*International Journal of Computer Applications (0975 - 8887)*
*Volume 126 - No.3, September 2015*

# Secure Anonymously Authenticated and Traceable Enterprise DRM System

Maged Hamada Ibrahim
Department of Electronics, Communications and Computers Engineering,
Faculty of Engineering, Helwan University
1, Sherif St., Helwan, Cairo, P.O. Box 11792, Egypt

## ABSTRACT

Anonymity is one of the important services that must be available to users in the digital world as long as they behave honestly. Users' communication must be kept authenticated and anonymous unless malicious behaviors are detected. In this case the accused user's clear identity must be traced and revealed by the system to solve accusations. Enterprise Digital Rights Management (E-DRM) protects business digital applications by allowing an author in an organization to securely upload his confidential package/file(s) 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. In this paper, we extend our previously proposed E-DRM protocols and propose an E-DRM protocol that allows authorized users to upload, store and download packages in an efficiently secure, anonymous and authenticated way. On the other hand, in case of an accusation or a dispute, our system is able to trace the user to his clear identity to solve accusations.

### General Terms

Cryptography, Network Security

### Keywords

Enterprise security, Digital rights management, Anonymity, Traceability, Threshold cryptography, Authentication, Group signatures

## 1. INTRODUCTION

Digital Rights Management systems (DRM) in general deals with the main issues [26, 45, 3, 2, 4, 7]: 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. DRM mainly provides two types of systems: (i) Systems for distributing contents to consumers in a controlled way against piracy; (ii) Systems for managing access to sensitive document contents 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.

Efforts have focused on rights-centric security policies and enhanced mechanisms. Consumer-centred 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. [26, 25, 14, 45, 34, 35]). 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 [43, 12], whereas the embedded 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 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 challenging issue is that such protocols must satisfy a number of security services that could be complex by their nature. In secure E-DRM protocols, anonymity and traceability are two important services, yet, achieving a satisfactory security level for both of them –with acceptable complexity– is not an easy task due to the contradicting requirements: anonymous transmission must not be traceable by any individual while if a transmission is traceable, then anonymity is threatened.

When anonymous and authenticated transmission is considered, *Group signatures* (GS) come to play [28, 13, 9, 8]. This cryptographic tool originally introduced in [16] allows members belonging to a group to sign messages on behalf of the group such that, the signature verifier (whether a group member or a non-member)

is able to check that the signature is a valid group signature but cannot trace the identity of the signer. In case of a dispute, the trusted authority (group manager) can trace the identity of the signer.

*Ring signatures* introduced in [39] and further studied in [30] and [11] do not require any group manager to form a group. For signature generation, every user builds a set of public keys that includes his public key and the public keys of other users. A generated signature does not reveal the public key of the signer, but a set of public keys of all possible signers. Therefore, ring signatures cannot be used for a direct communication between a verifier and a signer. Additionally, ring signatures provide unconditional anonymity, i.e., no party can reveal the signer's identity. Although ring signatures have many cryptographic applications, they are not suitable for our system since traceability is impossible.

*Democratic group signatures* (DGS) [32] eliminates the role of a group manager by (i) allowing the group members themselves to initialize and setup the group, (ii) controlling it over dynamic changes in a collective manner and (iii) distributing traceability rights to each group individual. In this case, every group member has the individual right to trace and disclose the identity of the signer and hence, anonymity is provided against non-members. The model in [32] requires unlinkability of signatures, i.e., the signature verifier cannot distinguish signatures issued by the same group member without this member being traced and disclosed.

DGS schemes differ from *threshold signature* schemes (e.g. [18, 19, 23, 21, 20, 22]) in the sense that, in DGS each group member is granted the right to generate a signature on a given message individually; a non-member verifier recognizes the signature as anonymously generated by the group. On the other hand, in threshold signatures, the signature on a given message is generated by the majority of the group (exceeding a certain threshold), yet, no coalition of minority (less than or equals the threshold) can generate the signature.

*Linkable democratic group signatures* (LDGS) [33] realize linkability of signatures issued by the group members in a way that preserves the anonymity of the signer. More precisely, a non-member verifier is able to distinguish signatures issued by the same signer for future reference without being able to trace the identity of this signer. To achieve this property, LDGS actually employs the idea of pseudonym systems introduced in [31]. In this scenario, each group member (in addition to his unique identity) will be assigned a unique pseudonym. Given a certain group member, all signed messages generated by this particular member will carry his unique pseudonym. A non-member verifier is able to link signed messages of the same signer via his pseudonym, yet, the verifier gains no information about the signer's identity from this pseudonym. On the other hand, each group member knows the secret tracing trapdoor parameter, by which, he is able to extract the identity of the signer from the signer's unique pseudonym.

LDGS consists mainly of four protocols/algorithms: protocol `setup`, algorithm `sign`, algorithm `verify` and algorithm `trace`. The `setup` protocol takes as input a security parameter $l$, and a number of members $n$. For each member $M_i$, the public output is an identity $id_i$ from the set of identities $ID$ and a pseudonym $ps_i$ from the set of pseudonyms $PS$ while the private output is the secret signing key $sk_i$ from the set of secret keys $SK$ and the secret tracing trapdoor parameter $k$ known to each member. The protocol requires the existence of a PKI that allows the group members to authenticate their messages during setup via their certified public keys. Yet, there is no third party actively involved in the protocol. The `sign` algorithm takes as an input a secret key $sk_i$ and a message $m$ and outputs a signature $\sigma$ on $m$. The `verify` algorithm takes as an input a signature $\sigma$, a message $m$ and the set of pseudonyms $PS$ and outputs a pseudonym $ps_i$ if it accepts the signature or $\perp$ if the signature is rejected. Algorithm `trace` takes as input a signature $\sigma$, a message $m$, the secret tracing trapdoor parameter $k$, and the set of pseudonyms $PS$ and outputs either an identity $id_i$ or $\perp$ in case of failure.

During the setup algorithm, in the computation of the secret tracing trapdoor parameter $k$, the LDGS employs an authenticated DH-based group key agreement protocol as so-called the contributory group key agreement (CGKA) protocols e.g. [29, 1]. In a CGKA protocol, each member $M_i$ of the group contributes his public key $y_i$ for the interactive computation of the secret common key $k$.

In an improvement to the LDGS discussed above, in [24], we performed modifications to the `setup` and `trace` protocols to allow the group to withstand possible traitors members. Our idea is to remove the CGKA protocol from the `setup` protocol and employ efficient threshold cryptographic tools. In this case, the secret tracing trapdoor parameter $k$ will be shared among the members on a threshold basis. The members are able to jointly share a secret parameter which allows the tracing to be performed by the majority of the members. As a result, minority traitors will not be able to perform the tracing and are prevented from disclosing the identity of any signer without a legal reason. Also, the system must be robust against possible malicious behavior of the traitors during the `setup` and `trace` protocols.

## 2. PREVIOUS WORK

The requirements of a well-designed DRM system have been studied in several subsequent publications: Arnab et al. [3, 2, 4], Bartolini et al. [7], Mulligan et al. [34], and Park et al. [35]. Our recent contribution of [26] is to device a stronger digital rights management protocol for enterprise applications that overcomes the security problems and efficiency drawbacks in previous protocols such as [14, 45, 7, 35] and others. In [26], 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. Some storage efficiency improvements to the protocol of [26] was introduced in [42]. In [25], We refined and extended the E-DRM protocol of [26] for better efficiency in retrieving files of very large sizes. The 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. However, none of the above proposals considered the availability of the anonymity service to the users.

## 3. MOTIVATIONS AND CONTRIBUTIONS

### 3.1 Motivations

Anonymity service is one of the important services that must be provided to the users by an E-DRM system. In an E-DRM system, the users may wish to keep their behavior/activities anonymous to others as long as they behave legally. The attempt of uploading packages by the author and downloading these packages by the authorized users must be kept anonymous to everyone unless there are accusations (e.g. a dispute). The previously proposed E-DRM

protocols such as [26, 25, 14, 45, 7, 35] do not provide such service since the identities of the users as well as the identities of the uploaded/downloaded files are known to the authority and all other entities within the organization.

### 3.2 Our contribution

In this paper we study previously proposed E-DRM systems and extend our recent protocols of [26, 25] for robust and secure E-DRM to allow transfer of packages among legitimate users in a completely anonymous and authenticated manner where the users identities are kept anonymous yet traceable in case of raised accusations. We employ ideas from [33, 32, 24], threshold cryptographic tools from [41, 17, 36, 19, 27, 10], information dispersal tools from [37, 38] and proofs of knowledge tools from [15, 40, 5, 24] to allow the users to securely and anonymously sign and upload/download the packages they are authorized for. On the other hand, with the help of a distributed tracing trapdoor parameter, the package servers are able to (jointly) trace and identify any user in case of raised accusations.

## 4. OUR TOOLS

In this section we describe the basic tools that will be used to build our E-DRM protocol. These tools are partitioned into three categories: threshold cryptography tools, proofs of knowledge tools and information dispersal tools. The reader must be familiar with these tools in order to follow the description of our protocol.

### 4.1 Threshold Cryptography Tools

**Secret sharing over a prime field.** Let $s \in Z_q$ be a secret held by the dealer where the group $Z_q$ is a prime field. In order to share this secret among a set $P=\{P_1, ..., P_n\}$ of $n>t$ players [19], the dealer defines a polynomial $f(x) = \sum_{j=0}^{t} a_j x^j \bmod q$, 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, $s=f(0) = \Sigma_{i\in B} \lambda_i f(i) \bmod q$ where $B \subset P$, $|B| = t+1$ and $\lambda_i$ is Lagrange coefficient for player $P_i$.

**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 [20] and the unconditionally secure VSS due to Pedersen [21]. To achieve best security [22], both of them will be used in our protocol. In *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 $f(x) = \sum_{j=0}^{t} a_j x^j \bmod q$, the dealer also broadcasts the $t+1$ commitments $c_j=g^{a_j} \bmod p \ \forall j= 0, ...,t$. These commitments allow each player $P_i$ to verify the consistency of his share $f(i)$ by checking that, $g^{f(i)} = \prod_{j=0}^{t} c_j^{i^j} \bmod p$. 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 $g^k \bmod p$, Feldman's VSS alone is not secure due to the attacks described in [19]. In *Pedersen's VSS*, 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 $r(x) = \sum_{j=0}^{t} b_j x^j \bmod q$. He secretly delivers $(f(i),r(i))$ to player $P_i \ \forall i= 1, ...,n$. The dealer also publishes the commitments $c_j=g^{a_j} h^{b_j} \bmod p \ \forall j= 0, ...,t$. Each player $P_i$ verifies the consistency of his share $f(i)$ by checking that, $g^{f(i)} h^{r(i)} = \prod_{j=0}^{t} c_j^{i^j} \bmod p$. 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.

**Joint secret sharing.** Joint secret sharing allows the players to jointly share some secret among themselves without the help of the dealer. *Joint random secret sharing (JRSS)* [27] allows a set of $n$ players to jointly share a random secret without the help of the dealer. Each player $P_i \in P$ 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 $f_i(x) = k_i + \sum_{j=1}^{t} a_j x^j \bmod q$. Each player $P_i \in P$ 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 $k= \sum_{i=1}^{n} k_i \bmod q$. *Joint random verifiable secret sharing (JRVSS)* is 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 $P_i \in P$ 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 $f(0) = \sum_{i=1}^{n} k_i \bmod q$. *Joint zero secret sharing (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)* is as in the JRVSS, to withstand 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 $P_i \in P$ 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 $g^0 = 1$.

**The distributed multiplication protocol.** Given two secrets $a$ and $b$ shared over $t$-degree polynomials $A(x)$ and $B(x)$ respectively, the `multiplication` protocol [10] computes $\xi = ab \bmod q$ in a robust way with no information revealed about $a$ or $b$. Each player $P_i$ locally computes $C(i) = A(i)B(i) \bmod q$. In this case, each $C(i)$ is a share on a $2t$-degree polynomial $C(x) = A(x)B(x) \ mod q$ with $C(0) = \xi$. However, publishing and interpolating the shares $C(1), ..., C(n)$ reveals information about $A(x)$ and $B(x)$ [10], consequently, randomizing the shares of $C(x)$ is necessary. To randomize the shares of $C(x)$ without changing $C(0)$ the players run a `JZVSS` to share a zero over a $2t$-degree polynomial $R(x)$ with $R(0) = 0$. Finally, each player $P(i)$ computes and publishes $D(i) = C(i) + R(i)$. The result $\xi$ could be reached by interpolating the $2t$-degree polynomial $D(x)$ with the help of the Berlekamp-Welch decoding scheme [44] to filter out corrupted shares. Since we are interpolating a polynomial of degree $deg = 2t$ and we have a maximum of $t$ malicious players (i.e. at most $t$ possible $faults$), using the Berlekamp-Welch bound, the number of shares needed in order to correctly interpolate the polynomial is at least $deg + 2faults + 1 = 4t + 1$. Hence, we need $n > 4t$.

**The distributed reciprocal protocol.** In the tracing process of our protocol we are faced with the following problem. Given a secret $k$ which is shared among the players, generate a sharing of the reciprocal of $k$ modulo $q$ with no information revealed about $k$. Each player $P_i$ holds a share $f(i)$ which is a point on a $t$-degree polynomial $f(x)$ with $f(0) = k$. To compute shares of $k^{-1}$, we need $n > 4t$, the $n$ players run the reciprocal protocol [19, 10, 6] as follows:

—The players run the `JRVSS`, at the end each player holds a share $v(i)$ of a random secret $v$ over a polynomial of degree $t$.

—The players run the `multiplication` protocol and reconstruct the value $\xi = kv \bmod q$.

—Finally each player $P_i$ computes his share of the reciprocal as $\xi^{-1} v(i) \bmod q$, which is a share over a $t$-degree polynomial with its free term equals $k^{-1} \bmod q$.

## 4.2 Proofs of Knowledge Tools

**Proof of equality of two discrete logarithms.** We review the protocol of [15, 40] and also in [24] 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 ($P$) proves to a verifier ($V$) that he knows $x \in Z_q^*$ such that $G_1 = \alpha^x \bmod p$ and $G_2 = \beta^x \bmod p$. The protocol is as follows:

— $P \rightarrow V$: Choose $r \in_R Z_q^*$ and send ($A = \alpha^r \ mod p$, $B = \beta^r \bmod p$).

— $V \rightarrow P$: Choose $c \in_R Z_q^*$ and send $c$.

— $P \rightarrow V$: Send $y = r + cx \bmod q$.

— $V$: Check that $\alpha^y = AG_1^c \bmod p$ and $\beta^y = BG_2^c \bmod p$.

The above protocol can be made non-interactive (we denote it by $\Pi_{LogEq} \leftarrow P_{LogEq}(\alpha, \beta, G_1, G_2, x)$ ) using a sufficiently strong hash function $H(.)$ [5] and setting $c = H(A, B)$. The protocol $\Pi_{LogEq}$ becomes as follows:

— $P \rightarrow V$: Choose $r \in_R Z_q^*$ and send ($A = \alpha^r \bmod p$, $B = \beta^r \bmod p$, $c = H(A, B)$ and $y = r + cx \bmod q$).

— $V$: Check that $\alpha^y = AG_1^c \bmod p$ and $\beta^y = BG_2^c \bmod p$.

**Proof of existence of a discrete log equality.** Let $y_i = \alpha^{x_i} \bmod p$ for $i = 1, .., n$ and let $z = \beta^{x_i} \bmod p$ for some $i \in \{1, ..., n\}$. A prover $P$ demonstrates to a verifier $V$ that he knows one of the logarithms of $y_i$ ($i \in \{1, ..., n\}$) to the base $\alpha$ and that $\log_\alpha y_i = \log_\beta z \bmod q$ without revealing which $i$. Let wlog the relation holds for $i = 1$ (i.e. $x_1 = \log_\alpha y_1 = \log_\beta z \bmod q$). The protocol is as follows [5, 24]:

—$P \rightarrow V$: Choose $k_i \in_R Z_q^*$ for $i = 1, ..., n$, $c_j \in_R Z_q^*$ for $j = 2, ..., n$ and compute:

—$r_1 = \alpha^{k_1} \bmod p$, $r_i = \alpha^{k_i} y_i^{-c_i} \bmod p$ for $i = 2, ..., n$.

—$t_1 = \beta^{k_1} \bmod p$, $t_i = \beta^{k_i} z^{-c_i} \bmod p$ for $i = 2, ..., n$.

Send the values $(r_1, ..., r_n, t_1, ..., t_n)$.

—$V \rightarrow P$: Choose and send $c \in_R Z_q^*$.

—$P \rightarrow V$: Calculate $c_1 = c - \sum_{i=2}^{n} \bmod q$, $s = x_1 c_1 + k_1 \bmod q$ and set $s_i = k_i$ for $i = 2, ..., n$. Send $(c_1, ..., c_n, s_1, ..., s_n)$.

—$V$: Check that $c = \sum_{i=1}^{n} c_i$ and that $\forall i = 1, ..., n$, $\alpha^{s_i} = y_i^{c_i} r_i \bmod p$ and $\beta^{s_i} = z^{c_i} t_i \bmod p$.

The above interactive proof can be transformed into a non-interactive proof that we will denote it by,
$\Pi_{\exists LogEq} \leftarrow P_{\exists LogEq}(\alpha, \beta, y_1, ..., y_n, z)$ using a strong hash function $H(.)$ [5]. This can be done by setting,

$$c = \mathcal{H}(y_1, ..., y_n, \alpha, z, \beta, \alpha^{s_1} y_1^{-c_1}, ..., \alpha^{s_n} y_n^{-c_n}, \beta^{s_1} z^{-c_1}, ..., \beta^{s_n} z^{-c_n}).$$

## 4.3 Information Dispersal Tools

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* [41] 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) [37] will help to provide efficient storage of information. Given a threshold $t$ and $n > t$ servers, $S_1, \ldots, S_n$, 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 $(m_0, \ldots, m_t)$, 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*.

The information dispersal algorithm introduced in [38] combines the "all-or-nothing-transform" (AONT) and the IDA of [37] to provide slightly better storage efficiency over that of [37] and also

incorporates the Reed-Solomon (RS) codes for error control. The AONT-RS in [38] can straight forward replace the IDA of [37] used in our protocol, however, for a better understanding of our protocol, we apply only the original IDA introduced in [37].

## 5. ASSUMPTIONS AND MODEL

We follow the network topology and communication model of [26, 25]: 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 $\mathcal{U}=\{u_1, ..., u_m\}$ and a special user $u \in \mathcal{U}$ which is the author of the confidential file $M$. Any user can be the author at any time. User $u$ 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) to realize private and authenticated channels for all entities. 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 [18, 19]. Our protocol assumes the number of servers in $\mathcal{S}$ to be $n= 4t+1$ with at most $t$ of them could behave maliciously. This lower bound on $n$ could be reduced to $n> 3t$ and further to $n> 2t$ but with a dramatic increase in complexities. 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 [43, 12]. Although such techniques can be easily integrated with our protocol, the development of such techniques is 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. For clarity and simplicity, in the description of our protocol, we assume a single group of users and authors. However, the protocol could be easily extended to allow multiple groups of users each with their own authorization attributes.

## 6. OUR PROTOCOL CONCRETE DESCRIPTION

We are ready to describe the details of our E-DRM. We assume that there exists $n> 4t$ servers with at most $t \geq 1$ possible malicious servers. Also we assume that each server and user has his own private and certified public keys to realize authenticated and confidential communications. The initial public parameters of the system are: two large primes $p, q$ such that $q|p-1$ and two generators $g, h \in Z_p^*$ such that $\log_g h$ is unknown and assumed hard to compute.

### 6.1 Identities setup by users

The users setup their identities, each user $u_i \in \mathcal{U}$ performs as follows:

—Picks a private key $x_i \in_R Z_q$.

—Computes and publishes the corresponding authenticated identity (or public key) $y_i = g^{x_i} \bmod p$ and a proof of knowledge of $x_i$.

At the end, we have the set $\mathcal{ID}=\{y_1, ..., y_m\}$ of published clear identities. Notice that authentication of user $u_i$ in publishing his identity $y_i$ is achieved using his private key $sk_{u_i}$ of a conventional digital signature scheme.

### 6.2 Distributed key generation by the servers

The set $\mathcal{S}=\{S_1, \ldots, S_n\}$ 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 $\kappa$ and publicize the corresponding public key $pk_S = y = g^x \bmod p$ as follows [18, 19]:

—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 $g^{a_j^{(i)}} \bmod p \ \forall j = 1, ..., t$.

—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 $g^{a_j^{(i)}} \bmod p \ \forall j = 1, ..., t$ in the clear.

—For the remaining $n$ good servers, each server $S_i$ holds a share $X(i) = \sum_{j=1}^{n} X_j(i) \bmod q$ of the private key $x = \sum_{i=1}^{n} x_i \bmod q$ on a $t$-degree polynomial $X(z) = x + \sum_{i=1}^{t} a_i z^i \bmod q$. Each server also computes the commitments $g^{a_i} = \Pi_{j=1}^{n} g^{a_i^{(j)}} \quad \forall i = 0, ..., t$ 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 $g^{X(i)} = \Pi_{j=0}^{t} g^{a_j i^j} \bmod p$.

### 6.3 Tracing parameter setup by the servers

— The servers in $\mathcal{S}$ run a `JRVSS` with Pedersen's VSS as the VSS in place. At the end, each server $S_i \in \mathcal{S}$ holds a share $K(i)$ of a random secret $k \in_R Z_q^*$ over a $t$-degree polynomial $K(x)$ with $K(0) = k$.

—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 $K_i(x) = k_i + \sum_{j=1}^{t} a_j$ is the polynomial of server $S_i$, $S_i$ broadcasts $g^{k_i}$ and $g^{a_j} \bmod p \ \forall j = 1, ..., t$.

—For any server $S_i$ who receives at least one valid complaint in the previous step, the other Servers join to reconstruct his polynomial $K_i(x)$ and values $g^{k_i}$ and $g^{a_j} \bmod p \ \forall j = 1, ..., t$ in the clear.

—Finally, the remaining good servers join to safely compute $g^k = \prod_{i=1}^{n} g^{k_i}$.

At this point, the servers in $\mathcal{S}$ share the tracing trapdoor parameter $k$ over a $t$-degree polynomial and have jointly computed the blinded tracing trapdoor, $bk{=}g^k \mathrm{mod} p$. As a preparation for the tracing protocol, the servers need to compute shares of $k^{-1} \mathrm{mod} q$, so they proceed:

—The servers run the `reciprocal` protocol, at the end, each server holds a share $D(i)$ on a $t$-degree polynomial, $D(x)$ with its free term $D(0){=}k^{-1} \mathrm{mod} q$.

—Each server broadcasts Feldman's VSS commitments (i.e. to the base $g$) of all his chosen random polynomials during the reciprocal protocol. These commitments allow the servers to validate the quantities, $G_i{=}g^{D(i)} \mathrm{mod} p \; \forall i$.

### 6.4 Generating pseudonyms by the users

Next, the users compute their pseudonyms, each user $u_i$ performs as follows:

(1) Computes $y_i^*{=}\,(bk)^{x_i} \mathrm{mod} p$ .

(2) Parses his pseudonym as $ps_i{=}\,(bk, y_i^*)$.

This finalizes the setup protocol, each user $u_i$ sets his secret information $sk_i$ as $sk_i{=}\,(x_i, D(i), ps_i)$. Notice that, $ps_i$ will be used by user $u_i$ for transmission instead of his clear identity, hence the mapping of $ps_i$ to the corresponding clear identity must be kept secret. The public parameters are the set of identities, $\mathcal{ID}{=}\{y_1, ..., y_m\}$, the set of commitments, $G{=}\{G_1, ..., G_n\}$ where $G_i{=}g^{D(i)} \mathrm{mod} p$ and the blinded tracing trapdoor parameter, $bk$.

### 6.5 Package anonymous uploading by the author

The author $u_\mathrm{i}$ possessing a file/package $M$, provided with 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 $C_K{=}E_{pk_S}(K)$.

The author proceeds to anonymously sign as follows:

—Picks an integer $r{\in_R}Z_q^*$, computes the hash $H{=}H(C_M, C_K, id_f, r)$ and computes $z{=}H^{x_i} \mathrm{mod} p$.

—Generates a NIZK proof $\Pi_{LogEq} \leftarrow P_{LogEq}(H, bk, z, y_i^*, x_i)$, which proves that $\log_H(z) {=} \log_{bk}(y_i^*)$.

—Generates a NIZK proof, $\Pi_{\exists LogEq} \leftarrow P_{\exists LogEq}(g, bk, y_1, ..., y_m, y_i^*)$, which proves that there exists some index $i \in \{1, ..., m\}$ such that $\log_g y_i {=} \log_{bk} y_i^*$.

—Parses $\sigma_{u_i}$ as $(r, z, ps_i, \Pi_{LogEq}, \Pi_{\exists LogEq})$.

The author $u_i$ completes his upload by sending to the authority server $S_0$ the tuple, $T_{u_i}{=}\langle C_M, C_K, id_f, \sigma_{u_i}\rangle$.

### 6.6 Author's signature verification by the authority

On the reception of $T_{u_i}$, the authority $S_0$:

—Parses $\sigma_{u_i}$ as $(r, z, ps_i, \Pi_{LogEq}, \Pi_{\exists LogEq})$ and parses $ps_i$ as $(bk, y_i^*)$.

—Computes $H = \mathcal{H}(C_M, C_K, id_f, r)$.

—Runs the verification algorithm, $V_{LogEq}(H, bk, z, y_i^*, \Pi_{LogEq})$, if not successful then reject $T_{u_i}$ and abort. Else,

—Runs the verification algorithm, $V_{\exists LogEq}(g, bk, y_1, ..., y_m, y_i^*, \Pi_{\exists LogEq})$, if not successful then reject $T_{u_i}$ and abort. Else, accept $T_{u_i}$.

On the acceptance of $T_{u_i}$, $S_0$ sends $C_M$, $C_K$, 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.

### 6.7 Deposit and dispersal of the author's package

Each server in $\mathcal{S}$ runs the IDA algorithm as in [26, 42, 25], 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 $f(x) = \sum_{\ell=0}^{t} C_M^{(\ell)} x^\ell \mathrm{mod} p$.

—Computes $f(j)$ and $\mathcal{H}(f(j)) \; \forall j{=}\,1, ...n$.

—Stores $f(i)$ and $\mathcal{H}(f(j)) \; \forall j{=}\,1, ..., n$. Erases all other $f(j \neq i)$ 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 $C_K$ does not worth the effort since it is already small (usually one group element). Therefore, each server in $\mathcal{S}$ stores a copy of $C_K$ as it is. Finally, each server $S_i \in \mathcal{S}$ ends up storing for each package the tuple, $\langle f(i), \mathcal{H}(f(1)), ..., \mathcal{H}(f(n)), C_K, id_f\rangle$.

### 6.8 Package request and retrieval

**Package request.** The user $u_i$ requests a package/file $id_f$ that he is authorized for as follows:

— Picks a blinding parameter $r$ and using $\mathcal{S}$'s $pk_S$ encrypts $r$ as $C_r{=}E_{pk_S}(r)$.

—Prepares a request for the file $id_f$ he wants to download and proceed to anonymously sign his request,

—Computes the hash $H = \mathcal{H}(req, id_f, C_r)$ and computes $z{=}H^{x_i} \mathrm{mod} p$.

—Generates a NIZK proof, $\Pi_{LogEq} \leftarrow P_{LogEq}(H, bk, z, y_i^*, x_i)$, which proves that $\log_H(z) {=} \log_{bk}(y_i^*)$.

—Generates a NIZK proof, $\Pi_{\exists LogEq} \leftarrow P_{\exists LogEq}(g, bk, y_1, ..., y_m, y_i^*)$, which proves that there exists some index $i \in \{1, ..., m\}$ such that $\log_g y_i {=} \log_{bk} y_i^*$.

—Parses $\sigma_{u_i}$ as $(z, ps_i, \Pi_{LogEq}, \Pi_{\exists LogEq})$.

—Sends to $S_0$ the tuple $T_{u_i}{=}\langle req, id_f, C_r, \sigma_{u_i}\rangle$.

—On the reception of $T_{u_i}$, $S_0$ verifies the signature and ensures that $u_i$ of pseudonym $ps_i$ is authorized for $id_f$.

—The authority $S_0$ forwards a signed version of $id_f$ and $C_r$ to all servers in $\mathcal{S}$.

**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 $\mathbf{C_M}$:**

—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, $C_{rK}=C_rC_K$.

—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 $\varepsilon_{S_i}$ of $E_{sk_s}(C_{rK})$.

—Each $S_i$ signs his partial decryption $\varepsilon_{S_i}$ using his personal private key.

—The signed partial decryptions are sent to $S_0$ to reconstruct the blinded key, $rK$.

Finally, $S_0$ sends $C_M$ and $rK$ to the anonymous user of pseudonym $ps_i$ signed with $S_0$'s private key. User $u_i$ verifies the signature, extracts $K$ from $rK$ (since he knows $r$) and decrypts for $M$.

### 6.9 Tracing an accused user

In case of – for example – a dispute, $S_0$ requests the servers in $\mathcal{S}$ to join to reveal the identity of the signer from the argued pseudonym $ps_j=(bk, y_j^*)$. Each server $S_i$:

—Broadcasts $Y_i=(y_j^*)^{D(i)} \bmod p$, where $D(i)$ is $S_i$'s share of $k^{-1} \bmod q$.

—Broadcasts $\Pi_{LogEq} \leftarrow P_{LogEq}(g, y_j^*, G_i, Y_i, D(i))$ a prove that $\log_g G_i = \log_{y_j^*} Y_i \bmod q$.

From any $t+1$ quantities, $Y_i$'s, that pass the proof $\Pi_{LogEq}$ successfully, each server can perform interpolation in the exponent to compute $(y_j^*)^{1/k}=y_j$. Interpolation in the exponent is as simple as computing: $\prod_{i\in B}(y_j^*)^{D(i)\lambda_i}=(y_j^*)^{\sum_{i\in B} D(i)\lambda_i}=(y_j^*)^{k^{-1}}=y_j$, where $|B|=t+1$ and $\lambda_i$ is Lagrange coefficient of server $S_i$.

## 7. SECURITY ANALYSIS

In the core of our protocol, over that in [26, 25], the generation of the tracing trapdoor parameter (which is a random, uniformly distributed value $k$) and the private decryption key (which is a random, uniformly distributed value $x$) are distributed on a threshold basis and the values $bk=g^k$ and $y=g^x$ are made public. The protocol is called $t$-secure, that is, in the presence of at most $t$ malicious servers:

—All subsets of $t+1$ valid shares reconstruct to the same unique secret parameter $k$ and private key $x$.

—Each server is able to compute the common public values $bk=g^k$ and $y=g^x$.

—The parameters $k$ and $x$ are uniformly distributed in $Z_q$ and hence, $bk$ and $y$ are uniformly distributed in the subgroup generated by $g$.

—No information on $k$ or $x$ can be learned by the coalition of at most $t$ servers except for what is implied by the values $bk=g^k$ and $y=g^x$.

Performing JRVSS with Feldman's VSS alone is insecure, since, traitors can influence the distribution of the result of Feldman's VSS to a non-uniform distribution [19]. More precisely, the attack works as follows: Assume that two traitors – say $S_1$ and $S_2$ – want to bias the distribution towards values $bk$ whose last bit is 0. $S_1$ gives members, $S_3, ..., S_{t+2}$ shares which are inconsistent with his broadcast values, the rest of the members receive consistent shares. Thus, there will be $t$ complaints against $S_1$, yet $t$ complaints are not enough for disqualification. The traitors compute $\alpha=\sum_{i=1}^{n} g^{k_i}$ and $\beta=\sum_{i=2}^{n} g^{k_i}$. If $\alpha$ ends with 0 then $S_1$ will do nothing and continue the protocol as written. If $\alpha$ ends with 1 then force the disqualification of $S_1$, this is achieved by asking $S_2$ to also broadcast a complaint against $S_1$, which brings the number of complaints to $t+1$. This action sets the public value $bk$ to $\beta$ which ends with 0 with probability 1/2. One must notice that synchronous broadcast does not prevent such attack to take place. Hence, the third requirement for correctness and the secrecy requirement dramatically fail. In Pedersen's VSS, the view of the traitors is independent of the value of the secret $k$, and therefore the secrecy of $k$ is unconditional; this eliminates the possibility of the described attack. The security of the JRVSS can be proven via simulation as in [19].

In our protocol, when the verifier receives a signed message from a signer for the first time, the signer must prove that the included pseudonym is valid (i.e. related to an identity in the set $\mathcal{ID}=\{y_1, ..., y_m\}$), hence, the signer $u_i$ includes the proof $P_{\exists LogEq}(g, bk, y_1, ..., y_m, y_i^*)$ which proves to the verifier that there exists some index $i \in \{1, ...,m\}$ such that $\log_g y_i = \log_{bk} y_i^*$ with no information revealed about the exponent $x_i$ or the index $i$.

In the verification algorithm, since $\Pi_{LogEq}$ is ZK, a verifier that receives a signed message with a certain pseudonym $ps_i=(bk, (bk)^{x_i})$ is faced with the computational Diffie-Hellman problem (CDHP) to compute $x_i$ given $bk$ and $(bk)^{x_i}$. Given the set of identities $\mathcal{ID}$, anonymity of the signer is preserved, since $\Pi_{\exists LogEq}$ is ZK. The verifier is faced with the decisional Diffie-Hellman problem (DDHP), that is, given, $g, g^k, g^{kx_i}$, distinguish $g^{x_i}$ from $g^{x_j}$ for a random $x_{j\neq i} \in Z_q^*$.

The tracing protocol is $t$-resilient, that is at most $t$ malicious servers cannot trace the identity of the signer. This statement follows from the properties of threshold structures [19, 10].

Our protocol assumes that the authority $S_0$ 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. [18]). 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. 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 emphasize that $S_0$ is assumed semi-honest 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.

## 8. COMPARISONS AND DISCUSSION

Since we add a new security service to the protocol of [26, 25], it is predictable that the computations and communications complexi-

ties will increase accordingly. In the setup protocol by the servers, a verifiable secret sharing scheme is used to share $k$ and to resist possible malicious behaviors attempted by at most $t$ servers. There is an increase in the computations and communications complexities due to the need to compute and publicize polynomial commitments chosen by each server. As a result of the setup protocol, the group public parameters increases over that in [26, 25] by $n$ group elements (contained in the set of public commitments, $G$). One must notice that the setup protocol is performed only once.

In the anonymous signature algorithm, the length of the signature (as a first signature from this group signer) increases due to the need to include $\Pi_{\exists LogEq}$ in the signature. However, this proof is performed by the signer only once per verifier to prove that the signer's pseudonym is a valid pseudonym related to some anonymous identity $y_i$, once the verifier successfully verifies the correctness of the pseudonym; he simply stores the pseudonym for future messages from this particular signer. Hence, $\Pi_{\exists LogEq}$ is included only in the first signed message to this verifier and removed from the future messages. The increase of complexity in the algorithm 'verify' is also due to the need to perform the verification step, $V_{\exists LogEq}(.)$ for each new group signer.

In the tracing protocol, which is not supposed to be frequently performed unless there is a legal reason (e.g. a dispute) agreed by the majority of the servers, the complexities increase due to the need to perform the proof $\Pi_{LogEq}$ for each broadcasted quantity $y_i$ and the need to perform interpolation in the exponent.

Finally we notify that our proposed protocol could be implemented over other number theoretic settings of groups over finite fields such as Elliptic cure [20].

## 9. CONCLUSIONS

In this paper we proposed an E-DRM protocol that allows authorized users to upload, store and download packages in an efficiently secure, anonymous and authenticated way. On the other hand, in case of an accusation or a dispute, our protocol is able to trace the user to his clear identity to solve accusations. The power to trace and identify the signer is distributed among the servers on a threshold bases to resist traitors and prevent them from violating anonymity of the users. Due to the new security service we provided, the complexities increase accordingly, yet, our solution still efficient and practical.

## 10. REFERENCES

[1] Yair Amir, Yongdae Kim, Cristina Nita-Rotaru, and Gene Tsudik. On the performance of group key agreement protocols. *ACM Trans. Inf. Syst. Secur.*, 7(3):457–488, August 2004.

[2] Alapan Arnab and Andrew Hutchison. Digital rights management-an overview of current challenges and solutions. In *Proceedings of Information Security South Africa (ISSA) Conference 2004*, 2004.

[3] Alapan Arnab and Andrew Hutchison. Digital rights managementa current review. *Technical report*, 2004.

[4] Alapan Arnab and Andrew Hutchison. Requirement analysis of enterprise drm system. In *In proceedings of ISSA'05*, pages 1–14, 2005.

[5] Endre Bangerter, Jan Camenisch, and Ueli Maurer. Efficient proofs of knowledge of discrete logarithms and representations in groups with hidden order. In *In PKC 2005, LNCS 3386*, pages 154–171. Springer-Verlag, 2005.

[6] J. Bar-Ilan and D. Beaver. Non-cryptographic fault-tolerant computing in constant number of rounds of interaction. In *Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing*, PODC '89, pages 201–209, New York, NY, USA, 1989. ACM.

[7] Franco Bartolini, A Piva, A Fringuelli, M Barni, et al. Electronic copyright management systems: Requirements, players and technologies. In *Database and Expert Systems Applications, 1999. Proceedings. Tenth International Workshop on*, pages 896–898. IEEE, 1999.

[8] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *Proceedings of the 22Nd International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'03, pages 614–629, Berlin, Heidelberg, 2003. Springer-Verlag.

[9] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. pages 136–153. Springer-Verlag, 2004.

[10] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 1–10, New York, NY, USA, 1988. ACM.

[11] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. *J. Cryptol.*, 22(1):114–138, December 2008.

[12] Dion Boesten and Boris kori. Asymptotic fingerprinting capacity in the combined digit model. In Matthias Kirchner and Dipak Ghosal, editors, *Information Hiding*, volume 7692 of *Lecture Notes in Computer Science*, pages 255–268. Springer Berlin Heidelberg, 2013.

[13] Jan Camenisch and Jens Groth. Group signatures: Better efficiency and new theoretical aspects. In Carlo Blundo and Stelvio Cimato, editors, *Security in Communication Networks*, volume 3352 of *Lecture Notes in Computer Science*, pages 120–133. Springer Berlin Heidelberg, 2005.

[14] Chin-Chen Chang and Jen-Ho Yang. A group-oriented digital right management scheme with reliable and flexible access policies. *I. J. Network Security*, 15(6):471–477, 2013.

[15] David Chaum and Torben P. Pedersen. Wallet databases with observers. In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '92, pages 89–105, London, UK, UK, 1993. Springer-Verlag.

[16] David Chaum and Eugène Van Heyst. Group signatures. In *Advances in CryptologyEUROCRYPT91*, pages 257–265. Springer, 1991.

[17] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, SFCS '87, pages 427–438, Washington, DC, USA, 1987. IEEE Computer Society.

[18] Rosario Gennaro, Stanisaw Jarecki, Hugo Krawczyk, and Tal Rabin. Robust threshold DSS signatures. In *Advances in Cryptology EUROCRYPT 96*, volume 1070 of *Lecture Notes in Computer Science*, pages 354–371. Springer Berlin Heidelberg, 1996.

[19] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *J. Cryptol.*, 20(1):51–83, January 2007.

[20] Maged H Ibrahim, IA Ali, II Ibrahim, and AH El-sawi. A robust threshold elliptic curve digital signature providing a new verifiable secret sharing scheme. In *Circuits and Systems, 2003 IEEE 46th Midwest Symposium on*, volume 1, pages 276–280. IEEE, 2003.

[21] Maged H. Ibrahim, I. I. Ibrahim, and A. H. El-Sawy. Fast three-party shared generation of rsa keys without distributed primality tests. In *in Proceedings of the Information Systems: New Generations (ISNG04*, 2004.

[22] Maged Hamada Ibrahim. Verifiable threshold sharing of a large secret safe-prime. In *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*, volume 1, pages 608–613. IEEE, 2005.

[23] Maged Hamada Ibrahim. Eliminating quadratic slowdown in two-prime RSA function sharing. *I. J. Network Security*, 7(1):106–113, 2008.

[24] Maged Hamada Ibrahim. Resisting traitors in linkable democratic group signatures. *International Journal of Network Security*, 9(1):51–60, 2009.

[25] Maged Hamada Ibrahim. Efficient robust and secure E-DRM with encrypted content search. *International Journal on Information (Information-Tokyo)*, 18(6(A)):2531–2546, 2015.

[26] Maged Hamada Ibrahim. Secure and robust digital rights management protocol with efficient storage. *International Journal on Information (Information-Tokyo)*, 18(2):625–640, February 2015.

[27] Ingemar Ingemarsson and Gustavus J. Simmons. A protocol to set up shared secret schemes without the assistance of mutually trusted party. In *Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques on Advances in Cryptology*, EUROCRYPT '90, pages 266–282, New York, NY, USA, 1991. Springer-Verlag New York, Inc.

[28] Aggelos Kiayias and Moti Yung. Group signatures with efficient concurrent join. In *In proceedings of EUROCRYPT 05, LNCS series*, pages 198–214. Springer-Verlag, 2005.

[29] Yongdae Kim, Adrian Perrig, and Gene Tsudik. Tree-based group key agreement. *ACM Trans. Inf. Syst. Secur.*, 7(1):60–96, February 2004.

[30] Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups. In Huaxiong Wang, Josef Pieprzyk, and Vijay Varadharajan, editors, *Information Security and Privacy*, volume 3108 of *Lecture Notes in Computer Science*, pages 325–335. Springer Berlin Heidelberg, 2004.

[31] Anna Lysyanskaya, RonaldL. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In Howard Heys and Carlisle Adams, editors, *Selected Areas in Cryptography*, volume 1758 of *Lecture Notes in Computer Science*, pages 184–199. Springer Berlin Heidelberg, 2000.

[32] Mark Manulis. Democratic group signatures - on an example of joint ventures. In *Proceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS 2006*, 2006.

[33] Mark Manulis, Ahmad-Reza Sadeghi, and Jrg Schwenk. Linkable democratic group signatures. In Kefei Chen, Robert Deng, Xuejia Lai, and Jianying Zhou, editors, *Information Security Practice and Experience*, volume 3903 of *Lecture Notes in Computer Science*, pages 187–201. Springer Berlin Heidelberg, 2006.

[34] Deirdre K Mulligan, John Han, and Aaron J Burstein. How drm-based content delivery systems disrupt expectations of personal use. In *Proceedings of the 3rd ACM workshop on Digital rights management*, pages 77–89. ACM, 2003.

[35] Jaehong Park, Ravi Sandhu, and James Schifalacqua. Security architectures for controlled digital information dissemination. In *Computer Security Applications, 2000. ACSAC'00. 16th Annual Conference*, pages 224–233. IEEE, 2000.

[36] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '91, pages 129–140, London, UK, UK, 1992. Springer-Verlag.

[37] Michael O. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *J. ACM*, 36(2):335–348, April 1989.

[38] Jason K. Resch and James S. Plank. Aont-rs: Blending security and performance in dispersed storage systems. In *Proceedings of the 9th USENIX Conference on File and Stroage Technologies*, FAST'11, pages 14–14, Berkeley, CA, USA, 2011. USENIX Association.

[39] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *Proceedings of 7th international conference on the theory and application of cryptology and information security*, pages 554–567. Springer-Verlag, 2001.

[40] C. P. Schnorr. Efficient signature generation by smart cards. *J. Cryptol.*, 4(3):161–174, January 1991.

[41] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979.

[42] Ahmed H. Soliman, Maged H. Ibrahim, and Adel E. El-Hennawy. Improving security and efficiency of enterprise digital rights management. In *proceedings of the 6th IEEE International Conference on Computing, Communications and Networking Technologies (ICCCNT 2015)*. IEEE, 2015.

[43] Gábor Tardos. Optimal probabilistic fingerprint codes. In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*, STOC '03, pages 116–125, New York, NY, USA, 2003. ACM.

[44] L.R. Welch and E.R. Berlekamp. Error correction for algebraic block codes, dec 30 1986. US Patent 4,633,470.

[45] Yang Yu and Tzi-cker Chiueh. Enterprise digital rights management: Solutions against information theft by insiders. *Research Proficiency Examination (RPE) report TR-169*, 2003.