Journal of Information Hiding and Multimedia Signal Processing ©2024 ISSN 2073-4212
Ubiquitous International Volume **15**, Number **3**, September **2024**

An Efficient Cryptographic-based Access Control Mechanism for Cloud Storage

Hisham Dahshan

Computer Science Department
Future University in Egypt
End of 90th St., Fifth Settlement, New Cairo, Cairo, Egypt
Hisham.Mohamed@fue.edu.eg

Hazem M. Eldeeb

Department of Communications
The Military Technical College
Cairo, Egypt
hazem.eldeeb@ucalgary.ca

Alaa El-Din R. Shehata

Department of Communications
The Military Technical College
Alkhalifa Almamoun st, Cairo, Egypt
rohiem@yahoo.co.uk

Received May 2024; revised July 2024, accepted July 2024

Abstract. *When data owners outsource their encrypted data files to a cloud storage system, a set of access control rules is defined and enforced by the data owners to determine who will be granted/denied access to each file of the outsourced data. Most of the providers for cloud storage service depend on a trusted third party to enforce the access control rules. This paper proposes a new, simple, fine-grained and cryptographic-based access control mechanism that does not require the existence of a trusted third party. The proposed mechanism is based on pairing cryptography and the utilization of a processor smart card as a tamper-resistant hardware device for secure storage and secure processing. The proposed mechanism involves the utilization of a new concept named timed – expiry decryption which denies access/decryption to an encrypted file after a pre-determined period. In addition to timed – expiry feature, the proposed mechanism guarantees a fast and flat revocation time for all outsourced files.*
Keywords: Cloud storage users, access control, pairing cryptography, smart cards.

1. **Introduction.** Cloud storage service is an internet-based service provided by cloud computing technology. Data owners with limited storage resources, outsource their data files to a cloud storage server to benefit from the unlimited storage resources, anytime anywhere accessibility, pay-per-use, and high-quality data storage service. Cloud storage system could be roughly categorized as private or public storage clouds. In private storage clouds, the individual's data is stored in a group of on-premises/ off-premises servers that are owned and managed by the data owner. Apple iCloud offered by Apple Inc. is an example of private clouds. In public storage clouds, the storage servers are located off-premises, and owned and controlled by a semi-trusted third party (cloud service provider).

Amazon's S3 offered by Amazon Web Services (AWS), is an example of business public clouds [1].

2. **Related Work.** Sahai and waters [2], were the first to introduce the principal of attribute-based encryption (ABE). ABE is one-to-many and public key-based encryption scheme in which confidentiality is provided along with access control over the encrypted files. The main two variants/kinds of ABE are: key-policy ABE (KPABE) and ciphertext-policy ABE (CP-ABE). In KPABE,ciphertext is associated with a set of attributes while the private keys of the data users are associated with an access policy on the attributes. The problem with KP-ABE is that the key issuer (not the data encryptor) decides who can decrypt the file and get access to its content. On the contrary, in CP-ABE, the private keys of the data users are associated with a set of attributes while the ciphertext is associated with an access policy on the attributes. Data encryptor decides who can decrypt the file and get access to its content [3, 4]. ABE schemes are considered expensive type of encryption in terms of data size and the number of computations. When implementing the ABE schemes, the size of the ciphertext increases according to the number of associated attributes, which accordingly increases the decryption time. Moreover, decryption process consumes one pairing operation for each associated attribute [5]. In [6], the authors adopted another approach to provide access over the outsourced files by using a secure hardware device for each data user to hold the private keys. This approach provided strong protection for the private key but on the other hand, it requires some computational intervention from the cloud server, which is not available in public cloud storage servers.

3. **Technical Preliminaries.** In this section, a brief review of Smart Card as a Secure Hardware Device, and Pairing-based Cryptography will be introduced.

3.1. **Smart Card as a Secure Hardware Device.** Smart card is a very tiny computer in the form of pocket–sized plastic card. Its software and hardware resources are very limited. Yet, it is widely used in various applications that require secure processing or secure storage. Smart cards are manufactured and engineered to be tamper-resistant. Besides, the selfcontainment feature makes them resistant to attacks, as they do not need to depend upon potentially vulnerable external resources [7]. Smart cards have many security features that make them perfectly suitable for secure business transactions, secure data storage, and application processing. Smart card needs a card acceptance device (CAD) to communicate with the outer world [8]. Exchanged data is framed in a message structure known as APDU (Application Protocol Data Units). A mutual active authentication protocol takes place between the smart card and CAD to identify each other before exchanging data. Once the connection is established, exchanged data can be verified using message authentication code/ digital signature. Smart cards also protect inter-exchanged data between internal blocks by using non-constant bus-scrambling techniques [9]. Also, power analysis techniques such as visual inspection of traces are avoided by implementing time constant key operations, which eliminate time dependence in intermediate values. Cryptographic operations are completely isolated from the card operating system and their input/output visibility is restricted. This measure helps in providing protection against memory dump attacks and buffer overflow attacks [10].

3.2. **Pairing-based Cryptography.** Elliptic curve cryptography (ECC) is some sort of public key cryptography in which all the operations is executed over an elliptic curve. The set of points on the elliptic curve E is the set of solutions to the non-singular Weierstrass equation:

$$y^2 = x^3 + ax + b \tag{1}$$

Where coefficients a, and b lie in the finite field $GF(K), K > 3$, and satisfy: $4a^3+27b^2 \neq 0$. The discrete logarithm problem was re-defined over an elliptic curve cryptography to be Elliptic Curve Discrete Logarithm Problem (ECDLP). For any two arbitrary points defined over the curve $(X, Y), Y = k \cdot X$ such that k is a large integer. Given X, Y it is computationally unfeasible to find k, and k is considered the discrete logarithm of the point Y to the base X [11]. Pairing (e_m) is a mapping function over a non-singular elliptic, curve $E[m]$. The function takes as input a pair of two points P, Q from two additive cyclic subgroup G_1, G_2 (known as the source subgroups), and map them to another element (μ) in multiplicative subgroup (known as the target group) in a finite field $(\mathbb{F})$ in such a way that the exponent multiplies. Pairing must be bilinear and efficiently computable. If $G_1 = G_2$, the paring is considered symmetric with one source group G such that:

$$\mu = e_m(P, Q) \quad \forall\, P, Q \in E[m], \mu \in \mathbb{F} \tag{2}$$

$$e_m(g^a, g^b) = e_m(g, g)^{ab} \quad \forall\, a, b \in Z \; and \; g \in G. \tag{3}$$

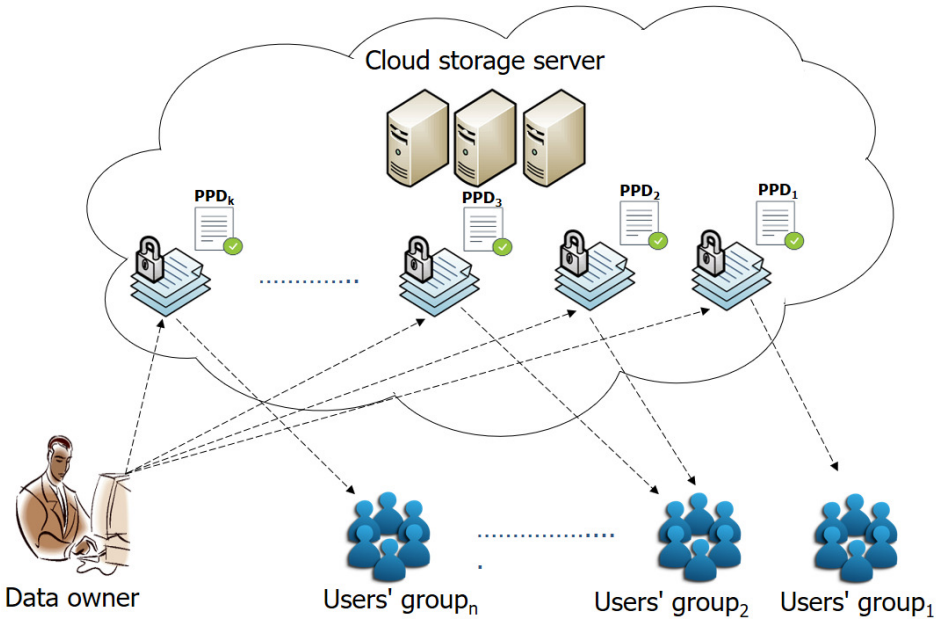

FIGURE 1. The General Architecture of the Proposed Mechanism.

4. **The Proposed Mechanism.** This work aims to build on the earlier work done in [12] to provide more secure file sharing and fine-grained access control. The main idea of the proposed access control mechanism is to grant access right to each privileged/ intended data user by enabling him/her to generate the cipher key of each outsourced encrypted file. On the other hand, denying access (revoking data user), is conducted by disabling the data user's capability to generate/ use the cipher key. Any data user can download any encrypted file and get some public parameters attached to the file metadata. However, without the cipher key, data users can get no access to the content of the encrypted files. Intended data user can generate the cipher key from the parameters acquired from three different actors (objects); smart card in the data user's possession, the encrypted file outsourced to the cloud server and a public directory in the form of a file named PPD (public privilege directory) (see Fig. 1). Data user needs three pieces of information to generate the cipher key: the data user's private key (U_i^{pr}) kept in the smart card, an elliptic curve point (P_i^g) attached to the encrypted file metadata representing a value used in the pairing operation named cipher key parameter, and an elliptic curve point (P^v) kept

in the PPD. Our design is premised on two different cryptographic techniques: symmetric key system acting as a pseudo random number generator (PRNG) and pairing over non-singular elliptic curve with large prime. Data user uses the collected pieces of information as parameters for a pairing operation to generate a parameter (seed) value. The data user uses the generated seed value and the PRNG to generate the cipher key. The proposed mechanism grants access to the intended users at the file creation by including the ID of the intended user in the pairing operation. This means that the data user would not be able to generate the cipher key for an encrypted file unless his/her ID is mathematically included in the pairing operation. On the other hand, the proposed mechanism denies access for an existing user in two different ways:

- Disabling the data user's capability to generate the cipher key by changing the pairing information exists in both the encrypted file and user information in the PPD. This action takes place when data user loses the privilege granted by the data owner due to the loss of the secure hardware device (smart card).
- Setting an expiry date for the encrypted outsourced files for a period determined by the data owner. After this period and even with the correct information needed, the file decryption would be no longer possible.

Before granting/ denying access to a data user, a setup process takes place at the data owner side to prepare the system for the actual work and define all the parameters needed for the mechanism to operate.

4.1. **Setup.** The setup process includes five different sub-processes; selecting the elliptic curve parameters, generation of private keys for data owner and all the data users, design and implementation of the cipher key generation algorithm, and finally smart card preparation/personalization.

1. Elliptic curve parameters

   The proposed mechanism is mathematically based on elliptic curve cryptography. The first step in our design is to choose an elliptic curve E, defined over finite field $\mathbb{F}_q$ with large prime order $q > 512$ bits (to achieve security level similar to symmetric key encryption with 256 bits). Two subgroups: $\mathbb{G}, \mathbb{G}_T$ are picked, with order(r) such that:

   $$r \mid \#E(\mathbb{F}_q),\ r \mid q^k - 1. \tag{4}$$

   The value (k) represents the embedding degree of the chosen curve. The embedding degree should be small in order to be suitable for pairing-based applications. A super-singular elliptic curve has been selected with an embedding degree value $k < 6$ (typically $k = 2$), which is suitable for efficient computation [13]. The value (r) is picked to be Solinas prime, with the form:

   $$2^{exp1} \pm 2^{exp2} \pm 1, \quad 0 < exp2 < exp1,\ q = 3\, mod\, 4. \tag{5}$$

   The value (h) represents the curve cofactor with regard to the working subgroup, such that $(h = n/r)$. Where, n is the order of the curve (the number of all its points), and r is the order of the subgroups: $\mathbb{G}$, and $\mathbb{G}_T$.

   The typical selected parameters for the selected elliptic curve are shown in Table 1.

2. Private Keys Generation

   as a second step in our design, a private key for the data owner and each data user is created based on his/her unique ID (e.g. email address). A mapping process takes place to map the email address of each data user to a unique index (incremental/random). This index is converted to an elliptic curve point representing his/her unique ID (U_i^{id}) that is used afterward to compute his/her unique private key. A

Table 1. Elliptic Curve Parameters

Parameter	**Value**
q	87807107996633125224377819847540498158068831994142082110286533992664756308802229570786251794226622214231558587695823174592777133673174813249251299982247 91
h	12016012264891146079388821366740534204802954401251311822919615131047207289359704531102844802183906537786776
$exp1$	159
$exp2$	107

hash function(H_1)is used to convert the selected user index to a point on the used elliptic curve group $E(\mathbb{F}_q)$. Data owner generates the private keys for all the data users by executing the following steps:

- Generates a random value (t), computes the data user's ID point ($U_i^{id} = t \cdot H_1(index(i))$) such that: $t \in \mathbb{Z}_P, H_1 : index(i) \longrightarrow E(\mathbb{F}_q)$.
- Generates a random value (s) to compute each data user's private key ($U_i^{pr} = s \cdot U_i^{id}$) such that: $s \in \mathbb{Z}_P, U_i^{id}$: data user's ID point.

Table 2. Users ID Mapping Table

User attribute(e.g. email address)	**User Index (u_i)**	**User Index (EC point)**	**User ID (U_i^{id})**	**Private Key (U_i^{pr})**
John @gmail.com	u_0	$H_1(u_0)$	$t \cdot H_1(u_0)$	$s \cdot H_1(u_0)$
Rick @yahoo.com	u_1	$H_1(u_1)$	$t \cdot H_1(u_1)$	$s \cdot H_1(u_1)$
.....	.....	.....	.....	.....
Jane.doe @usa.net	u_n	$H_1(u_n)$	$t \cdot H_1(u_n)$	$s \cdot H_1(u_n)$

Table 2, shows the full operation of the generation of both user's ID and user's private key (with incremental index).

3. Key Generation Algorithm

in order to generate the cipher key to be used to encrypt/decrypt the outsouced file, a pseudo-random number generator is used to generate a different cipher key for each encrypted outsourced file. The PRNG consists of two shift registers (R1,R2) each of 128 bit length. R2 is a kind of linear feedback with non-sparse primitive feedback polynomial which guarantees a very long period. R1 is a kind of non-linear feedback shift register that modifies the taps of the linear feedback to provide the system with the required non-linearity. Each of the cipher key and the initialization vector (IV) has a size of 128 bit.

4. Smart Card Preparation (personalization)

a smart card is issued for each data user to act as a decryption platform for the downloaded encrypted file. Firstly, the smart card is programmed with a set of commands to act as a mini-operating system to handle the card input/output process

and manage communication with data users's application programs. Each card is also programmed with the PRNG along with the data user's private key. Finally, the card is programmed with some security logic (8-digit PIN code) to restrict card usage to the intended user.

5. PPD Files Creation

a PPD file is created and configured for each group of encrypted files. Each user (U_i^{id}) in the group is assigned two records in the PPD. The first record (U_i^{PPD}) contains an elliptic curve point used in the pairing operation and represents a group private key for all the users except user (U_i^{id}). The second record (P^v) contains another elliptic curve point which is used in the pairing operation (by the data user) in addition to the revocation process (by the data owner). In order to avoid the Collusion attacks, the first record (U_i^{PPD}) is encrypted with the intended user ID:

$$U_i^{PPD} = Enc_{U_i^{id}}(H_2(s \cdot \sum_{k=0,k\neq i}^{n} U_k^{id})) \tag{6}$$

such that: H_2 is a hash function: $F_q^* \longrightarrow$ Bit string. In some cases, one/more of the users might be privileged to access files in two/more different groups of encrypted files. In this case, one PPD file is assigned for each group of encrypted files. As a result of this feature, a compromised data user might be denied access to one group of encrypted files while being able to access another group(s) of encrypted files.

4.2. **Granting Access to a New User.** Granting access is conducted at the data owner side by mathematically including all the intended users IDs in the pairing operation. On the other side (intended data user's side), the same pairing operation is executed to securely compute the cipher key parameter. Data owner performs the following steps to grant access to the intended user:

- Selects a random and public point $P \in E(\mathbb{F}_q)$.
- Computes a group ID (Q_{all}^{pub}) for (n) intended users, $Q_{all}^{pub} \in E(\mathbb{F}_q)$ such that: $Q_{all}^{pub} = \sum_{k=0,k\notin J}^{n} H_1(U_k^{id})$, J: the revoked data users's set.
- Computes the public key of the data owner: $DO^{Pub} = s.P$, s is a secret $\in \mathbb{Z}_P$.
- Computes the PPD value for each user (see Table III).
- Generates a different pairing parameters (γ_i^q, γ^u) for each outsourced file (i) such that: $\gamma_i = \gamma_i^q + \gamma^u$; where γ_i^q, and $\gamma^u \in \mathbb{Z}_P$.
- Computes $P^v = \gamma^v \cdot P$, $P_i^g = \gamma_i^g \cdot P$. The value P_i^g is included in the outsourced file metadata, while P^v is a fixed value for every data user and it is used in the PPD for revocation purpose.
- Computes the cipher key parameter (μ) for the intended group of users as follows:

$$\begin{aligned}
\mu &= H_2(e(Q_{all}^{pub},\, DO^{pub})^{\gamma_i} && (7)\\
&= H_2(e(Q_{all}^{pub},\, P)^{s\cdot\gamma_i} && (8)\\
&= H_2(e(\gamma_i \cdot \sum_{k=0,k\notin J}^{n} U_k^{id}, P \cdot s)) && (9)\\
&= H_2(e(s \cdot \sum_{k=0,k\notin J}^{n} U_k^{id}, P \cdot \gamma_i)) && (10)\\
&= H_2(e(\sum_{k=0,k\notin J}^{n} U_k^{id}, P)^{s\cdot\gamma_i}) && (11)
\end{aligned}$$

- Compute the cipher key $(K_c) = PRNG(\mu)$ and perform file encryption using the computed key.

At the intended user side, data user collects the information needed to compute the cipher key by downloading the encrypted file and decrypt his/her own PPD record. The steps are executed as follows:

- Download the encrypted file $(File_c)$.
- Extract (P_i^g) from the encrypted file.
- Extract (P^v) from the PPD.
- Decrypt PPD record and get the value (UPP^D).
- Compute the pairing operation to get the value (μ) as follows:

$$\mu = H_2(e(U_i^{PPD} + s \cdot U_i^{id}, P^v + P_i^g)) \tag{12}$$

$$= H_2(e(s \cdot \sum_{k=0,k\notin J,k\neq i}^{n} U_k^{id} + s \cdot U_i^{id}, P \cdot \gamma_i)) \tag{13}$$

$$= H_2(e(\sum_{k=0,k\notin J}^{n} U_k^{id}, P)^{s\cdot\gamma_i}) \tag{14}$$

$$= H_2(e(Q_{all}^{pub}, P)^{s\cdot\gamma_i} \tag{15}$$

- Compute the cipher key $K_c = PRNG(\mu)$.

4.3. **Revoking Access for an Existing User.** Revocation is the act of preventing a data user from the current and future decryption of the encrypted files. User revocation may take place when data user loses the privilege granted by the data owner or when the smart card of the data user is reported as lost or stolen. The direct and effective approach for revocation is the re-encryption for all the outsourced data with a new cipher key. Normally, Re-encryption is conducted by the data owner / the cloud service provider or a trusted third-party entity. When the size of data is very huge (order of Petabytes/Exabyte/..), this approach is computationally feasible but it comes with very high cost in terms of time and processing power. At the worst case, we assume that the revoked user and before being revoked, had already downloaded and decrypted all the files he is privileged to download from the cloud server. In this case, the revocation mechanism will prevent the data user from a future decryption only. This assumption contradicts with the main purpose of huge data outsourcing to an off-site storage system. Accordingly, this assumption is very highly unlikely and therefore, currently encrypted files must be protected along with the newly encrypted files against unauthorized access. In order to deny access to a file/set of files for a compromised data user, Data owner executes the following:

- Generates a new random value (α) and computes the value $(\alpha, P\ )$.
- The old point P_i^g in each file metadata is updated with new value: new $(P_i^g) = \alpha \cdot P + P_i^g$.
- The new P_i^g will result in a new pairing parameter (γ_i^{new}) and accordingly a new cipher key parameter and consequently a new cipher key for the encrypted file:

$$\mu_{new} = H_2(e(s \cdot \sum_{k=0,k\notin J}^{n} H_1(U_k^{id}), P \cdot \gamma_i^{new})) \tag{16}$$

- After this step, none of the data users (intended and revoked) would be able to recover the cipher key unless a correction action is conducted by the data owner.

- Leveraging the fact that adding a point on an elliptic curve to its inverse will result in the identity element ($P + [-P] = O$), Data owner computes the inverse of the point ($\alpha \cdot P$) and add it to every user in the PPD expect the revoked user as follows:

$$\mu = H_2(e(s \cdot \sum_{k=0, k \notin J}^{n} H_1(U_k^{id}), P \cdot \gamma_i^{new} + inv(\alpha \cdot P))) \tag{17}$$

Table 3. PPD and Revocation Process

User ID	P^v	U_i^{PPD}
U_0^{id}	$inv(\alpha \cdot P) + P^v$	$U_0^{PPD} = Enc_{U_0^{PPD}}(H_2(s \cdot \sum_{k=0,k\neq 0}^{n} U_k^{id}))$
U_1^{id}	$inv(\alpha \cdot P) + P^v$	$U_1^{PPD} = Enc_{U_1^{PPD}}(H_2(s \cdot \sum_{k=0,k\neq 1}^{n} U_k^{id}))$
U_2^{id} (revoked)	P^v	$U_2^{PPD} = Enc_{U_2^{PPD}}(H_2(s \cdot \sum_{k=0,k\neq 2}^{n} U_k^{id}))$
.....	.....	.....
U_{n-1}^{id}	$inv(\alpha \cdot P) + P^v$	$U_{n-1}^{PPD} = Enc_{U_{n-1}^{PPD}}(H_2(s \cdot \sum_{k=0,k\neq n-1}^{n} U_k^{id}))$

Every user will get the updated PPD value except the revoked data user (see Table 3).

4.4. **Timed- Expiry Decryption.** The concept of timed-release encryption (TRE) was introduced by [14]. In this concept, an encrypted message could not be decrypted by any one until a predetermined period has passed. The pre-determined period is based on a concept of computational complexity named "time-lock puzzles" [14]. Leveraging the secure processing and secure storage features of the smart card, we propose a new simple method to set an expiry date for the file decryption and we name it TED (Timed-Expiry Decryption). In this method, a certain encrypted file cannot be decrypted after a pre-determined period/date determined by the data owner. Unlike timed-release encryption, the main idea of TED is not based on a computational problem; it is based on limiting the file decryption to a certain tamper-resistant hardware device (e.g. smart card) that is capable of keeping track of the time. Smart card as a miniature computer has no internal clock neither an internal battery. Accordingly, powered-off/inactive smart card cannot keep track of the time. Nevertheless, when smart card is connected to any computer via card acceptance device, the number of milliseconds that have elapsed since 00:00:00, January first, 1970, which is known as UNIX time, can be sent to the card and kept in the card permanent memory EEPROM. The card is programmed to compare any received time to the saved time and suspend the decryption routine execution according to the comparison result. When a contact smart card is inserted in the card reader, an electrical reset happens and an output message is sent from the card to the hosting PC via the card reader. The received message is known as ATR (Answer To Reset) and it is complying with ISO/IEC 7816 standards. ATR contains some simple row of parameters showing the communication protocol used, card type, card manufacture , card serial number . . . etc. In our design, the ATR is created to include a byte code. This code revokes the hosting PC to read the UNIX time and send it to the smart card right after the cold reset. When

the data owner wants to set a validity date, he/she selects a pairing parameter γ_i to be equal to the expiry date. In order to correctly compute the pairing result and the cipher key, data user is obliged to use the pairing parameter (representing the expiry date) as is without any modification . Data owner can set an expiry date for any encrypted file by conducting the following steps:

- Before issuing the card, data owner reads the current UNIX time and save it on the card EEPROM (reference time). When the data owner wants to set an expiry date for an encrypted file, he/she sets the date, convert it to UNIX time format, and attach it to file meta-data as a pairing parameter (γ).
- Inside data user's card, a comparison process takes place between the saved UNIX time and the extracted pairing parameter (in the form of UNIX time format). According to the comparison result, the card is programmed not continue with the cipher key generation routine if the saved dated is earlier than the received date as a pairing parameter.

5. **Analysis.** In this section, security analysis and efficiency of the proposed scheme are discussed.

5.1. **Security Analysis.** The proposed mechanism security strength is mainly based on three security points: the security of the distributed cipher key parameter, the security of the generated cipher key while being stored and processed, and finally the security of the cipher key generation algorithm. We assume that the outsourced data files are secured while being transmitted over the communication networks (e.g. Internet) using Transport Layer Security (TLS) protocol or Secure Sockets Layer (SSL). We also assume that cloud storage service provider prevents unauthorized access from users sharing the same storage, and it provides: protection against data alteration, backup and redundancy techniques, and data recovery. Finally, we assume that the key generation algorithm is public knowledge and the only secret is the computed key parameter.

1. Security of the Key Generation Algorithm

   One of the basic requirement in the output of PRNG is to be indistinguishable from truly random number generator [16]. The NIST statistical test package has been used to evaluate the randomness property of the output sequence. An arbitrary long binary sequence of 8-MByte size has been generated and subjected to test under a set of 15 different NIST tests. The tested sequence has passed the different 15 statistical test with significance level equals to 98% ($\alpha = 0.02$) with the value of calculated test statistic $\leq$ threshold value for each test. Another important metric to evaluate the security strength of the pseudo random number generators is the equivalent linear complexity (ELC) [17]. ELC or linear span of random sequence $s^n = (s)_{(i=0)}^{(n-1)}$, is the smallest positive integer L such that a linear recursion relation with fixed constants $c_1, c_2 \cdots c_L; c_i \in 0, 1$ satisfying:

   $$s_j + c_1 s_{j-1} + ... + c_L s_{j-L} = 0; \quad L \leq j < n. \tag{18}$$

   In another words, it is the length of the smallest linear feedback shift register that can be used to generate the same output sequence. Berlekamp - Massey (BM) algorithm is an algorithm that is used to efficiently calculate the equivalent linear complexity of a binary sequence of length n. The time complexity of the algorithm is $O(n^2)$. The key generator algorithm has been run to generate different outputs with different sizes. The BM algorithm has been applied to each output to calculate its equivalent linear complexity.

TABLE 4. ELC for different sizes binary sequence

input size (bits)	Equivalent Linear Complexity
512	0.002432000
1024	0.005376000
2048	0.009728000
4096	0.021504000
8192	0.038912000
16384	0.08601000
32768	0.155648000
65536	0.344064000

Results have been tabulated in (Table 4) which shows that no matter the size of the input bits to be tested (n) is, the length of the equivalent linear feedback shift register length almost equals to $n/2$.

2. Security of the Generated Cipher Key

   Unlike Most of the pairing-based protocols (identity-based encryption and attribute-based encryption), the proposed mechanism does not require a secure channel for the private key distribution since that all the private keys are distrusted during the smart card issuance. Using a secure hardware device like the smart card to perform the cryptographic computations, prevents the leakage of sensitive data and the generated cipher key since they never leave the card. Once the data user unplugs the smart card of the hosting device (card reader), all the sensitive information (including the cipher key parameter and the generated cipher key itself) is erased immediately.

5.2. **Efficiency.** The time-efficiency of an access control mechanism is not only measured with how fast access is granted, but it is measured with how fast access is denied as well. In the proposed mechanism, access is granted using only one pairing operation at each side. In average, a pairing operation implemented on average speed machine (800-1000 MHz CPU clock frequency), and using Pairing-Based Cryptography (PBC) library, consumes less than 3 millisecond. On the other hand, when the card thief possesses the card, he/she will try to download as much files as he/she can, and decrypt them later. The direct re-encryption approach efficiency depends on the size of each file to be re-encrypted. The larger the file size the longer the re-encryption time. The proposed revocation consumes very short time (order of seconds) to replace the information attached to the file metadata and update all the involved PPD files. The consumed time for each file, (order of milliseconds)is fixed regardless the file size (see Fig.2).

6. **Conclusion.** In this paper, a new fine-grained and cryptographic-based access control mechanism for cloud storage users has been introduced. In order to provide strong physical protection for the cipher key, a smart card acting as a tamper-resistant hardware device has been used as a platform to execute all the cryptographic operations. The proposed mechanism grants access for intended data users using pairing-based cryptography over an elliptic curve with large prime order. The paper introduced two different methods to revoke a compromised data user's access to the encrypted files, the first by disabling the given capability to generate the cipher key, and the second by setting an expiry date for the file decryption. The PRNG used to generate the cipher key has been tested for

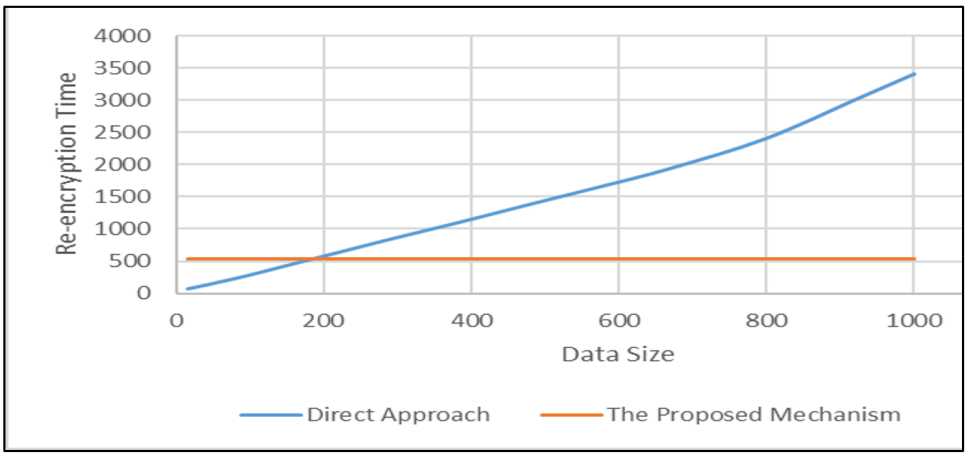

FIGURE 2. Revocation Time.

randomness and unpredictability using NIST statistical package. The long cycle of the PRNG has been assured by using the BM algorithm to measure the equivalent linear complexity for different sizes of the output.

REFERENCES

[1] Andreas Wittig; Michael Wittig, "Amazon Web Services in Action," Third Edition: An in-depth guide to AWS , Manning, 2023.
[2] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in Annual International Conference on the Theory and Applications of Cryptographic Techniques," pp. 457–473, Springer, 2005.
[3] A. Balabhadra, R. Alla, R. K. Mallipudi, N. S. Bikkina, N. Vurukonda and V. P. Kunda, "A Study On Ciphertext Policy Attribute Based Encryption," 2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2023, pp. 2398-2402.
[4] H. Elshoush, R. Mohammed, M. Abdelhameed, and A. Mohammed,"Mitigating Man-in-the-middle Attack In Online Payment System Transaction Using Polymorphic AES Encryption Algorithm," Journal of Information Hiding and Multimedia Signal Processing, vol. 14, no. 3, pp. 102–112, 2023.
[5] C. Kumar, D. Kumar, and A. K. Reddy, "Concrete attribute-based encryption scheme with verifiable outsourced decryption," arXiv preprint arXiv:1407.3660, 2014.
[6] J. K. Liu, K. Liang, W. Susilo, J. Liu, and Y. Xiang, "Two-factor data security protection mechanism for cloud storage system," IEEE Transactions on Computers, vol. 65, no. 6, pp. 1992–2004, 2016.
[7] G. Selimis, A. Fournaris, G. Kostopoulos, and O. Koufopavlou, "Software and hardware issues in smart card technology", IEEE Communications Surveys & Tutorials, vol. 11, no. 3, pp. 143–152, 2009.
[8] H. M. Eldeeb, H. M. Dahshan, and A. R. Shehata, "C2s, a new "cryptographic command set for smart card operating system," in 2018 14th International Computer Engineering Conference (ICENCO), pp. 236–241, IEEE, 2018.
[9]] B. B. Gupta, M. Quamara, A taxonomy of various attacks on smart card-based applications and countermeasures, Concurrency and Computation: Practice and Experience, 2018 John Wiley & Sons ltd, e4993.
[10] J. Lancia and G. Bouffard, "Fuzzing and overflows in java card smart cards," in SSTIC Conference, Rennes, France, 2016.
[11] M. Maas, "Pairing-based cryptography", Master's thesis, Technische Universiteit Eindhoven, 2004.
[12] H. M. Eldeeb, H. M. Dahshan, and A. R. Shehata, "A new cryptographic pairing-based security solution for cloud storage", in 2018 International Japan-Africa Conference on Electronics, Communications and Computations (JAC-ECC), pp. 73–78, IEEE, 2018.
[13] D. Freeman, M. Scott, and E. Teske, "A taxonomy of pairing-friendly elliptic curves", Journal of cryptology, vol. 23, no. 2, pp. 224–280, 2010.
[14] N. El Mrabet and M. Joye, "Guide to Pairing-Based Cryptography", Chapman and Hall/CRC, 2017.
[15] S. B. Sadkhan, "Development of Solving the ECDLP," 2021 7th International Engineering Conference "Research & Innovation amid Global Pandemic" (IEC), Erbil, Iraq, 2021, pp. 206-210.

[16] K. Marton and A. Suciu, "On the interpretation of results from the nist statistical test suite", Science and Technology, vol. 18, no. 1, pp. 18–32, 2015.
[17] J. Massey, "Shift-register synthesis and bch decoding", IEEE transactions on Information Theory, vol. 15, no. 1, pp. 122–127, 1969.