term Electronic Signatures. Internet Engineering Task Force (IETF) Networking Group, Request for Comments 3126 (2001)
27. Shamir, A.: How to Share a Secret. Communications of the ACM 22(11), 612–613 (1979)
28. Storer, M.W., Greenan, K.M., Miller, E.L., Voruganti, K.: POTSHARDS—a secure, recoverable, long-term archival storage system. ACM Transactions on Storage 5(2), 1–35 (2009)
29. Wallace, C., Pordesch, U., Brandner, R.: Long-term Archive Service Requirements. Internet Engineering Task Force (IETF) Networking Group, Request for Comments 4810 (2007)
30. Wang, E., Yau, J., Hui, L., Jiang, Z., Yiu, S.: A Key-Recovery System for Long-term Encrypted Documents. IEEE, Los Alamitos (2006)
31. Wong, T., Wing, J.: Verifiable Secret Redistribution for Archive Systems. In: Proceedings of First International IEEE Security in Storage Workshop, 2002, pp. 94–105 (December 2002)
32. Zimmer, W., Langkabel, T., Hentrich, C.: ArchiSafe: Legally Compliant Electronic Storage. IT Professional 10(4), 2633 (2008)