June, 2015

Efficient Robust and Secure E-DRM with Encrypted Content Search

Maged Ibrahim, *Helwan University*

Efficient Robust and Secure E-DRM with Encrypted Content Search

ARTICLE *in* INTERNATIONAL JOURNAL ON INFORMATION · JUNE 2015

Impact Factor: 0.36

1 AUTHOR:

Maged Hamada Ibrahim
Helwan University
58 PUBLICATIONS **157** CITATIONS

SEE PROFILE

Efficient Robust and Secure E-DRM with Encrypted Content Search

Maged Hamada Ibrahim
Departement of Electronics, Communications and Computers Engineering,
Faculty of Engineering, Helwan University,
1, Sherif st., Helwan, Cairo, P.O. Box 11792, Egypt
mhii72@gmail.com

Abstract

Enterprise digital rights management (E-DRM) systems protect business digital applications by allowing an author in an organization to securely upload his confidential package/file and store the contents in a private way on secure servers. This is done in a way that – later – allows an authorized user who is able to prove his authorization for the package to an authorization authority to download and use these contents in a private way. However, the user maybe authorized for many archived packages and files of very large sizes and at the same time, he is targeting only a file that contains certain private keywords. It is inadequate that the user downloads and decrypts all the materials he is authorized for in order to pick one. In this paper we propose an efficient (E-DRM) protocol for robust and secure storage and retrieval of digital contents with the property that allows the user in a non-interactive way to search the files in their encrypted form for matching private keywords and then to download these particular files.
Keywords: Enterprise security, digital rights management, secure storage, information dispersal, encrypted file search, private keyword search.

1 Introduction

The copyright infringe, such as free distribution, unauthorized usage, illegal sharing of copyrighted digital contents is becoming an undesired phenomenon for the authors or the owners of these electronic materials since, such materials (e.g. e-books, images, music, movies, games, application software, etc.) are very easily duplicated without the reduction in quality. Thus, the digital contents industry will be heavily affected and deteriorated. Digital Rights Management (DRM) in general deals with the main issues: The ability to deliver digital contents to authorized users in an efficient, robust and reliable way, the ability to prevent access from unauthorized users and the ability to prevent theft and illegal redistribution which may occur from authorized access users.

Efforts have focused on rights-centric security policies and enhanced mechanisms. Consumer-centered security considerations is receiving more attention. In a secure DRM system, digital contents should be encrypted based on the cryptographic technology, and then consumers acquire encrypted contents by

means of the Pull or Push mode. Much more DRM applications are requiring a fine grained contents usage control and the rights definition, expression as well as interpretation. As a result , an inter-operable, well-defined rights expression language is indispensable. Transmission of the digital contents and of their licenses are both protected by means of cryptographic mechanisms such as encryption and digital signature (e.g. [1, 2, 19]). Besides, the execution of the license needs a close or trust environment, which includes trusted DRM agent, trusted key storage, trusted I/O, and so forth. For copyrights protection and pirates tracing, it is needed to embed a section of imperceptible data into contents by using the watermarking and fingerprints technology [17, 18], whereas the embed data is authenticated only by special equipments or approaches.

Beside cryptographic mechanisms for robustness and authenticity and beside watermarking and digital fingerprints, Display-Only File-Servers (DOFS), can transparently and effectively stop information theft by insiders in most cases, even if the insiders have proper authorities to read/write the protected information. The DOFS architecture ensures that bits of a sensitive file never leave a protected server after the file is checked in and users can still interact with the protected file in the same way as if it is stored locally. Essentially, DOFS decouples "display access" from other types of access to a protected file, and provides users only the "display image" rather than bits of the file. Therefore, DOFS can have less dependency on the trusted client software against information theft by insiders.

The current DRM techniques mainly have two types of applications: (1) Systems for distributing content to consumers in a controlled way against piracy; (2) Systems for managing access to sensitive document content within an enterprise. The first application is called "Pirates Tracing" while the second application is often called " Enterprise Digital Rights Management (E-DRM)". E-DRM plays an extremely important role in fighting against information theft, especially the theft due to insider threats. A good survey on the security requirements and technical issues of digital rights management is found in [4]. The work in this paper focuses on constructing a cryptographic protocol for E-DRM that ensures robustness of storage and retrieval of large media packages against malicious adversaries that are willing to corrupt file-servers to destroy the digital media or to capture cryptographic encryption. The protocol protects the privacy of the digital media against unauthorized users and provides the authorized users the ability to decrypt the contents in a fully private and secure way. The protocol allows the storage of large media packages in an efficient way on multiple servers to ensure integrity and robustness. Sometimes, the user is authorized for several number of media files from which he is only willing to pick one. It is impractical that the user downloads and decrypts all files to search for the one he needs. For this purpose, our protocol allows the user to search keyword(s) within the encrypted files that he is authorized for on the server and then decrypt and use.

2 Related Work

The requirements of a well-designed DRM system have been studied in several subsequent publications, Arnab et al. [4, 5, 6], Bartolini et al. [7], Mulligan et al. [8], and Park et al. [9]. The recent contribution of [1] is to device a stronger digital rights management protocol for enterprise applications (SDRMP) that

overcomes the security problems and efficiency drawbacks in previous protocols [2, 3, 10, 19, 20]. A different network topology and communication model were considered to reduce the computation burden on the author, the authority, as well as the users and provide efficient and robust storage of large files on the servers. The protocol is secure against malicious behavior of a minority of the servers.

3 Motivations and Contribution

Motivations. In the E-DRM protocol of [1], since it allows storage of large packages, the user maybe authorized for many stored packages and files of very large sizes and at the same time, he is targeting only a file that contains a certain keyword(s). For example, a user that is authorized for a package of online movies maybe interested in watching/downloading movies with certain keywords (e.g. action, horror, romantic, etc.) or keywords that specify a certain celebrity. It is inadequate that the user downloads and decrypts all the materials he is authorized for, in order to pick one.

Our contribution. We construct an (E-DRM) protocol for robust and secure storage and retrieval of digital contents with the property that allows the user in a non-interactive way to search the files in its encrypted form for matching private keyword(s) and then to download this particular file(s). The keywords must be kept confidential due to several reasons, some of which are: The keywords give the adversary a strong estimate about the contents of the package and hence violate its privacy. Also, when a user reveals the keywords he is searching for, they reveal to the adversary information about his activities, interests or progress (e.g. in a certain project) which is undesired specially in a competitive environment. Hence, the keywords uploaded by the author of the package as well as the keywords that the user is looking for must be kept confidential and their confidentiality is – in most cases – comparable to the confidentiality of the package itself. Our work in this paper is to refine and extend the protocol of [1] to allow this important service.

4 Our Model and Assumptions

We follow the network topology and communication model of [1]: There is a set of n servers (or package servers) $\mathcal{S} = \{S_1, ..., S_n\}$ which are fully connected via private and authenticated channels. There is a special server S_0 (the gateway, authority or URL) which is responsible for the communication between $\mathcal{S}$ and the users in the network. There are a set of users $\mathcal{U}$ and a special user $u_0 \in \mathcal{U}$ which is the author of the confidential file M. User u_0 is able to deposit his file M to the servers in an efficient, robust and secure way. Only an authorized user in $\mathcal{U}$ is allowed to retrieve and access M from $\mathcal{S}$ through S_0 in a fully authenticated and private way.

The users in $\mathcal{U}$ are not allowed to interact with each other by any means, neither with any of the servers in $\mathcal{S}$, they only interact with the gateway S_0 which communicates any of the users to the servers in $\mathcal{S}$.

We assume the existence of a public key infrastructure (PKI), that each user

$u \in \mathcal{U}$ has his own public/private key pair (pk_u, sk_u) of a public key cryptosystem where every pk_u is registered on the server S_0 with u's identity id_u. Let (pk_S, sk_S) be the public/private key pair assigned for $\mathcal{S}$. The public key pk_S is published to all users in the network, while the private key sk_S is shared among the n servers on a threshold basis using an efficient (t, n)- verifiable distributed key generation protocol where t is the threshold [14]. Our protocol assumes the optimal setting on the number of servers in $\mathcal{S}$ to be $n = 2t + 1$ with at most t of them could behave maliciously. Let sk_{S_i} be the private key share assigned to server S_i of the private key sk_S.

For the purpose of our protocol we also assume that the employed public key cryptosystem has a homomorphic property (e.g. El-gamal cryptosystem) to allow blind decryption of messages.

We assume that the authorized user u who retrieves the file M will not misuse it by any means (e.g. will not redistribute the contents to unauthorized users) or else we are facing a traitor tracing problem that requires watermarking and fingerprint techniques [17, 18]. Although such techniques can be easily integrated with our protocol, the development of such techniques are beyond the scope of this paper.

The authority S_0 is assumed honest-but-curious (or semi-honest) in the sense that, she follows the execution steps of the protocol word for word but she is willing to know and use any sensitive information that could be leaked during execution.

5 Our Basic Tools

This section describes the cryptographic primitives used in our E-DRM protocol.

5.1 El-gamal public key encryption

It is known that the El-gamal cryptosystem [15] works with any family of groups for which the discrete logarithm is considered intractable. Part of the security of the scheme actually relies on the Diffie-Hellman assumption, which implies the hardness of computing discrete logarithms modulo a large prime. We will present our results for subgroups G_q of order q of Z_p^*, where p and q are large primes such that $q|p-1$. Other practical families can be obtained for elliptic curves over finite fields. We will now briefly describe the El-gamal cryptosystem, where the primes p and q and at least one generator g of G_q are treated as system public parameters. The key pair of a receiver in the El-gamal cryptosystem consists of a private key x (randomly chosen by the receiver from Z_q^*) and the corresponding public key $y = g^x$. Given a message $m \in Z_p$, encryption proceeds as follows. The sender chooses a random $r \in Z_q$, and sends the pair $(A, B) = (g^r, y^r m)$ as the ciphertext to the receiving party. To decrypt the ciphertext (A, B) the receiver recovers the plaintext as $m = B/A^x$, using the private key x. The El-gamal encryption is known to be a CPA-secure cryptosystem, there exist extensions to this cryptosystem that achieve CCA1 and CCA2 security.

5.1.1 Homomorphisms and blinding

Let E be an encryption scheme (in this paper we use E to refer to the encryption, the decryption or the digital signature function, the distinction is understood

from the used key in the subscript and the context). Let E_{pk} and E_{sk} be the encryption and the decryption functions respectively. The encryption scheme $\mathcal{E}$ is said to be blindable if it possesses some homomorphic property. For example, The RSA and the El-gamal cryptosystems are $(\times, \times)$-homomorphic [11] that is, $E_{pk}(a)E_{pk}(b) = E_{pk}(ab)$.

Suppose that the ciphertext c (held by the user u) is the encryption of the message m using a public key pk, that is, $c = E_{pk}(m)$ and $m = E_{sk}(c)$. Suppose also that the decryption key sk is held by a server S and that u wants to obtain the decryption of m without allowing S to know neither m nor c. In this case, u simply picks a random integer r, encrypts r as $z = E_{pk}(r)$ and sends the blinded ciphertext $e = zc$ to S. On the other hand, S decrypts e as $mr = E_{sk}(e)$ as usual and returns the blinded plaintext mr back to u from which u is able to obtain m.

5.1.2 El-gamal plaintext equality test

Alice holds the private key x of an El-gamal cryptosystem while the public key $y = g^x$ is publicly known. Now Bob holds two El-gamal ciphertexts, $(A_1, B_1) = (g^{r_1}, y^{r_1}m_1)$ and $(A_2, B_2) = (g^{r_2}, y^{r_2}m_2)$ of plaintexts m_1 and m_2 that he does not know. Both ciphertexts are encrypted using Alice's public key. The only information that Bob wants to know is whether $m_1 \stackrel{?}{=} m_2$.

Trivially, Bob may hand both ciphertexts to Alice who performs the decryption for both ciphertexts and tells Bob the result. However, Bob does not want Alice to know any information about neither m_1 nor m_2. Bob picks a secret blinding integer r_b, computes $(A, B) = ((A_1/A_2)^{r_b}, (B_1/B_2)^{r_b})$. Notice that $(A, B) = (g^{kr_b}, y^{kr_b}(m_1/m_2)^{r_b})$ is an El-gamal encryption of $(m_1/m_2)^{r_b}$ under the public key y, where $k = r_1 - r_2$. Now Bob hands (A, B) to Alice from which Alice decrypts for $(m_1/m_2)^{r_b}$. Obviously, if $m_1 \neq m_2$ then Alice gains no information about neither m_1 nor m_2 since they are kept blinded by r_b, however, if $m_1 = m_2$ the result of the decryption is unity.

5.2 Information dispersal schemes

Consider a large sized file (e.g. multimedia file) M of size $s = |M|$ stored on some server S and that we want to protect this file from an adversary that has the power to compromise this server by either disclosing its contents or destroying it permanently. The privacy of M could be achieved by simply encrypting M as $C = E_k(M)$ (where $|C| = |M| = s$) using any available symmetric cryptosystem and storing the key in a safe place, yet, still the file is vulnerable to destruction!. One may suggest to copy the ciphertext C on multiple servers (say n servers), yet, the amount of occupied memory becomes ns which is of significant concern assuming s is large. Using the well-known perfectly secure Shamir's secret sharing scheme (SSS) to share M [12] will not help reducing the memory usage since, it is well-known that for an SSS to be secure, the size of the share is at least equal to the size of the shared secret which is s and hence, we are still faced with an inefficient storage space (ns).

As long as computational security is of concern, the information dispersal algorithm (IDA) [13] helps to provide efficient storage of information. Given a threshold t and $n > t$ servers $S_1, ..., S_n$, a simple implementation of the IDA algorithm (which is slightly different, yet equivalent to the original scheme) is

as follows: In the disperse phase, partition M into $t+1$ segments, $m_0, ..., m_t$, such that $\forall i$, $m_i < p$ for a selected prime p and a threshold t. Notice that the size of each m_i is $s/(t+1)$. Construct a polynomial $f(x) = \sum_{j=0}^{t} m_j x^j \mod p$. A share $f(i)$ of size $s/(t+1)$ is assigned to server S_i for storage. The memory space occupied by all servers is now, $ns/(t+1)$. In the reconstruction phase, each server S_i publishes his share $f(i)$, the original file M is reconstructed using Lagrange interpolation or matrix elimination method. Finally, notice that the IDA does not provide confidentiality service, yet, confidentiality could be achieved by simply dispersing the encrypted version, C of M instead of M [27].

The information dispersal algorithm introduced in [26] combines the "all-or-nothing-transform" (AONT) of [25] and the IDA of [13] to provide slightly better storage efficiency over that of [13] and also incorporates the Reed-Solomon (RS) codes for error control. The AONT-RS in [25] can straight forward replace the IDA of [13] used in our protocol, however, for a better understanding of our protocol, we apply only the original IDA introduced in [13].

5.3 Secret sharing tools

5.3.1 Secret sharing over a prime field

Let $s \in Z_q$ be a secret held by some dealer where Z_q is a prime field. In order to share this secret among a set $\mathcal{P}= \{P_1, ..., P_n\}$ of $n > t$ players [12], the dealer defines a polynomial $f(x) = \sum_{j=0}^{t} a_j x^j \mod q$, he sets $a_0 = s$ and each other coefficient $a_{j\neq 0} \in_R Z_q$. $\forall i = 1, ..., n$, the dealer secretly delivers $f(i)$ to player P_i. In the reconstruction phase, each player P_i broadcasts $f(i)$, the players are able to compute s from any $t+1$ shares using Lagrange interpolation formula, $s = f(0) = \Sigma_{i\in\mathcal{B}} \lambda_i f(i) \mod q$ where $\mathcal{B} \subset \mathcal{P}$, $|\mathcal{B}|= t+1$ and λ_i is Lagrange coefficient for player P_i.

5.3.2 Verifiable secret sharing

Verifiable secret sharing (VSS) extends polynomial secret sharing in a way that allows the recipients of the shares to verify that their shares are consistent (i.e., that any subset of $t+1$ shares determines the same unique secret). Assuming $n > 2t$, the protocol can tolerate malicious behaviors (e.g., illegal collaboration, sending wrong values, deleting values, etc.) of at most t players. We distinguish two different contributions of VSS; the conditionally secure VSS due to Feldman [21] and the unconditionally secure VSS due to Pedersen [22]. To achieve best security, both of them will be used in our protocol.

Feldman's VSS. Two large primes p and q are chosen such that $q|p-1$. The primes p and q and an element $g \in Z_p^*$ of order q are published as the system public parameters. The dealer shares the secret s among the players on a t-degree polynomial $f(x) = \sum_{j=0}^{t} a_j x^j \mod q$, the dealer also broadcasts the $t+1$ commitments $c_j = g^{a_j} \mod p \ \forall j = 0, ..., t$. These commitments allow each player P_i to verify the consistency of his share $f(i)$ by checking that, $g^{f(i)} = \prod_{j=0}^{t} c_j^{i^j} \mod p$. If this check fails for any share $f(i)$, P_i broadcasts a complaint. If more than t players broadcasted a complaint, then at least one of them is honest and consequently the dealer is disqualified. Otherwise the dealer reveals the share $f(i)$ for each complaining player P_i, if the share is correct, P_i

is disqualified, otherwise, if the share does not satisfy the commitments or if the dealer does not respond, the dealer is disqualified. During reconstruction of the secret, any player can check the validity of the share broadcasted by any other player via the published commitments to filter out bad shares and safely perform the interpolation. When it comes to the distributed generation of a secret key k and the joint computation of $g^k mod p$, Feldman's VSS alone is not secure due to the attacks described in the security analysis section.

Pedersen's VSS. Unlike Feldman's VSS, Pedersen's VSS is secure in the informations theoretic sense. The idea is to use double exponentiation which allows randomization. The public parameters are p, q, g and h where p, q and g are as in Feldman's VSS and h is another element in Z_p^* subject to the condition that $\log_g h$ is unknown and assumed hard to compute. In addition to the polynomial $f(x) = \sum_{j=0}^{t} a_j x^j \mod q$ which holds the secret s as the free term, the dealer sets up a randomizing t-degree polynomial $r(x) = \sum_{j=0}^{t} b_j x^j \mod q$. He secretly delivers $(f(i), r(i))$ to player P_i $\forall i = 1, ..., n$. The dealer also publishes the commitments $c_j = g^{a_j} h^{b_j} \mod p$ $\forall j = 0, ..., t$. Each player P_i verifies the consistency of his share $f(i)$ by checking that, $g^{f(i)} h^{r(i)} = \prod_{j=0}^{t} c_j^{i^j} \mod p$. If this check fails for any share $f(i)$, P_i broadcasts a complaint. If more than t players broadcast a complaint, then at least one of them is honest and consequently the dealer is disqualified. Otherwise the dealer reveals the pair $(f(i), r(i))$ for each complaining player P_i, if the pair is correct, P_i is disqualified, otherwise, if the pair does not satisfy the commitments or if the dealer does not respond, the dealer is disqualified. During reconstruction of the secret, any player can check the validity of the share broadcasted by any other player via the published commitments to filter out bad shares and safely perform the interpolation.

5.3.3 Joint secret sharing

Joint secret sharing allows the players to jointly share some random secret among themselves (in an ad-hoc manner) without the help of the dealer.

Joint random secret sharing (JRSS). JRSS [14,16] allows a set of n players to jointly share a random secret without the help of the dealer. Each player $P_i \in \mathcal{P}$ picks a random integer $k_i \in Z_q$ and plays the role of the dealer to share k_i among the players over a t-degree polynomial $f_i(x) = k_i + \sum_{j=1}^{t} a_j x^j \mod q$. Each player $P_i \in \mathcal{P}$ simply sums the shares he receives from the other players to compute a share $f(i) = \sum_{j=1}^{n} f_j(i)$ which is a point on a t-degree polynomial $f(x)$ with its free term equals a random secret $k = \sum_{i=1}^{n} k_i \mod q$.

Joint random verifiable secret sharing (JRVSS). To withstand malicious behavior of at most $t < n/2$ players during the JRSS, JRVSS combines the JRSS with Feldman's VSS for computational security or Pedersen's VSS for unconditional security. Simply, each player $P_i \in \mathcal{P}$ picks a random secret integer $k_i \in Z_q$ and plays the role of the dealer in the VSS protocol to share this secret among the other players. Complaints are solved as in the VSS protocol. Finally, each player sums what he has to compute his share on a t-degree polynomial, $f(x)$ with its free term $f(0) = \sum_{i=1}^{n} k_i \mod q$.

Joint zero secret sharing (JZSS). JZSS is a special case of the JRSS in which the random secret shared by each player is a zero. At the end of the JZSS, each player holds a share $f(i)$ on a t-degree polynomial $f(x)$ with its free term $f(0) = 0$.

Joint zero verifiable secret sharing (JZVSS). As in the JRVSS, to with stand malicious behavior of at most $t < n/2$ players during the JZSS, JZVSS combines the JZSS with Feldman's VSS for computational security or Pedersen's VSS for unconditional security. Simply, each player $P_i \in \mathcal{P}$ plays the role of the dealer in the VSS protocol to share a zero among the other players. Complaints are solved as in the JRVSS protocol. Finally, each player sums what he has to compute his share on a t-degree polynomial, $f(x)$ with its free term $f(0) = 0$. Notice that, from the published commitments, each player can verify that the shared secret is really a zero. This is true since the commitment to a zero will be $g^0 = 1$.

5.4 Proof of equality of two discrete logarithms

We review the protocol of [23] and also in [24, 28] that is believed to be a zero knowledge proof of equality of two discrete logarithms. In this protocol, the public parameters are two large primes p and q such that $q|p-1$, two elements $\alpha, \beta \in Z_p^*$ and the two quantities $G_1, G_2 \in Z_p^*$. The prover ($\mathcal{P}$) proves to a verifier ($\mathcal{V}$) that he knows $x \in Z_q^*$ such that $G_1 = \alpha^x \mod p$ and $G_2 = \beta^x \mod p$. The protocol is as follows:

- $\mathcal{P} \rightarrow \mathcal{V}$: Choose $r \in_R Z_q^*$ and send $(A = \alpha^r \mod p,\ B = \beta^r \mod p)$.
- $\mathcal{V} \rightarrow \mathcal{P}$: Choose $c \in_R Z_q^*$ and send c.
- $\mathcal{P} \rightarrow \mathcal{V}$: Send $y = r + cx \mod q$.
- $\mathcal{V}$: Check that $\alpha^y = AG_1^c \mod p$ and $\beta^y = BG_2^c \mod p$.

The standard method to convert the above protocol to a non-interactive protocol (we denote it, $\Pi_{LogEq} \leftarrow P_{LogEq}(\alpha, \beta, G_1, G_2, x)$) is by using a sufficiently strong hash function H and setting $c = H(A, B)$. The protocol Π_{LogEq} becomes as follows:

- $\mathcal{P} \rightarrow \mathcal{V}$: Choose $r \in_R Z_q^*$ and send $(A = \alpha^r \mod p,\ B = \beta^r \mod p,\ c = \mathcal{H}(A, B)$ and $y = r + cx \mod q)$.
- $\mathcal{V}$: Check that $\alpha^y = AG_1^c \mod p$ and $\beta^y = BG_2^c \mod p$.

6 Detailed Description of our Protocol

Global parameters setup. Given a security parameter κ, our system works in subgroups G_q (or Z_q^*) of order q in Z_p^* where p and q are large primes and $q|p-1$. Other practical families could be obtained from Elliptic curves over finite fields. The primes p and q and a generator g of G_q are the system global parameters. In number theory, a generator g is picked as follows: Let $p = \mu q + 1$, select $\alpha \in_R Z_p^*$ and $\alpha \neq 1$, if $(\beta = \alpha^\mu \mod p \neq 1)$ then β is a valid generator g of order q since in this case $g^q \mod p = 1$. If $\beta = 1$ (with negligible probability), repeat for another α.

6.1 Setup by the servers

6.1.1 Distributed verifiable private key generation

The set $\mathcal{S} = \{S_1, \ldots, S_n\}$ (with $n = 2t + 1$ for a threshold t) of package servers join together to compute shares of an El-gamal and DSS private key $sk_S = x \in Z_q^*$ of bit-length κ and publicize the corresponding public key $pk_S = y = g^x \mod p$ as follows [14]:

- Run a `JRVSS` with Pedersen's VSS as the VSS in place. At the end, each server S_i holds a share $X(i)$ of a random secret $x \in_R Z_q^*$ over a t-degree polynomial $X(z)$ with $X(0) = x$.
- The servers that are not disqualified in the JRVSS in the previous step broadcast the commitments to their shared polynomial based on Feldman's VSS. More precisely, if $X_i(z) = x_i + \sum_{j=1}^{t} a_j^{(i)} z^j$ is the polynomial of server S_i then S_i broadcasts g^{x_i} and $g^{a_j^{(i)}} \mod p \ \forall j = 1, ..., t$.
- For any server S_i who receives at least one valid complaint in the previous step, the other servers join to reconstruct his polynomial $B_i(z)$ and values g^{x_i} and $g^{a_j^{(i)}} \mod p \ \forall j = 1, ..., t$ in the clear.
- For the remaining n good servers, each server S_i holds a share $X(i) = \sum_{j=1}^{n} X_j(i) \mod q$ of the private key $x = \sum_{i=1}^{n} x_i \mod q$ on a t-degree polynomial $X(z) = x + \sum_{i=1}^{t} a_i z^i \mod q$. Each server also computes the commitments $g^{a_i} = \Pi_{j=1}^{n} g^{a_i^{(j)}} \ \forall i = 0, ..., t$ where $g^{a_0} = g^x = y$.

Finally, each server S_i holds the tuple, $\langle X(i), g^x, g^{a_1}, ..., g^{a_t} \rangle$ which consists of his share $X(i)$ of the private key x, the public key $y = g^x$ and the commitments to the coefficients of the private key polynomial $X(z)$. Notice that $g^{X(i)} = \Pi_{j=0}^{t} g^{a_j i^j} \mod q$.

6.1.2 Sharing a blinding integer

The set of package servers $\mathcal{S} = \{S_1, \ldots, S_n\}$ join together to compute shares of a random blinding integer $b \in Z_q^*$ of bit-length κ as follows:

- Run a `JRVSS` with Pedersen's VSS as the VSS in place. At the end, each server S_i holds a share $B(i)$ of a random secret $b \in_R Z_q^*$ over a t-degree polynomial $B(x)$ with $B(0) = b$.
- The servers that are not disqualified in the JRVSS in the previous step broadcast the commitments to their shared polynomial based on Feldman's VSS. More precisely, if $B_i(x) = b_i + \sum_{j=1}^{t} a_j^{(i)} x^j$ is the polynomial of server S_i then S_i broadcasts g^{b_i} and $g^{a_j^{(i)}} \mod p \ \forall j = 1, ..., t$.
- For any server S_i who receives at least one valid complaint in the previous step, the other servers join to reconstruct his polynomial $B_i(x)$ and values g^{b_i} and $g^{a_j^{(i)}} \mod p \ \forall j = 1, ..., t$ in the clear.

- For the remaining n good servers, each server S_i holds a share $B(i) = \sum_{j=1}^{n} B_j(i) \mod q$ of the blinding integer $b = \sum_{i=1}^{n} b_i \mod q$ on a t-degree polynomial $B(x) = b + \sum_{i=1}^{t} a_i x^i \mod q$. Each server also computes the commitments $g^{a_i} = \Pi_{j=1}^{n} g^{a_i^{(j)}} \ \forall i = 0, ..., t$ where $a_0 = b$.

Finally, each server S_i holds the tuple, $\langle B(i), g^b, g^{a_1}, ..., g^{a_t} \rangle$. Notice that $B(i) = \sum_{j=0}^{t} a_j i^j \mod q$ and hence $g^{B(i)} = \Pi_{j=0}^{t} g^{a_j i^j} \mod p$.

6.2 Package upload and deposit

The author u_0 possessing a file/package M with search keyword(s) w, provided with his personal public/private key pair (pk_{u_0}, sk_{u_0}) and the public key pk_S of the set of servers $\mathcal{S}$, performs as follows:

- Picks a random symmetric secret key k for a secure symmetric cryptosystem.
- Using k, encrypts M as $C_M = E_k(M)$.
- Using pk_S of $\mathcal{S}$, encrypts k as $C_k = E_{pk_S}(k)$.
- Using pk_S, encrypts each w as $C_w = E_{pk_S}(w)$.
- Using sk_{u_0}, generates the signature $\sigma_{u_0} = E_{sk_{u_0}}(H(C_M, C_k, C_w, id_{u_0}, id_f))$.
- Sends to the authority server S_0 the tuple, $T_{u_0} = \langle C_M, C_k, C_w, id_{u_0}, id_f, \sigma_{u_0} \rangle$

At this point, the author u_0 has completed uploading his file and keywords.

- On the reception of T_{u_0}, the authority S_0 uses pk_{u_0} to verify the signature σ_{u_0} and the identity id_{u_0}.
- S_0 sends C_M, C_k, C_w and id_f to all servers in $\mathcal{S}$. Notice that the forwarded message must be signed by S_0's personal private key for authenticity purpose.
- S_0 also signs a receipt back to u_0 using her private signature key. For more discussion about issuing receipts and digital signatures, please refer to section 9.

6.3 Package dispersal

Next, each server in $\mathcal{S}$ runs the IDA algorithm, the servers agree on a dispersal strategy (i.e. the prime p and the polynomial $f(x)$), each server S_i performs as follows in a non-interactive way:

- Segments C_M into $t+1$ segments $(C_M^{(0)}, ..., C_M^{(t)})$.
- Sets the polynomial $f(x) = \sum_{\ell=0}^{t} C_M^{(\ell)} x^\ell \mod p$.
- Computes $f(j)$ and $H(f(j)) \ \forall j = 1, ...n$.
- Stores $f(i)$ and $H(f(j)) \ \forall j = 1, ..., n$. Erases all other $f(j)$ shares.

The IDA algorithm is performed only for the file ciphertext C_M, as it is assumed a large ciphertext. Applying the IDA algorithm on the symmetric secret key ciphertext C_k and the keyword ciphertext C_w does not worth the effort since they are already small (usually one group element). Therefore, each server in $\mathcal{S}$ stores a copy of C_k and C_w as they are.

Finally, each server $S_i \in \mathcal{S}$ ends up storing for each package the tuple, $\langle f(i), H(f(1)), ..., H(f(n)), C_k, C_w, id_f \rangle$.

6.4 Packages search and request

A user u wants to search the packages stored on the servers and to request the package with a keyword w'. u performs as follows (in a one-move non-interactive way):

- Encrypts w' using $\mathcal{S}$'s public key as $C_{w'} = (A_{w'}, B_{w'}) = (g^{r'}, w'y^{r'})$.
- Computes $H(C_{w'}, id_u)$ and signs using his personal private key as $\sigma_u = E_{sk_u}(H(C_{w'}, id_u))$.
- Sends to S_0 the tuple $\langle C_{w'}, id_u, \sigma_u \rangle$.

After checking σ_u by S_0, S_0 determines the identities of the files (id_f's) that id_u is authorized for, then S_0 forwards the files identities and $C_{w'}$ to all servers in $\mathcal{S}$ signed with her personal private key for authenticity purpose. Each server locally computes $\gamma = C_w/C_{w'} = (\alpha, \beta) = (A_w/A_{w'}, B_w/B_{w'})$. The servers in $\mathcal{S}$ proceed to blind γ as $C = \gamma^b$. Recall that $C = (A, B)$ is an El-gamal encryption where $A = \alpha^b$ and $B = \beta^b$. Also, recall that each server $S_i \in \mathcal{S}$ holds a share $B(i)$ of the blinding integer b and commitments to the polynomial coefficients. The servers join to compute (α^b, β^b) as follows:

- Each server S_i computes and broadcasts to other servers in $\mathcal{S}$, $\alpha^{B(i)}$ and the proof $\Pi_{LogEq} \leftarrow P_{LogEq}(\alpha, g, \alpha^{B(i)}, g^{B(i)}, B(i))$.
- Each server S_i computes and broadcasts to other servers in $\mathcal{S}$, $\beta^{B(i)}$ and the proof $\Pi_{LogEq} \leftarrow P_{LogEq}(\beta, g, \beta^{B(i)}, g^{B(i)}, B(i))$.
- From any valid $t+1$ pairs $(\alpha^{B(i)}, \beta^{B(i)})$, each server S_j performs interpolation in the exponent to compute $(\alpha^b, \beta^b) = (\alpha^{\sum_{i \in \mathcal{C}} B(i)\lambda_i}, \beta^{\sum_{i \in \mathcal{C}} B(i)\lambda_i})$ where $|\mathcal{C}| = t + 1$ and λ_i is Lagrange coefficient.

The decryption of C is B/A^x. Recall that each server S_i holds a share $X(i)$ of the secret key x and commitments to the polynomial coefficients, the servers then join to decrypt C in a threshold way as follows:

- Each server $S_i \in \mathcal{S}$ computes and broadcasts to other servers $A^{X(i)}$ and a proof $\Pi_{LogEq} \leftarrow P_{LogEq}(A, g, A^{X(i)}, g^{X(i)}, X(i))$.
- From any valid $t+1$ quantities $A^{X(i)}$, each server S_j performs interpolation in the exponent to compute $A^x = A^{\sum_{i \in \mathcal{C}} X(i)\lambda_i}$ where $|\mathcal{C}| = t + 1$ and λ_i is Lagrange coefficients.
- Finally, the servers compute $(w/w')^b = B/A^x$ and check for unity.

The above protocol is repeated for every keyword w and w'. For those keywords w that match the user u's keyword w' (i.e. the check results for unity), the servers send the corresponding file identities id_f's to S_0 who forwards them to user u as the search results. Notice that all good servers (the majority) will agree on these identities.

The user u requests a package/file id_f (if exists a match) as follows:

- u picks a blinding integer r and using $\mathcal{S}$'s pk_S encrypts r as $C_r = E_{pk_S}(r)$.
- u prepares a request for the file id_f he wants to download and computes $H(req, id_u, id_f, C_r)$.
- Using his private key sk_u, u signs his request as $\sigma_u = E_{sk_u}(H(req, id_u, id_f, C_r))$ and sends to S_0 the tuple, $T_u = \langle req, id_u, id_f, C_r, \sigma_u \rangle$.
- On the reception of T_u, S_0 verifies the signature and ensures that id_u is authorized for id_f.
- S_0 forwards a signed version of id_f and C_r to all servers in $\mathcal{S}$.

Remark. We emphasize that S_0 is assumed semi-honest and follows the execution steps as written. Therefore, S_0 will not alter C_r created by the user u or else, the signature by u on C_r must be forwarded to be verified by all servers in $\mathcal{S}$. Time-stamp embedded with the signature disables the possibility of replay attacks.

6.5 Package retrieval

Next, the set of servers $\mathcal{S}$ and the authority S_0 reconstruct C_M of identity id_f and decrypt for the blinded symmetric secret key rk as follows:

Retrieval of C_M:

- To S_0, each server $S_i \in \mathcal{S}$ sends his segment $f(i)$ and the hashes $H(f(j))$ $\forall j = 1, ..., n$ signed by his personal private key.
- S_0 collects the segments, hashes them and decides on their integrity by comparison to all other received hashes (on a majority basis).
- From any valid $t + 1$ segments, S_0 is able to recover C_M.

Retrieval of the symmetric secret key:

- Each server $S_i \in \mathcal{S}$ computes the ciphertext, $C_{rk} = C_r C_k$.
- The set $\mathcal{S}$ runs a verifiable distributed threshold decryption protocol to decrypt C_{rk}. At the end, each server $S_i \in \mathcal{S}$ holds a partial decryption ε_{S_i} of $E_{sk_S}(C_{rk})$.
- Each S_i signs his partial decryption ε_{S_i} using his personal private key.
- The signed partial decryptions are sent to S_0 to reconstruct the blinded key, rk.
- S_0 sends C_M and rk to the user u signed with S_0's private key. u verifies the signature, extracts k from rk (since he knows r) and decrypts for M.

7 Security Analysis

Privacy against the authority. The semi-honest authority S_0 receives the ciphertext $C_M = E_k(M)$ and an encryption of k, $C_k = E_{pk_S}(k)$ under $\mathcal{S}$'s public key pk_S. The corresponding decryption key sk_S is unknown to S_0, yet, it is shared among $\mathcal{S}$ on a threshold basis. A computationally bounded S_0 is unable to know k. On the other hand, in the file request phase, S_0 receives an encryption $C_r = E_{pk_S}(r)$ from the user u, again, since the corresponding decryption key sk_S is unknown to S_0, S_0 gains no information about r. In the file retrieval phase of the protocol, S_0 receives (collects) a blinded decryption of $C_{rk} = E_{pk_S}(rk)$ as rk. Since S_0 does not know neither r nor k, S_0 gains no information about k and hence, M is secure against S_0.

Assuming that S_0 is semi-honest is essential since a malicious S_0 may replace the encryption $C_r = E_{pk_S}(r)$ with another value $C_{r'} = E_{pk_S}(r')$ where she knows r' and since the servers return $r'k$, she is able to know the private key k.

We therefore state the following lemma.

Lemma 1. *Assuming that the public key cryptosystem is a secure homomorphic cryptosystem and that the symmetric cryptosystem is secure, S_0 gains no information about the package content M in the cryptographic sense.*

Privacy against $\mathcal{S}$. The information collected by the set $\mathcal{S}$ during the execution of our protocol are $C_M, C_k, C_w, C_{w'}$ and C_r. From the threshold security of threshold decryptions, no coalition of t or less servers are able to decrypt for C_r, C_k , $C_{w'}$ or C_w. The decryption is performed only on the blinded version of the keyword relative ciphertext C (which is a blinded equality test of the keywords) and the product $C_rC_k = C_{rk} = E_{pk_S}(rk)$ and consequently, only rk is known to the servers with no information revealed about neither k nor r. Hence M is kept private from the set of servers $\mathcal{S}$. We then state the following lemma.

Lemma 2. *Assuming that the threshold public key cryptosystem is a t-resilient homomorphic encryption, none of the servers in $\mathcal{S}$ gains no information about M, w or w' in the cryptographic sense.*

Malicious behaviors. We emphasis that, the user u downloading the file is assumed to be honest about not distributing the clear-text M to other unauthorized users, or else she becomes a pirate and we are facing a traitor tracing problem requiring embedded watermarking and fingerprint codes. The *verifiability* property of distributed threshold decryption and digital signature schemes ensures that a coalition of at most t malicious servers will always be tolerated by the scheme and that at most t malicious servers will not be able to forge a signature. Hence, the following lemma holds.

Lemma 3. *Assuming that the distributed threshold public key cryptosystem is verifiable and t-resilient, then at most t malicious servers will always be tolerated.*

8 Efficiency and Complexity Evaluation

Complexity of the author. Considering communication complexity, the only interaction of u_0 is the communication with the authority S_0 during the *File*

uploading and later while receiving the *receipt*. During file uploading, the author transmits the tuple $T_{u_0} = \langle C_M, C_k, C_w, id_{u_0}, id_f, \sigma_{u_0} \rangle$, the size of id_{u_0} and id_f are short strings while σ_{u_0}, C_k and $C_w = (A_w, B_w)$ are only four group elements. The receipt is only one group element σ_S transmitted from S_0 back to u_0.

In the computation complexity, for file uploading, u_0 performs a symmetric encryption of M as C_M which is computationally trivial. u_0 also performs four group computations as: The encryption for C_w, a public key encryption on k as $E_{pk_S}(k)$ and his own signature as σ_{u_0} which are two group operations on one block of data: the key k and the hash $H(C_M, C_k, C_w, id_{u_0}, id_f)$. We conclude that the complexity of the author u_0 is $|M| + \mathcal{O}(1)$.

Complexity of the authority. Considering communication complexity, the interaction with u_0 was already discussed. In the dispersal and deposit phase, S_0 sends C_M, C_k, C_w and id_f to all servers in $\mathcal{S}$ with her personal signature on them which is in the order of $|M|$. In the package request phase, S_0 receives the tuple $\langle C_{w'}, id_u, \sigma_u \rangle$ consisting of three group elements and a short string. Also, S_0 receives the tuple $T_u = \langle req, id_u, id_f, C_r, \sigma_u \rangle$ consisting of short strings plus two group elements. During the retrieval of C_M, S_0 receives the segments $f(1), ..., f(n)$ totaling a size $n|M|/(t+1)$ and almost n^2 short hash strings of these segments. In the retrieval of the symmetric secret key k, S_0 receives n signed partial decryptions $\varepsilon_{S_1}, ..., \varepsilon_{S_n}$ each of two group elements. S_0 sends C_M of size M and one group element rk to user u side by side with his personal signature on them which is a one group element.

In the computation complexity, during package deposit, S_0 performs one signature verification and one digital signature. During the retrieval of package ciphertext C_M, S_0 performs n hash operations and one polynomial reconstruction. During the retrieval of the secret key, S_0 performs threshold decryption reconstruction in the order of $\mathcal{O}(n)$ modulo operations and one personal signature.

Complexity of the user. During file request, performs one public key encryption as $C_{w'}$, one public key encryption to generate C_r and one digital signature. After retrieval of the secret key, performs one modulo inverse/multiplication and a symmetric decryption for M. The complexity is therefore roughly $|M| + \mathcal{O}(1)$.

9 Notes on Digital Signatures and Receipts

Our protocol assumes that the authority is semi-honest (honest-but-curious) in the sense that, it follows the execution steps of the protocol word for word (as written) but she is willing to learn and misuse any private information that could be leaked during execution. This assumption reduces the complexity by allowing the authority to issue digital signatures on behalf of the servers to authenticate the servers to the users and also to issue receipts to the author. If the authority is assumed malicious, then she is not trusted to sign information and hence all the digital signatures issued in our protocol are insecure. In this case, digital signatures are to be performed by the servers in $\mathcal{S}$ on a threshold basis using threshold distributed verifiable digital signature schemes (e.g. [16]). Therefore the computation complexity of the servers dramatically increases and the number of servers may need to be extended accordingly. For example, the

threshold DSS scheme of [16] requires a number of servers $n > 4t$ with at most t of them may behave maliciously. This lower bound on n can be reduced to $n > 3t$ and further to $n > 2t$ but with extensive increase in communications and computations complexities. In some situations, even if the authority is assumed honest, the authority may not trust the servers (e.g. the authority and the servers are from different organizations), in the sense that , she does not issue any receipts to the author unless she is sure that his package is dispersed on the servers. In this case, the servers must jointly sign a receipt for the package ciphertext to the authority. Regardless of the above circumstances, the assumption that the authority is semi-honest and not malicious is essential for the security of our protocol.

10 Conclusions

In this paper, we inspected several recent protocols that provide cryptographic solutions to E-DRM. We then refined and extended the recent E-DRM protocol of [1] for better efficiency in retrieving files of very large sizes. Our protocol allows the user to search the encrypted archived contents for private keywords, then to download and decrypt only those packages with contents matching the user's desired keywords. The protocol is robust, secure, provides efficient storage of large files and reduces the computation and communication burden on the author, the authority as well as the users.

References

[1] M. H. Ibrahim, Secure and robust enterprise digital rights management protocol with efficient storage, International Journal on Information (Information-Tokyo), Tokyo, Japan, Vol. 18, No.2, pp. 625-640, 2015.

[2] C. L. Chen, Y. Y. Chen, and Y. H. Chen, Group-based authentication to protect digital content for business applications, Int. J. of Innovative Computing, Information and Control, Vol. 5, No. 5, pp. 1243–1251, 2009.

[3] Y. Yang, and T. Chiueh, Enterprise digital rights management: Solutions against information theft by insiders, RPE report TR-169, Stony Brook Univ. 2004.

[4] A. Arnab and A. Hutchison, Digital rights management a current review, Departmental technical report, no. CS-04-00, University of Cape Town, 2004.

[5] A. Arnab and A. Hutchison, Digital Rights Management - An overview of Current Challenges and Solutions, in Proc. of ISSA, South Africa, 2004.

[6] A. Arnab and A. Hutchison, Requirement Analysis of Enterprise DRM systems. In Proc. of ISSA, Johannesburg, South Africa, 2005.

[7] F. Bartolini, V. Cappellini, A. Piva, A. Fringuelli, and M. Barni, Electronic Copyright Management Systems: Requirements, Players and Technologies, in proceedings of DEXA'99, Florence, Italy, pp.896–898, 1999.

[8] D. Mulligan, J. Han, and A. Burstein, How DRM Based Content Delivery Systems Disrupt Expectations of Personal Use, ACM workshop on Digital Rights Management, pp.77-89, 2003.

[9] J. Park, R. Sandhu, and J. Schifalacqua, Security architectures for controlled digital information dissemination, ACSAC '00, USA, pp. 224–233, 2000.

[10] Z. Zhang, B. Fang, M. Hu and H. Zhang, Security of Session Initiation Potocol, Int. J. of Innovative Computing, Information and Control, Vol. 3, No. 2, pp. 457–469, 2007.

[11] A. J. Menezes, P. C. Orschot, and S. A. Vanstone, Hand-book of Applied Cryptography, CRC Press, 1996.

[12] A. Shamir, How to Share a Secret, Communications of the ACM, Vol.22, No.11, pp.612–613, 1979.

[13] Michael O. Rabin, Efficient dispersal of information for security, load balancing, and fault tolerance, Journal of the ACM, 36(2), pp. 335–348, 1989.

[14] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, Secure Distributed Key Generation for Discrete-Log Based Cryptosystems, J. of Cryptology 20(1), pp. 51–83, 2007.

[15] T. Elgamal, A public key cryptosystem and a signature scheme based on discrete logarithms, IEEE Trans. on Information Theory 31(4): 469-472, 1985.

[16] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, Robust Threshold DSS Signatures, in proceedings of EUROCRYPT'96, pp. 354–371, 1996.

[17] G. Tardos, Optimal probabilistic fingerprint codes, JACM, 55(2), 2008.

[18] D. Boesten and Boris kori, Asymptotic fingerprinting capacity in the Combined Digit Model, in Information Hiding, Springer, pp. 255–268, 2013.

[19] C. Chin-Chen, and J. Yang, A Group-oriented Digital Right Management Scheme with Reliable and Flexible Access Policies, Int.l J. of Network Security, Vol.15, No.6, pp. 471–477, Nov. 2013.

[20] Z. Zhang, B. Fang, M. Hu and H. Zhang, Security of Session Initiation Potocol, International Journal of Innovative Computing, Information and Control, Vol. 3, No. 2, pp. 457–469, 2007.

[21] P. Feldman, A Practical Scheme for Non Interactive Verifiable Secret Sharing, In Proc. of the 28th IEEE Symposium on the Foundations of Computer Science, pp. 427-437, 1987.

[22] T. Pedersen, Non-interactive and information theoretic secure verifiable secret sharing, Crypto '91, LNCS 576, pp. 129-140, 1992.

[23] C. C. P. Schnorr, Efficient signature generation by smart cards, Journal of Cryptology, vol. 4, no. 3, pp. 161-174, 1991.

[24] E. D. Chaum, and T.P. Pedersen, Wallet databases with observers, Advances in Cryptology - Crypto '92, pp. 89-105, 1992.

[25] R. L. Rivest, All-or-nothing encryption and the package transform, Lecture Notes in Computer Science Volume 1267, pp 210-218, 1997.

[26] K. Jason and, S. J. Plank, AONT-RS: Blending Security and Performance in Dispersed Storage Systems, in 9th USENIX, pp. 14-14, 2011.

[27] H. Krawczyk, Secret Sharing Made Short, CRYPTO '93, pp. 136-146, 1993.

[28] M. H. Ibrahim, Resisting Traitors in Linkable Democratic Group Signatures, Int. J. of Network Security, Vol. 9, No. 1, pp. 51-60, 2009.