February, 2015

Secure and Robust Enterprise Digital Rights Management Protocol with Efficient Storage

Maged Ibrahim, *Helwan University*

Secure and Robust Enterprise Digital Rights Management Protocol with Efficient Storage

ARTICLE *in* INTERNATIONAL JOURNAL ON INFORMATION · FEBRUARY 2015

Impact Factor: 0.36

1 AUTHOR:

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

SEE PROFILE

Draft Version

Secure and Robust Enterprise Digital Rights Management Protocol with Efficient Storage

Maged Hamada Ibrahim

Department of Electronics, Communications and Computers, Faculty of Engineering, Helwan University
1, Sherif St., Helwan, Cairo, P.O. Box 11792, Egypt
E-mail: mhii72@gmail.com

Abstract

In the digital world, protection against information theft by unauthorized entities is the most demanding services for enterprise business organizations. Although firewalls and intrusion detection systems are able to prevent outsider attacks (i.e. attacks attempted by individuals outside the organization), still insider attackers are the most serious threat. Enterprise Digital Rights Management (E-DRM) are strategies and schemes to protect sensitive information by managing and enforcing access and usage rights to the information throughout its life-cycle, no matter where the information is distributed. Among the recently proposed cryptographic solutions to E-DRM, we noticed several efficiency drawbacks and security weaknesses. In this paper we device a stronger digital rights management protocol for enterprise applications that overcomes the security problems and efficiency drawbacks in previous protocols. Our proposed protocol satisfies the requirements for an E-DRM protocol, provides more efficient and robust storage of large digital packages/files and provides stronger security against corruptive adversaries. Our proposed protocol ensures privacy against the authorization authority and the unauthorized users while reduces the computations and communications burden of the author, the authorization authority as well as the users.

Key Words: Enterprise security, Digital rights management, Threshold cryptography, Information dispersal, Homomorphic encryption, Robust and efficient storage.

1. Introduction

Secure and efficient storage and processing of information in its digital format (e.g. product overviews, marketing plans, customer lists, sales reports, software packages, multimedia, etc.) are becoming one of the main interests of many organizations such as financial institutions, government agencies and many other business applications. Organizations usually share those digital files/packages from protected file servers and distribute them by variety of download methods. Unfortunately, digital format, in spite of its efficiency in distribution and sharing, makes information more vulnerable to unauthorized disclosure and malicious use by unauthorized entities, either accidentally or deliberately.

Firewalls, proxies and other intrusion detection systems may help in protecting digital information from being disclosed to outsiders (those unauthorized users outside the organization). However, they do not help in controlling the handling of sensitive information by users (e.g. employees) inside the organization and do not protect against the handling of information by

unauthorized insiders.

A more comprehensive and practical solution is Enterprise Digital Rights Management (E-DRM) [1, 2], which is a category of information protection techniques that aim to manage rights to digital intellectual properties and help organizations protect sensitive information from unauthorized use. E-DRM solutions provide information owners the capability to specify fine-grained rights and to enforce these rights at the time when the files are accessed.

Some of the requirements of a well-designed DRM system have been presented by Arnab et al. [3, 4, 5], Bartolini et al. [6], Mulligan et al. [7], and Park et al. [8]. According to the E-DRM basic requirements, a good E-DRM system should meet the following requirements. *Integrity:* It should be possible to verify that a message receipt has not been modified in transit; an intruder cannot substitute a false message for a legitimate one. *Authentication:* It should be possible to ascertain a message receipt's origin; an intruder should not be able to masquerade as someone else. *Persistent protection:* Regardless of the location of the digital data, the access controls that are imposed by the rights holder must be enforced, or the device should not be able to read the file at all. *Track usage of DRM work:* The use of confidential data should be monitored; and should security be compromised, access logs and usage patterns should be made available to track down the source of the compromise. *Time-shifting*: The ability of the user to access the work whenever he or she wants. *Space-shifting:* The ability of the user to freely access the work in whichever device he or she wants.

In the recently proposed E-DRM protocols [1, 9, 10] we observed several security and efficiency issues. In this paper we device a stronger digital rights management protocol for enterprise applications that overcomes the security problems and efficiency drawbacks (discussed in subsection 3.1) found in previous protocols. Our proposed protocol in addition of satisfying the basic requirements for an E-DRM protocol, provides more efficient and robust storage of large digital packages/files and provides stronger security against corruptive adversaries. Our proposed protocol ensures privacy against the authorization authority and the unauthorized users and reduces the computations and communications burden of the author, the authorization authority as well as the users.

2. Related Work

In the E-DRM protocol by Chen et al. [1] there are four entities: the author of the digital content, the package server, a group of n authorization authorities and the user who wants to access the digital content. The package server is responsible for encrypting the digital content which is packed into E-DRM format file. In addition, the group of the authorization authorities is responsible for authenticating the user's access right for the digital content. In the following, we briefly describe their protocol which consists of two main phases.

In the *package uploading phase:* In step 1, the author creates the digital content (file or package) M and sends it to the package server. In step 2, the package server generates a symmetric key k to

encrypt M as $C_M = E_k(M)$ then; the package server generates a digital signature on the ciphertext C_M and uploads to a public directory. In step 3, the package server uses a (t, n) polynomial secret sharing to share the symmetric key k among the authorities on a threshold basis (where t is a certain threshold) and secretly delivers each authority her share/shadow. In the *license acquiring and downloading phase*: Any user is able to download the encrypted file from the public directory and verifies the server's digital signature. Still the user needs the symmetric key k to decrypt the file. The user sends a request to the authorities signed with his personal private key. After verification, the authorities secretly send her share of k to the user signed with her personal private key. From any $t + 1$ shares, the user is able to recover the symmetric key and decrypts the file.

Recently, Chin-Chen Chang and Jen-Ho Yang [9] noticed that although the symmetric key k is shared among the authorities on a threshold basis, the privacy of the secret key is still vulnerable by any coalition exceeding the threshold, They suggested that the server splits the symmetric key into two parts, say k_1 and k_2. k_1 is (t', n') shared among a group of n' users while k_2 is (t, n) shared among the authorities. In this case, a coalition of authorities still gains no information about the secret key k. It is required at least $t + 1$ authorities to recover k_1 and at least $t' + 1$ users to recover k_2 then both k_1 and k_2 are used in order to recover k. For more details on the number theoretic settings of the scheme, please refer to [9].

3. Motivations and Contributions

3.1 Motivations

The work in this paper is motivated by the observation that, the recently proposed E-DRM protocols [1, 9, 10] are vulnerable to several efficiency drawbacks and security weaknesses discussed next.

The author sends his package, M, in clear-text to the package server which makes it vulnerable to eavesdropping attacks, unless the link is physically secured. The symmetric secret keys of all packages and the encrypted packages themselves are stored on one sever making them vulnerable to either being known by an eavesdropping adversary or being destructed (corrupted or lost) by a corruptive adversary which is able to halt or malfunction the server. Notice that trivial replication of the server is not an efficient idea since, the packages (especially multimedia) are usually of very large sizes and hence, storage issues become very significant. Also, replicating private information (the symmetric encryption keys) makes its vulnerability to eavesdropping and theft attacks worse. Moreover, in the previously proposed protocols [1, 9], the user (package down-loader) is required to interact with a number of authorities to collect the shares of the symmetric key and to verify their signatures; such a concurrent multi-authority interaction is quite a burden on the users.

3.2 Our Contributions

Our contribution in this paper is to device a stronger digital rights management protocol for enterprise applications (abbreviated SDRMP) that overcomes the security problems and efficiency

drawbacks mentioned above. We consider a different network topology and communication model. We focus on reducing 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. We show that our protocol is secure against malicious behavior of a minority of the servers. Our protocol ensures privacy against the authorization authority and unauthorized users.

4. Our Model and Assumptions

4.1 Communication Model

In our proposed model, as shown in Figure 1, there is a set of n servers (or package servers), $\mathbf{S}=\{S_1, \dots S_n\}$ which are fully connected via private and authenticated channels. There is a special server S_0 (the gateway, authority or URL) that is responsible for the communication between $\mathbf{S}$ and the users in the network. There are a set of users $\mathbf{U}$ and a special user $u_0 \in \mathbf{U}$ who is the author of the confidential file M. User u_0 is able to deposit his file M to the servers in an efficient and confidential way. Only an authorized user in $\mathbf{U}$ is allowed to retrieve and access M from $\mathbf{S}$ through S_0 in a fully authenticated and private way. The users in $\mathbf{U}$ are not allowed to interact with each other by any means, neither with any of the servers in $\mathbf{S}$, they only interact with the gateway S_0 which communicates any of the users to the servers $\mathbf{S}$.

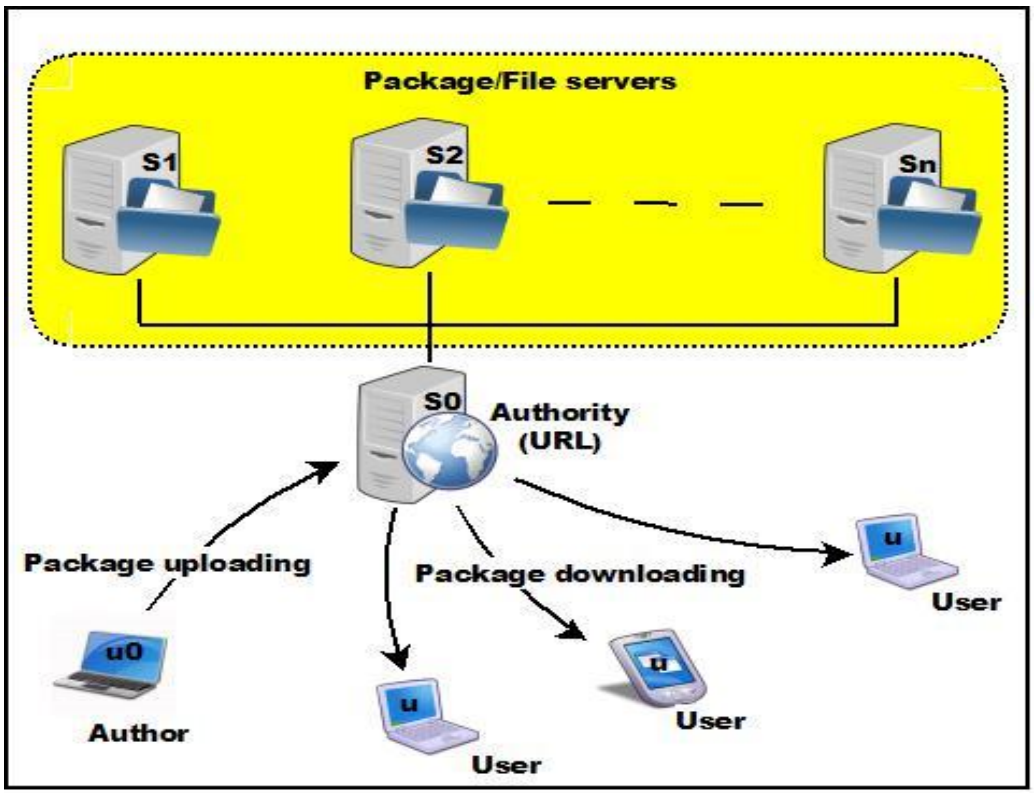

Fig. 1, Communication Model for our SDRMP

4.2 Public Key Model and Infrastructure

We assume that each user $u \in \mathbf{U}$ 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. Also, each server $S_i \in \mathbf{S}$ is equipped with his own personal public/private keypair.

Let (pk_s, sk_s) be the public/private key pair assigned for $\mathbf{S}$. 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)- fully distributed threshold key generation protocol where t is the threshold [14]. Let sk_{S_i} be the 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. RSA or Elgamal) to allow blind decryption of messages.

4.3 Adversary Model

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 [19, 20] or employing *display-only file-servers* (DOFS) [2]. However, although such techniques can be easily integrated with our protocol, 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. Hence, privacy against the authority is mandatory. Given a threshold t, we assume that at least $t + 1$ out of the n servers in $\mathbf{S}$ behave honestly while at most t of the servers in $\mathbf{S}$ could behave maliciously.

5. Basic Cryptographic Tools

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

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 hence, $m = E_{sk}(C)$. 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 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.

5.2 Information Dispersal Algorithm

Consider a large sized file *M* (e.g. multimedia file) 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 $C = E_k(M)$ using any secure symmetric cryptosystem (where $|C| = |M| = s$) and storing the key k in a safe place, yet, still the file is vulnerable to corruption! One may suggest copying the ciphertext *C* on multiple servers (say *n* servers) where 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 still *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] will help to provide efficient storage of information. Given a threshold t and $n > t$ servers, $S_1, \dots, S_n$, a simple implementation of the IDA algorithm (which is slightly different but equivalent to the original scheme) is as follows: In the *disperse phase*, partition *M* into $t + 1$ segments $(m_0, \dots, m_t)$, such that, $m_i < p \ \forall i$ for a selected prime p and a threshold t. Notice that the size of each m_i is $s/(t+1)$. The algorithm proceeds by constructing a polynomial $f(x) = \sum_{j=0}^{t} m_j x^j \bmod p$. 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 $ns/(t+1)$ instead of the inefficient *ns*. 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, we achieve confidentiality by simply dispersing the encrypted version, *C* of *M* instead of *M*.

5.3 Threshold Public Key Cryptosystems

In public key cryptography (either encryption or digital signatures mechanisms), the privacy and integrity of the private key is an important issue. Threshold distributed public key cryptosystems uses polynomial secret sharing [12] as a basic tool with the help of verifiable secret sharing tools [24, 25] and allow the private key to be distributed (shared) among a number of servers on a threshold and verifiable basis to ensure several security issues [14, 17, 18], some of which are: An adversary attacking a minority of the servers gains no information about the private key. A minority of malicious servers cannot disrupt the decryption or signature process. They prevent unauthorized use of sensitive encrypted data and unauthorized issuing of digital signatures.

Let the n servers $S_1, \dots, S_n$ hold shares $sk_{S_1}, \dots, sk_{S_n}$ of the private key sk_s while the corresponding public key pk_s is already published. Let t be a threshold satisfying the condition $n > 2t$.

A *distributed threshold decryption protocol* takes a ciphertext $C = E_{pk_s}(M)$ as an input, allows each server S_i to generate a partial decryption $\varepsilon_{S_i} = E_{sk_{S_i}}(C)$ and then, given at least $t+1$ valid partial decryptions, combines the partial decryptions to output the decrypted message *M*.

A *distributed threshold signature protocol* takes a message *M* as an input, allows each server S_i to generate a partial signature $\sigma_{S_i} = E_{sk_{S_i}}(H(M))$ and then, given at least $t+1$ valid partial signatures, combines the partial signatures to output the signature σ on *M*.

Threshold public key cryptography improves the security of the private key sk_s against malicious behavior (altering, deletion, erasure, crashes, misuse, etc.) of the servers and to omit the need to give full trust to a single server, the trust is distributed among the *n* servers. For more details and information about this field of research, the reader may refer to [14, 15, 16, 17, 18]. However, in our SDRM protocol we assume that the servers in **S** have run the *fully distributed threshold key generation protocol* in [14] and that they perform threshold decryptions and digital signatures according to the robust threshold DSS and El-Gamal decryption in [16]. Of course, other cryptosystems may be used but, we choose the schemes in [14, 16] due to their computation optimality.

6. Detailed Description of our SDRM Protocol

In this section, we present the detailed description of our SDRMP. To help the reader to follow our protocol description, the important notations are listed in Table 1.

Table 1: Important notations used in our SDRM Protocol

S_0	The authorization authority, gateway or URL	C_k	The encryption of *k* under pk_s
S	The set $\{S_1, \ldots S_n\}$ of package servers	C_r	The encryption of a nonce *r* under pk_s
u_0	The author responsible for package uploading	C_{rk}	Blinded encryption of *k*
u	A user requesting package for download	sk_{S_i}	The share of sk_s held by server S_i
E	Encryption/Decryption/Signature function	pk_{u_0}/sk_{u_0}	The public/private key pair of u_0
H	A strong hash function	pk_u/sk_u	The public/private key pair of *u*
M	The plaintext package or file to be uploaded	id_{u_0}, id_u, id_f	Identities of u_0, *u* and *M* respectively
pk_s/sk_s	The public/private key pair for the set **S**	σ_{S_i}	Partial signature of server S_i
C_M	The ciphertext of *M*	ε_{S_i}	Partial decryption held by server S_i
k	The symmetric secret key for encrypting *M*		

6.1 Initialization and Setup

All parties in our SDRMP initialize as follows:

- Each party is equipped with her own personal public/private keypair. This can be acheived by assuming the existance of an enterprise PKI.
- The servers in **S** run a *fully distributed verifiable threshold key distribution protocol* (e.g. the threshold key distribution in [14] for ElGamal and DSS and the robust threshold signature/decryption in [16]), at the end, each server $S_i \in$ **S** holds a share sk_{S_i} of the private key sk_s while the corresponding public key pk_s is publicly known to all

parties. *We emphasize that, the full private key sk_s is unknown to any of the parties in the protocol.*

6.2 Package Uploading by the Author

The author u_0 possessing a file/package *M*, provided with his own personal public/private keypair, (pk_{u_0}, sk_{u_0}) and the public key pk_s of the set of servers **S** performs as follows:

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

At this point, the author u_0 has completed uploading his package.

6.3 Package Dispersal and Deposit among the Servers by S_0

- On the reception of T_{u_0}, 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 and id_f to all servers in **S**. Notice that the forwarded message must be signed by S_0's personal private key sk_{S_0} for authenticity purpose.
- Next, each server in **S** runs the IDA algorithm, the servers agree on a dispersal strategy (i.e the polynomial $f(x)$). Each server S_i performs as follows in a non-interactive way:
  - Segments C_M into $t+1$ segments $(C_M^{(0)}, \dots, C_M^{(t)})$.
  - Set the polynomial $f(x) = \sum_{l=0}^{t} C_M^{(l)} x^l \bmod p$.
  - Computes $f(j)$ and $H(f(j)), \forall j = 1, \dots, n$.
  - Stores $f(i)$ and $H(f(j)), \forall j = 1, \dots, n$. Erases all other $f(j)$ shares for $j \neq i$.

The IDA algorithm is performed only for the package ciphertext C_M, and may be repeated according to the size of the ciphertext and the dispersal sterategy. Applying the IDA algorithm on the symmetric secret key ciphertext C_k does not worth the effort since C_k is already small (usually one group element). Therefore, each server in **S** stores a copy of C_k as it is.

6.4 Generating a Receipt

The author u_0 has the right to receive a *receipt* of the deposit as a proof that his package has been deposited correctly on the servers. For this purpose the protocol proceeds as follows:

- Each server $S_i \in$ **S** computes, $H(C_M, C_k, id_{u_0}, id_f)$.
- The set **S** runs a *verifiable threshold digital signature protocol*, at the end, each server S_i holds a partial signature σ_{S_i} of $\sigma_S = E_{sk_s}(H(C_M, C_k, id_{u_0}, id_f))$.
- Each $S_i \in$ **S** sends σ_{S_i} to the authority S_0.
- S_0 collects the received partial signatures $(\sigma_{S_1}, \dots, \sigma_{S_n})$ and reconstructs the signature σ_S from

any valid $t + 1$ partial signatures.

- S_0 sends σ_S to the author u_0 signed using its personal private key sk_{S_0}.
- Provided by **S**'s public key pk_s and S_0's public key, u_0 is able to verify the received signature an decide on its validity.

Once the receipt is generated, the original version C_M of the encrypted file is erased from all servers leaving only the dispersed version.

6.5 Package Request and Retrieval by the User

A user *u* of identity id_u which is supposed to be authorized for the file/package *M* of identity id_f wants to retrieve *M* from the servers. The subprotocol for requesting and retrieving *M* is described next.

Package request: The user *u* requestes the package *M* as follows:

- Picks a blinding integer *r* and using **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 **S**.

Next, the set of servers **S** and the authority S_0 reconstruct C_M and decrypt for the blinded symmetric secret key *rk* as follows:

Retrieval of C_M: To retrieve C_M the protocol proceeds as follows:

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

Retrieval of the blinded symmetric secret key: The retrieval of the blinded symmetric secret key is as follows:

- Each server $S_i \in$ **S** computes the ciphertext, $C_{rk} = C_r C_k$.
- The set **S** runs a verifiable distributed threshold decryption protocol to decrypt C_{rk}. At the end, each server $S_i \in$ **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 own personal private key.
- The signed partial decryptions are sent to S_0 who combines them to reconstruct the blinded key *rk*.

Finally, S_0 sends C_M and *rk* to the user *u* signed with S_0's private key. Then, *u* verifies the signature, extracts *k* from *rk* (since he knows *r*) and decrypts for *M*.

7. Security Analysis and Discussion

7.1 Integrity of the Stored and Retrieved Package

As a result of the IDA dispersal, each server $S_i \in \mathbf{S}$, stores the tuple, $\langle f(i), H'(f(1)), \ldots, H'(f(n))\rangle$. During retrieval, each S_i sends this tuple to the authority S_0. S_0 collects the tuples from all servers, hashes each share $f(i)$ as $H(f(i))$ and for every i she compares $H(f(i))$ to the corresponding $H'(f(i))$ received from each server and accepts $f(i)$ only if $H(f(i))= H'(f(i))$ for more than half of the servers, else, S_0 rejects $f(i)$. For $n = 2t + 1$, it follows that, the authority will always manage to recover C_M given that more than half of the servers (i.e. more than $t = \frac{n-1}{2}$ servers) remain uncorrupted.

7.2 Privacy against the Authority

During file upload, the authority S_0 receives the ciphertext $C_M = E_k(M)$ and an encryption of k as $C_k = E_{pk_s}(k)$ under $\mathbf{S}$'s public key pk_s. The corresponding decryption key sk_s is unknown to S_0, yet, it is shared among $\mathbf{S}$ 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 $C_{rk} = C_r C_k = E_{pk_s}(rk)$ as *rk*. Since S_0 does not know neither *r* nor *k*, S_0 gains no information about *k* and hence, the privacy of *M* holds against S_0. 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.*

7.3 Privacy against the Package Servers

The information collected by the set $\mathbf{S}$ during the execution of our SDRMP are C_M, C_k and C_r. From the threshold security of threshold decryptions, no coalition of t or less servers are able to decrypt neither C_r nor C_k. The decryption is performed only on the product $C_{rk} = C_r C_k = E_{pk_s}(rk)$ and consequently, only *rk* is known to the servers with no information revealed about neither *k* nor *r*. Hence *M* is kept private from all servers in the set $\mathbf{S}$. We then state the following lemma.

Lemma 2: *Assuming that the threshold public key cryptosystem is a t-resilient homomorphic encryption, no coalition of at most t servers in $\mathbf{S}$ gains any information about M in the cryptographic sense.*

7.4 Robustness against 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 or employing DOFS. 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 and the distributed threshold digital signature scheme are verifiable and t-resilient, then at most t malicious servers will always be tolerated.*

7.5 The Authority as a Single Point of Failure and Replication

Another issue to be discussed is that the authority is a single point of failure. Notice that there is no private information delivered to the authority in the clear, so we are not worried about privacy here. However, an adversary may halt or corrupt the authority preventing the execution of the SDRMP. Since no private parameters are known to the authority, the problem of failure could be solved through trivial replication. The author as well as the authorized user is free to contact any of these authorities that are up-and-running as long as they know her public key.

7.6 Proactive Security

Long-lived shared secrets are vulnerable to mobile adversary, an adversary that is able to jump among the servers attacking as much servers as she can. She has the whole life-time of the secret to attack $t+1$ or more players to possess the secret key, sk_s. Also, she can destroy $n-t$ or more servers to destroy the secret key, since in this case, at most *t* good servers remain, which are not enough to perform the threshold signature or decryption. In order to attain long-lived secrecy of the secret key, the shares held by the servers $(sk_{S_1}^{(\tau)}, \dots, sk_{S_n}^{(\tau)})$ must be periodically renewed to new shares $(sk_{S_1}^{(\tau+1)}, \dots, sk_{S_n}^{(\tau+1)})$ so that the information gained by the adversary in a time period say τ is nonsense in time period $\tau+1$. The renewed shares must sum up to the same secret key sk_s. A good implementation of proactive secret sharing can be found in [26] which employs the (+,+)-homomorphic property of Shamir's secret sharing scheme [12] and uses well know techniques such as, joint verifiable zero secret sharing (JVZSS) and Joint verifiable random secret sharing (JVRSS).

7.7 Anonymity Service

In some competitive business applications, a user may want to download the material in an anonymous way to protect his activity from being observed by other users, we did not consider

anonymity service in our protocol, yet, computationally secure anonymity could be realized by allowing all users in the network to establish a group signature protocol among themselves, either, using a group manager or in a democratic manner (i.e. without a manager) [21, 22]. Alternatively, for unconditionally secure anonymity, one may apply ring signature techniques [23] for the users in the enterprise. These notions immediately realize the anonymity service for our protocol.

8. Complexity and Efficiency Evaluation

In this section we provide a concrete analysis of the complexity of our protocol to help other researchers to improve over this protocol. The communication and computation complexity of the set of servers **S** are inherit from the complexity of the verifiable threshold decryption/signature protocols. Considering storage efficiency; for the optimal settings of $n = 2t + 1$, Figure 2 shows the storage reduction for a package of size $|M| = 1$GB. It is obvious that as the threshold value t representing the security level increases, the IDA efficiently reduces the total storage requirements on the n servers. From the graph, notice that, $\lim_{t\to\infty}\left\{\frac{2t+1}{t+1}|M|\right\}$ is $2|M|$ and hence, the storage requirements is asymptotic to $2|M|$. It is nice to notice that by employing the IDA, as the security level t increases, the storage requirements remain almost unchanged. What mainly concerns us next, is to analyze the burden on the author u_0, the user u and the authority S_0 .

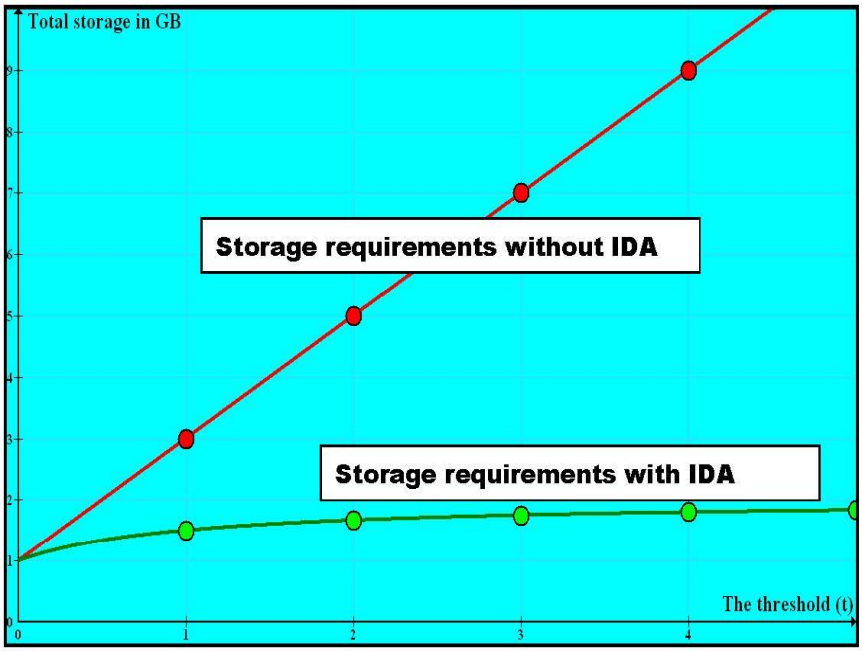

Fig. 2: Storage improvement for a 1GB sized package as a function of the threshold t

8.1 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, $T_{u_0} = \langle C_M, C_k, id_{u_0}, id_f, \sigma_{u_0}\rangle$, the size of id_{u_0} and id_f

are short strings while σ_{u_0} and C_k are only two 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 for any efficient symmetric cryptosystem. The authority u_0 also performs two group computations as: a public key encryption on k as $C_k = E_{pk_s}(k)$ and his own signature as σ_{u_0} which are two group operations on one block of data: the key k and the hash $H(C_M, C_k, id_{u_0}, id_f)$. We conclude that the complexity of the author u_0 is $|M| + O(1)$.

8.2 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 and id_f to all servers in $\mathbf{S}$ with her personal signature on them which is in the order of $|M|$. During file request, 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 shares $f(1), \dots, f(n)$ totaling a size of, $n|M|/(t+1)$ and almost n^2 short hash strings of these shares. In the retrieval of the symmetric secret key k, S_0 receives n signed partial decryptions, $\varepsilon_{S_1}, \dots, \varepsilon_{S_n}$, 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 generation of the receipt, S_0 reconstructs the threshold signature by $\mathbf{S}$ which requires $O(n)$ group operations. During the retrieval of package ciphertext C_M, S_0 performs n hash operations and on polynomial reconstruction. During the retrieval of the secret key, S_0 performs threshold decryption reconstruction in the order of $O(n)$ modulo operations and one personal signature.

8.3 Complexity of the User

User u burden is very light, during file request, performs 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 $|M| + O(1)$.

Unlike the protocols in [1, 9] that requires the users to interact with many authorities, our protocol allows the users to interact with one authority. This authority has no information about the contents of the package uploaded/downloaded.

9. Conclusions

In this paper, we have inspected several recent protocols that provides cryptographic solutions to E-DRM. We discussed several security weaknesses and efficiency drawbacks, then we introduced a protocol for efficient and secure enterprise digital right management (SDRMP) that is robust

against malicious corruptive adversaries. The protocol satisfies the basic requirements for E-DRM, provides efficient storage of large digital packages that is asymptotic in the number of package servers and overcomes the security weaknesses in previous protocols. The protocol ensures privacy against the authorization authority, the unauthorized users and malicious servers. The protocol provides very low communications and computations complexities for the author, the authorization authority and the users.

References

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

[2] Yu, Yang, and Tzi-cker Chiueh, "Enterprise digital rights management: Solutions against information theft by insiders", Research Proficiency Examination (RPE) report TR-169, Department of Computer Science, Stony Brook University, 2004.

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

[4] A. Arnab and A. Hutchison, "Digital Rights Management - An overview of Current Challenges and Solutions", in Proceedings of Information Security South Africa (ISSA), 2004.

[5] A. Arnab and A. Hutchison, "Requirement Analysis of Enterprise DRM systems", in Proceedings of Information Security South Africa (ISSA), Sandton, Johannesburg, South Africa, 2005.

[6] F. Bartolini, V. Cappellini, A. Piva, A. Fringuelli, and M. Barni, "Electronic Copyright Management Systems: Requirements, Players and Technologies", international Workshop on Database and Expert Systems Applications DEXA99, Florence, Italy, pp.896–898, 1999.

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

[8] J. Park, R. Sandhu, and J. Schifalacqua, "Security architectures for controlled digital information dissemination", in Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC '00), New Orleans, LA, USA, pp.224–233, 2000

[9] Chang Chin-Chen, and Jen-Ho Yang, "A Group-oriented Digital Right Management Scheme with Reliable and Flexible Access Policies", International Journal of Network Security, Vol.15, No.6, PP.471–477, Nov. 2013

[10] 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.

[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 (JACM), Vol 36, Issue 2, pp. 335–348, 1989.

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

[15] R. Gennaro, S. Halevi, H. Krawczyk, T. Rabin, “Threshold RSA for Dynamic and Ad-Hoc Groups”, in proceedings of EUROCRYPT’08, pp. 88–107, 2008.

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

[17] Maged H.Ibrahim, I. A. Ali, I. I. Ibrahim, and A. H. El-Sawy, “Fast fully distributed and threshold RSA function sharing”, in proceedings of the Information Systems: New Generations (ISNG’04), pp. 13–18, 2004.

[18] Maged H. Ibrahim, I. I. Ibrahim, and A. H. El-Sawy, “Fast three-party shared generation of RSA keys without distributed primality tests”, in proceedings of the Information Systems: New Generations (ISNG’04), pp. 7–12, 2004.

[19] G. Tardos, “Optimal probabilistic fingerprint codes”, Journal of the ACM vol 55, no. 2, 2008.

[20] D. Boesten and B. kori, “Asymptotic fingerprinting capacity in the Combined Digit Model”, in Information Hiding, Springer Berlin Heidelberg, pp. 255–268, 2013.

[21] M. H. Ibrahim, Resisting Traitors in Linkable Democratic Group Signatures, International journal of Network Security (IJNS), vol. 9, no. 1, pp. 51–60, 2009

[22] M. Manulis, “Democratic group signatures on an example of joint ventures”, in proceedings of the ACM Symposium on Information Computer and Communications Security, (ASIACCS’06), pp. 365–365, Taiwan, China, 2006.

[23] A. Shamir, R. L. Rivest, and Y. Tauman, “How to leak a secret”, Asiacrypt’01, Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security, LNCS 2248, pp. 552–565, Springer-Verlag, 2001.

[24] T. Pedersen, “Non-interactive and information theoretic secure verifiable secret sharing", Advances in Cryptology - Crypto '91, LNCS 576, pp. 129-140, Springer-Verlag, 1992.

[25] P. Feldman, “A practical scheme for non-interactive verifiable secret sharing", in proceeding 28th Annual Symposium on the Foundations of Computer Science, pp. 427-437, 1987.

[26] A. Herzberg, S. Jarekci, H. Krawczyk, and M. Yung, Proactive Secret Sharing or: How to Cope with Perpetual Leakage, In Don Coppersmith, editor, Proc. of CRYPTO 1995, the 15th Ann. Intl. Cryptology Conf., volume 963 of Lecture Notes in Computer Science, pp. 339-352. Intl. Assoc. for Cryptologic Research, Springer-Verlag, August 1995.