October 11, 2015

Enhancing Efficiency of Enterprise Digital Rights Management

Ahmed Soliman
Maged Ibrahim, *Helwan University*
Salwa El-Ramli

Enhancing Efficiency of Enterprise Digital Rights Management

Ahmed H. Soliman
ECE Dept., Faculty of Engineering
MSA University
6th of October City, Egypt
ahatem@msa.eun.eg

Maged H. Ibrahim
Electronics, Communications and Computers Dept.
Faculty of Engineering, Helwan University
Helwan, Egypt
mhii72@gmail.com

Salwa H. El-Ramly
ECE Dept., Faculty of Engineering
Ain Shams University
Cairo, Egypt
salwa_elramly@eng.asu.edu.eg

***Abstract*—Most of private enterprises and governmental institutions are in an increasing need for enterprise-oriented digital rights management (E-DRM) schemes. E-DRM schemes provide protection to digital contents that contain corporates' sensitive information and prevent unauthorized access to these data. Previous work proposed a storage reliable and efficient E-DRM systems based on the information dispersal algorithm. In this paper, we propose a computationally enhanced information dispersal and reconstruction algorithms. We achieve significant reduction in the computational complexity without affecting the E-DRM system security and with comparable storage requirements.**

Keywords*—*digital rights management; information dispersal algorithm; secure storage; information security.

I. Introduction

With the continuous development of computer technologies, most of private enterprises and governmental institutions replace traditional printed documents with digital files to provide efficiency and flexibility to their business. Digital rights management (DRM) schemes used in various commercial applications to provide content protection, content distribution, and access authorization [1]. Enterprise digital rights management (E-DRM) schemes provide information protection from insider malicious behaviors in addition to managing access rights for sensitive data [2]. Various E-DRM schemes have been proposed in the recent years [3]–[6].

Soliman et al. [6] proposed a system for an efficient and secure E-DRM. In their system, the author of a digital content encrypts this content and uploads it to a specific server (authority server) which will use the information dispersal algorithm (IDA) [7] to generate n shares of the encrypted content. The authority server then computes the hash of each share and stores it for checking the integrity of the shares when reconstructing the content in the future. The authority server then sends each share to the corresponding storage server. When reconstructing the content, only t +1 shares are needed to reconstruct the digital content.

Soliman et al.'s system specifies a simple way to implement the IDA algorithm without taking into account the significant increase in the computations overhead when dealing with large digital contents. Therefore, in this paper, we propose a computationally enhanced information dispersal and reconstruction algorithm. We achieve significant reduction in the computational complexity without affecting the E-DRM system security, and with comparable storage requirements.

The remainder of this paper is structured as follows: A review of Soliman et al.'s scheme in Sect. II. In Sect. III, the limitations of the previous system are mentioned, and we propose our enhancements. We discuss our concept in Sect. IV. Finally, the paper is concluded in Sect. V.

II. Review of Previous E-DRM System

In this section, the E-DRM system in [6] is briefly described. The system consists of four entities: the author u_o of the digital content, the user u_i who needs to download and consume a digital content, the authority server AS which acts as the interface between the system and all users, and the n storage servers SS that store the shares of a dispersed content. Fig. 1 shows the system components.

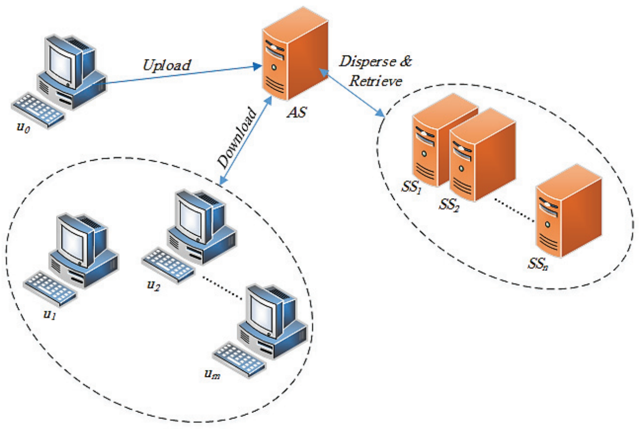

Fig. 1. Previous E-DRM system components (taken from [6])

This E-DRM system has two main protocols: the upload, and the download protocols, and it uses cryptographic tools such as blind decryption [8], and threshold cryptography [9]–[12]. A list of the used notations in [6] is shown in Table I. In the next subsections, we review these protocols and then focus on the dispersal and the reconstruction algorithms.

TABLE I. LIST OF NOTATIONS USED IN [6]

Notation	Description
u_i	The ith user
u_o	Author
AS	Authority Server
SS_i	The ith Storage server
C_{ID}	Digital content identity
R	The digital rights associated with DRM content
U_req / *D_req*	Upload / download requests sent by users
$Sig_X(.)$	Digital signature generated by entity *X*
$E_K(.)$ / $D_K(.)$	Symmetric-key encryption/decryption algorithm using key *K*
S_{ek} / S_{dk}	The storage servers' public encryption/private decryption key
S_{dki}	The ith Storage server's key-share of S_{dk}
$E_{sek}(.)$	Public-key encryption algorithm using S_{ek}
$PD_{S_{dki}}(.)$	Partial (threshold) decryption algorithm using S_{dki}
(d_i, π_i)	A partial decryption and the associated non-interactive zero-knowledge proof (NIZKP) generated by the ith storage sever
$H(.)$	Strong one-way hash function
IDA(.)	Information dispersal algorithm
IDA-rec(.)	Reconstruction algorithm of a file that is dispersed using *IDA*(.)

A. Upload Protocol

An author u_o who has a digital file *F* that he wants to upload to the system should do the following:

- Choose a random symmetric-key K and use it to encrypt *F* as: $C = E_K(F)$.
- Asymmetrically encrypt *K* using S_{ek} as: $K_e = E_{Sek}(K)$.
- Construct, and send the message: M1=<u_o, *U_req*, C, K_e, *R*> and Sig_{uo}(M1) to the *AS*.

After the *AS* receives the author's upload request, it verifies the author's signature and validates the request. If valid, then the *AS* should do the following:

- Run the IDA algorithm on the encrypted content C to generate *n* shares as: $\{C_1, C_2, \ldots . C_n\} = IDA(C)$.
- Compute and save: $h_i = H(C_i)\ \forall\ 1 \le i \le n$.
- Construct and send the message: M2$_i$=<C_i, C$_{ID}$, K_e, *R*> and Sig_{AS}(M2$_i$) to each storage server *i*.

At the end of the upload protocol, each storage server verifies the signature of the *AS* and if valid, it stores <C_i, C$_{ID}$, K_e, *R*>.

B. Download Protocol

A user u_i who wants to download and consume a selected digital content C$_{ID}$ must do his role in the download protocol. The protocol proceeds as follows:

- The user u_i should do the following:
  - o Randomly choose a blinding factor r, encrypt it as: $B = E_{S_{ek}}(r)$.
  - o Construct and send the message: M3=<u_i, *D_req*, C$_{ID}$, *B*> and Sig_{ui}(M3) to the *AS*.
- The *AS* verifies the user's signature and validates his request. If valid, then it constructs and sends the message: M4=<C$_{ID}$, *B*> and Sig_{AS}(M4) to all storage servers.
- Each SS_i verifies the *AS* signature. If valid, blind the encrypted key (i.e. compute $B.K_e$) and compute the distributed partial decryption: $(d_i, \pi_i) = PD_{S_{dki}}(B.K_e)$, then construct and send the message: M5$_i$=<SS_i, C$_{ID}$, C_i, d_i, π_i> and Sig_{SSi}(M5) to the *AS*.
- The *AS* then do the following:
  - o Verify the signatures on the received messages from the previous step. If at least t +1 are valid, then continue.
  - o Compute $H(C_i)$ of each C_i received from SS_i and compare it with the previously saved hash h_i of the same content C$_{ID}$. If they match, then this share is valid (not corrupted). If they do not, then discard this share and set the corresponding SS_i as a malicious adversary.
  - o From any valid set of t +1 shares *V*, reconstruct the encrypted content as: C =*IDA-rec*(*V*).
  - o Use the NIZKP π_i to validate the correctness of each partial decryption d_i [9], and from any valid t +1 partial decryptions reconstruct the blinded key ($r.K$).
  - o Construct and sent the message: M6=<C$_{ID}$, C, $r.K$, *R*> and Sig_{AS}(M6) to the requesting user u_i.

The user u_i verifies the *AS* signature. If valid, compute: $K = (r.K).r^{-1}$ and then: $F = D_K(C)$.

C. Content Dispersal and Reconstruction Algorithms

In this section, we will review two algorithms: *IDA*(.) and *IDA-rec*(.) that are used in the upload and the download protocols respectively.

IDA(.); is a deterministic algorithm that takes as an input a digital file C of size | C | and two parameters: *t*, *n*, and it outputs *n* shares such that any *t*+1 shares can be used to recover the original file using the algorithm *IDA-rec*(.). The disperser (*AS*) should do the following:

- Segment the file C into t +1 parts denoted by m_0, m_1, ……., m_t, each of size:

$$|m_i| = \frac{|C|}{t+1} \tag{1}$$

- Construct the polynomial:

$$f(x) = \sum_{j=0}^{t} m_j . x^j \ (mod\ p) \tag{2}$$

- Compute the shares as:

$$C_i = f(i), \quad \forall\ 1 \le i \le n \tag{3}$$

We note that the size of the chosen prime p must be greater than $|m_i|$ at least by a single bit to ensure that no information will be lost during the reconstruction process. Therefore, we can assume the prime size to be:

$$|p| = |m_i| + 1 = \left(\frac{|C|}{t+1} + 1\right) \text{ bits} \qquad (4)$$

The total storage required (i.e. sum of the sizes of all the n shares) is:

$$S_{IDA} = n|C_i| = n|p| = n\left(\frac{|C|}{t+1} + 1\right) \text{ bits} \qquad (5)$$

IDA-rec(.); is a deterministic algorithm that takes as an input a set of t+1 shares $V = \{v_{d_0}, \ldots, v_{d_t}\}$ that were dispersed using the algorithm *IDA*(.), and another set of the corresponding players' (or, servers) identities $D = \{d_i | d_i \in [1, n], 0 \le i \le t\}$.

- Construct the following system of t +1 equations with t +1 unknowns:

$$\begin{bmatrix} m_0 & \ldots\, m_i {d_0}^i \cdots & m_t {d_0}^t \\ \vdots & \ddots & \vdots \\ m_0 & \ldots\, m_i {d_t}^i \cdots & m_t {d_t}^t \end{bmatrix} = \begin{bmatrix} v_{d_0} \\ \vdots \\ v_{d_t} \end{bmatrix} (mod\, p) \qquad (6)$$

- Solve the above system by applying an elimination technique such as the Gauss's algorithm or the Gauss-Jordan's algorithm with all operations to be performed in *GF*(p).

III. The Proposed Enhancement

In this section, we provide a detailed description of how to improve the computation efficiency of the E-DRM system in [6] without affecting the security of the system. First, we give a description of some efficiency limitations in the algorithms *IDA*(.) and *IDA-rec*(.), then we describe our proposed solution to these limitations.

A. Limitations of Algorithm IDA(.)

Every time the disperser runs the algorithm *IDA*(.), he should choose a new large prime p because, from (4), the prime size depends mainly on the size of the dispersed file. However, when dispersing large files we need to choose a very large prime to be able to correctly reconstruct the dispersed file. For example, with a 100 MB file, t = 3, and n = 7, the prime size calculated from (4) will be $|p| \approx$25 MB. The problem arises from the large size of the generated prime, as common algorithms for generating a b bit prime have an average complexity of $O(b^4)$ or $O(b^4/\log b)$[13]–[16].

To practically demonstrate this limitation, we make use of the Java *BigInteger* [17] class to generate prime numbers of different bit-length. Fig.2 shows the time required for each generated prime significantly increases as the prime size increases. We note that, all tests in this paper are performed on a 4-core Intel-i5 at 2.50 GHz with 4 GB of memory and 64-bit Linux kernel 3.13.0 using a single thread.

It is now clear that the large generation time of prime in the algorithm *IDA*(.) is considered an efficiency limitation specially because the process of generating the prime is repeated each time we run this algorithm.

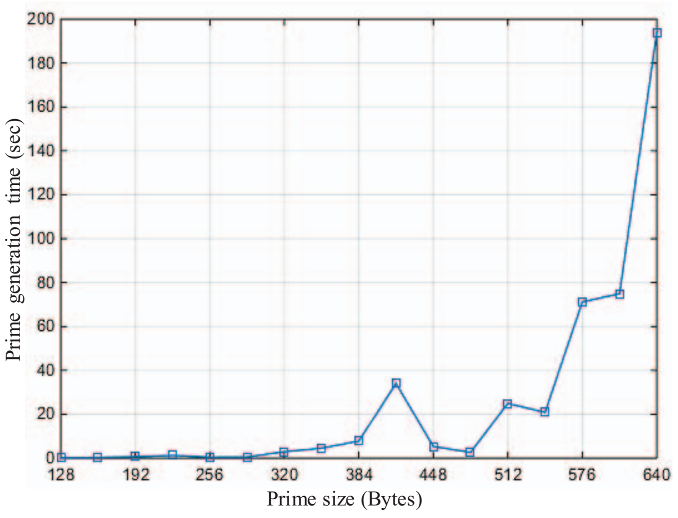

Fig. 2. Prime generation times for different sizes

B. Limitations of Algorithm IDA-rec(.)

Elimination techniques used in the file reconstruction process has a relatively large number of divisions and multiplications. For example, to solve a system of l equations with l unknowns, the Gauss's algorithm requires $(l^3 + 3l^2 - l)/3$ and the Gauss-Jordan's algorithm requires $(l^3 + 2l^2 - l)/2$ multiplications and divisions (or inverses) [18]. As the size of the original file $|C|$ increases, the size of each generated share $|C_i|$ also increases. Therefore, even for a fixed value of l the required time for running the elimination algorithm will increase as the size of the shares increases.

To practically demonstrate this limitation, we implement the Gauss-Jordan algorithm in *GF*(p) and run different tests for file reconstruction of different dispersed files but with a fixed value of t = 5 (therefore, $l = 6$ unknowns). Fig. 3 shows that the running time required for file reconstruction increases as the share size increases even with a fixed value of t.

C. Proposed Solution

To fix the above limitations, we found a way to reduce the generated shares size. As the size of each share at most equals that of the prime p. Therefore, from (4), we can reduce the prime

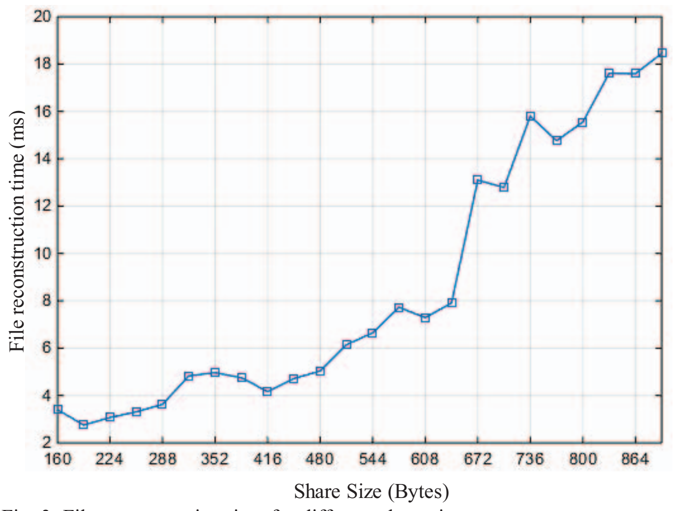

Fig. 3. File reconstruction time for different share sizes

size by decreasing the size of the file parts $|m_i|$. Our solution to this problem is to use multiple dispersing polynomials rather than a single polynomial for dispersing the same file. We call the new algorithms: *Efficient IDA*(.) or, *E-IDA*(.), and *E-IDA-rec*(.).

E-IDA(.); is a deterministic algorithm that takes as an input a digital file C of size $|C|$ and three parameters: $t, n,$ and the number of polynomials r and it outputs (nr) shares. The disperser (*AS*) should do the following:

- Segment the file C into $r(t+1)$ parts denoted by: $m_{10}, m_{11}, \ldots., m_{1t}, m_{20}, m_{21}, \ldots., m_{kj}, \ldots \ldots., m_{rt}$
- Construct the polynomials:
  $f_k(x) = \sum_{j=0}^{t} m_{kj}\, x^j \ (mod\, p)\ , \forall\ 1 \le k \le r$ (7)
- Compute the (nr) shares as:
  $C_{ik} = f_k(i), \quad \forall\ 1 \le i \le n,\ 1 \le k \le r$ (8)
- The i^{th} player (or, server) will have its share as a set of sub-shares: $\{C_{i1}, C_{i2}, \ldots, C_{ir}\}$.

The size of each file segment (or part) is:

$$|m_{kj}| = \frac{|C|}{r(t+1)} \text{ bits} \tag{9}$$

This is a significant reduction compared with that of algorithm *IDA*(.) in (1). In addition, by increasing the number of polynomials r, the size of the required prime modulus p is decreased significantly taking into account that the minimum prime size is:

$$|p| = \log_2 n + 1 \ \text{ bits} \tag{10}$$

E-IDA-rec(.); is a deterministic algorithm that takes as an input a set of t +1 shares $V = \{v_{d_0}, \ldots, v_{d_t}\}$, each share v_{d_i} is a subset of r sub-shares $v_{d_i} = \{v_{d_i,1}, \ldots, v_{d_i,r}\}$ that were dispersed using the algorithm *E-IDA*(.), and another set of the corresponding servers' identities $D = \{d_i |\ d_i \in [1,n], 0 \le i \le t\}$.

- The following two steps are repeated for $1 \le k \le r$:
  - Construct the following system of $t+1$ equations with $t+1$ unknowns:

$$\begin{bmatrix} m_{k0} & \ldots\ m_{ki}{d_0}^{i} \cdots & m_{kt}{d_0}^{t} \\ \vdots & \ddots & \vdots \\ m_{k0} & \ldots\ m_{ki}{d_t}^{i} \cdots & m_{kt}{d_t}^{t} \end{bmatrix} = \begin{bmatrix} v_{d_0,k} \\ \vdots \\ v_{d_t,k} \end{bmatrix} (mod\ p) \tag{11}$$

  - Solve the above system by applying an elimination technique as in the algorithm *IDA*-rec(.). The result is the $t+1$ segments: $m_{k0}, m_{k1}, \ldots., m_{kt}$.
- Append the resulting segments $m_{10}, m_{11}, \ldots., m_{1t}, m_{20}, m_{21}, \ldots., m_{kj}, \ldots \ldots., m_{rt}$ together to construct the original file C.

To demonstrate that the file reconstruction time is reduced by using multiple polynomial dispersing, algorithms *E-IDA*(.), and *E-IDA-rec*(.) are implemented and tested using different number of polynomials (r =1, 4, 8) with a fixed value of t =5, and using different file sizes. The results are shown in Fig. 4. We notice that dispersing a file using a single polynomial (r = 1) requires more reconstruction time than that when dispersing the same file using multiple polynomials.

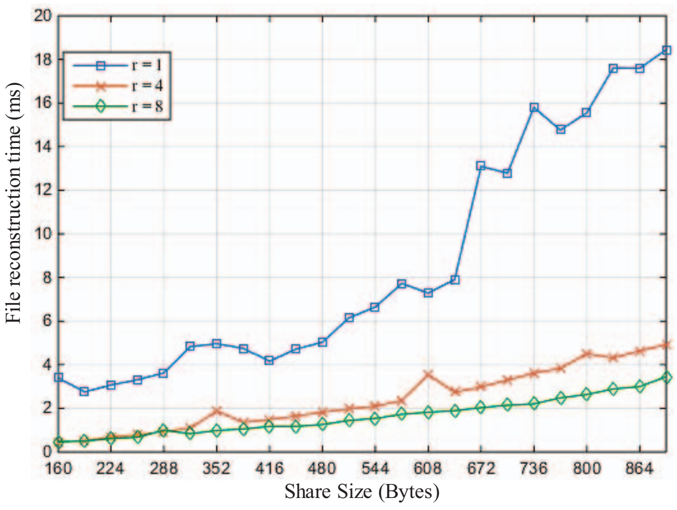

Fig. 4. Effect of multi-polynomial file dispersing on reconstruction time

IV. Discussion

A. *Total Storage Requirements of Using Algorithm E-IDA(.)*

In this section, we show the effect of using multiple dispersing polynomials, in the algorithm *E-IDA*(.), on the total storage requirement compared with the case of single polynomial used in the algorithm *IDA*(.).

We firstly calculate the total storage required for using the algorithm *E*-IDA(.). From (10), we can generally assume that:

$$|p| = G.\log_2 n + 1 \quad \text{(bits)} \tag{12}$$

, where G is an arbitrary positive integer. Recall that the part size $|m_{kj}|$ must be less than the prime size by at least 1 bit. So, we will have: $|m_{kj}| = |p| - 1$. Substituting from (12):

$$\frac{|C|}{r(t+1)} = G.\log_2 n \tag{13}$$

So, the required number of polynomials is now calculated from:

$$r = \frac{|C|}{G(t+1)\log_2 n} \tag{14}$$

The total storage needed for the algorithm *E-IDA*(.) is:

$$S_{E-IDA} = n\ \sum_{k=1}^{r} |C_{ik}| = n\, r|p| \tag{15}$$

By substituting from (12) and (14) we will get the total storage in bits:

$$S_{E-IDA} = n\left[\frac{|C|}{t+1} + \frac{|C|}{G(t+1)\log_2 n}\right] \text{bits} \tag{16}$$

Comparing the total storage for the multi-polynomial algorithm (*E-IDA*) with that of the single polynomial one (5); we notice that for a fixed $|C|, t,$ and n, the value of G controls the size of the total storage needed for *E-IDA*. As the value of G increased, the total storage required for *E-IDA* will be close to that for *IDA* (optimum value). Table II describes this by using a numeric example in which the file to be dispersed has a size of 100 MB, and we use two values for t (t = 5, 10).

It should be noted that from (14) that the value of G is inversely proportional to the number of polynomials r. Thus, a compensation between storage and computational complexities should be done by choosing the value of G that provides a suitable storage size as well as a suitable number of polynomials for an acceptable computational complexity. This adds flexibility in the design of efficient E-DRM systems.

TABLE II. A COMPARISON BETWEEN THE STORAGE OF SINGLE VS. MULTI-POLYNOMIAL IDA WHEN DISPERSING A 100 MB FILE

$t = 5, n = 11$			$t = 10, n = 21$		
Single Poly.	**Multi-Poly.**		**Single Poly.**	**Multi-Poly.**	
S_{IDA} (MB)	G	S_{E-IDA} (MB)	S_{IDA} (MB)	G	S_{E-IDA} (MB)
183.333	1	236.329	190.909	1	234.373
	20	185.983		20	193.082
	100	183.863		100	191.344
	200	183.598		200	191.126
	500	183.439		500	191.006

B. Preserving Integrity for Sub-shares

In the E-DRM system described in section II, malicious storage servers sending corrupted shares to the AS in the download protocol are detected by computing its hash and comparing it with an already saved one (when dispersing). Therefore, the shares integrity is preserved assuming the AS to be semi-honest [6].

When replacing the algorithm IDA(.) with E-IDA(.) in the upload protocol and also replacing the algorithm IDA-rec(.) with E-IDA-rec(.) in the download protocol, then using the hash of each sub-share for integrity preserving will add additional storage overhead. To overcome this problem, all sub-shares of each server could be treated as a single share then, storing the hash of all sub-shares of the i^{th} server as: $h_i = H(C_{i1}, C_{i2}, \dots, C_{ir})$. In the download protocol, the AS computes the hash of all sub-shares received from the i^{th} server and compares it with h_i. So, the storage server SS_i is considered malicious if at least one of its contributed sub-shares C_{ik} is corrupted. This will preserve the robustness of the E-DRM system.

V. CONCLUSION

We propose a computationally enhanced information dispersal and reconstruction algorithms by using multiple dispersing polynomials to disperse a digital file rather than using a single polynomial technique. We describe how these algorithms could be applied in a storage reliable E-DRM system to enhance its efficiency. A significant reduction in the computational complexity is achieved without affecting the E-DRM system security and robustness. We also propose a trade-off between computational and storage complexities, which adds more flexibility in the design of efficient E-DRM systems.

REFERENCES

[1] S. Guth, "A Sample DRM System," in *Digital Rights Management SE - 11*, vol. 2770, E. Becker, W. Buhse, D. Günnewig, and N. Rump, Eds. Springer Berlin Heidelberg, 2003, pp. 150–161.

[2] C.-C. Chang and J.-H. Yang, "A Group-oriented Digital Right Management Scheme with Reliable and Flexible Access Policies.," *I. J. Network Security*, vol. 15, no. 6, pp. 471–477, 2013.

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

[4] C.-C. L. C.-C. Lin, S.-C. W. S.-C. Wu, P.-H. C. P.-H. Chiang, and C.-C. C. C.-C. Chen, "Enterprise-Oriented Digital Rights Management Mechanism: eDRM," in *2009 International Conference on Availability, Reliability and Security*, 2009, pp. 923–928.

[5] M. H. Ibrahim, "Secure and Robust Enterprise Digital Rights Management Protocol with Efficient Storage," *International Journal on Information*, vol. 18, no. 2, pp. 625–640, 2015.

[6] A. H. Soliman, M. H. Ibrahim, and A. E. El-Hennawy, "Improving Security and Efficiency of Enterprise Digital Rights Management," in *6th International Conference on Computing, Communications and Networking Technologies (ICCCNT '15)*, 2015, In press.

[7] M. O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," *Journal of the ACM*, vol. 36, no. 2, pp. 335–348, Apr. 1989.

[8] M. Green, "Secure blind decryption," *Proceedings of the 14th international conference on Practice and theory in public key cryptography conference on Public key cryptography*, pp. 265–282, 2011.

[9] R. Cramer, R. Gennaro, and B. Schoenmakers, "A Secure and Optimally Efficient Multi-Authority Election Scheme," in *Advances in Cryptology EUROCRYPT 97*, vol. 1233, no. 5, 1997, pp. 103–118.

[10] M. H. Ibrahim, "Eliminating Quadratic Slowdown in Two-Prime RSA Function Sharing," *International Journal of Network Security*, vol. 7, no. 1, pp. 106–113, 2008.

[11] M. 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)*, 2004, pp. 7–12.

[12] M. H. Ibrahim, I. A. Ali, I. I. Ibrahim, and A. H. 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*, 2003, vol. 1, pp. 276–280.

[13] M. Joye, P. Paillier, and S. Vaudenay, "Efficient Generation of Prime Numbers," in *Cryptographic Hardware and Embedded Systems — CHES 2000 SE - 27*, vol. 1965, Ç. Koç and C. Paar, Eds. Springer Berlin Heidelberg, 2000, pp. 340–354.

[14] U. Maurer, "Fast generation of prime numbers and secure public-key cryptographic parameters," *Journal of Cryptology*, vol. 8, no. 3, pp. 123–155, 1995.

[15] P. Beauchemin, G. Brassard, C. Crépeau, C. Goutier, and C. Pomerance, "The generation of random numbers that are probably prime," *Journal of Cryptology*, vol. 1, no. 1, pp. 53–64, 1988.

[16] D. A. Plaisted, "Fast verification, testing, and generation of large primes," *Theoretical Computer Science*, vol. 9, no. 1, pp. 1–16, Jul. 1979.

[17] "Java Platform, Standard Edition 7 API Specification." [Online]. Available: http://docs.oracle.com/javase/7/docs/api/. [Accessed: 01-Jul-2015].

[18] R. W. Farebrother, *Linear Least Squares Computations*, vol. 32, no. 3. Marcel Dekker, 1988.