A Lightweight and Efficient Data Sharing Scheme for Cloud Computing

Saeid Rezaei, Mohammad Ali Doostari, and Majid Bayat
(Corresponding author: Saeid Rezaei)

Department of Computer Engineering, Shahed University
Tehran Province, Tehran, Hasan Abad-e-Baqerof, Iran
(Email: saeid69.rezaei@gmail.com)
(Received Aug. 20, 2018; revised and accepted Nov. 3, 2018)

Abstract

Cloud storage is a useful service of cloud computing which allows users to upload their data in the cloud and share it with others. Although, this new service is affordable and practical, but it is associated with several privacy and security challenges. Nowadays, Attribute Based Encryption (ABE) is widely used to provide secure data sharing in the distributed environment such as cloud computing. Unfortunately, most of the existing ABE schemes are not suitable for resource-constraint cloud systems, because they use expensive bilinear pairing operation and thus they cause a very high encryption and decryption computation overhead. In this paper, we propose an efficient no-pairing and revocable ABE data sharing scheme based on Elliptic Curve Cryptography (ECC) for cloud storage systems. Moreover, a comprehensive security and performance analysis shows that our scheme is both secure and efficient.

Keywords: Attribute-based Encryption; Cloud Computing; Data Sharing; Elliptic Curve Cryptography; Security

1 Introduction

In recent years, cloud computing has received increasing attention in both academia and industry. This technology provides on demand and unlimited computing services and resources for users through the Internet or a private network. People can easily upload their data into the cloud storages and users can access the shared data where and when they need them [13]. Although, this new technology brings many advantages for users and organizations, but since sensitive data is stored beyond the organizational boundaries and thus users lose physical control over their data, it faces major security and privacy threats which are the most important concerns in cloud computing.

Due to the aforementioned threats, most organizations and users afraid to take advantage of cloud storage systems and it is believed that security and privacy risks are the main obstacles in moving towards cloud services. To deal with the security threats and preserve data confidentiality, so far, many security solutions have suggested by researchers [22,29]. Among the proposed methods, data encryption is one of the most practical tools for improving the security of the stored data [8,17,18].

Although, traditional encryption primitives provide secure access control to remotely stored data, but they suffer several shortcomings such as complicated Key management process by increasing the number of users in the system, failure to support fine-grained access control and low scalability. In

addition, Data owners must always stay online to distribute keys among new users. Moreover, since traditional encryption models need to store a copy of each ciphertext for each single user with a different key, they require plenty of storage resources. To implement a secure, efficient and fine grained data sharing model, the following challenges need to be taken into consideration: firstly, data owners must provide access to data based on user's need; secondly, the scheme must allow the data owners to revoke existing users or add new users; thirdly, data confidentiality must be guaranteed against cloud server, attribute authority and unauthorized users and the users must be able to check the integrity of received data; finally, users must be able to access the shared data via their limited computing devices such as tablets and smartphones [4].

Recently, in order to overcome the said problems, Attribute-based encryption (ABE) technique has introduced [15]. ABE is a public key based one-to-many encryption technique that provides a flexible and fine-grained data access control in distributed systems such as cloud computing, smart grid, IoT and etc. Unfortunately, most of the existing data sharing schemes are not suitable for resource-constraint cloud systems, because they used expensive bilinear pairing operations and thus they involved a very high encryption and decryption computation overhead cost.

In this paper, with respect to the fact that ECC algorithm has stronger bit security than exponential-based public key cryptographic algorithms like RSA and can achieve the same level of security with smaller key sizes and higher computational efficiency, we propose a lightweight and efficient data sharing scheme for cloud storage systems. To preserve data confidentiality against unauthorized users, we use a key policy attribute based encryption (KP-ABE) and combine it with an elliptic curve encryption technique. In our scheme, expensive bilinear pairing operation in KP-ABE replaces with point scalar multiplication on ECC and makes a lightweight data sharing scheme which is quite suitable for using in computationally limited devices such as smart phones. Our proposed scheme also presents two important security requirements, user revocability and DoS attack resiliency. Finally, we compare the lightweight feature of our scheme with the existing ABE schemes and show that it is more efficient and practical than others.

1.1 Related Works

ABE can be seen as a generalization of Identity-based encryption [3] which was first proposed by Sahai and Waters. In an ABE scheme, data owner encrypts the message under a set of descriptive attributes and selects a threshold value as d. A user can decrypt the ciphertext if and only if there exist d of the given attributes in his/her key. In recent years, ABE has been widely used as a popular technique to provide user privacy and data security in distributed environment such as cloud computing. Based on the access control policy, ABE can be divided into two classes called key-policy ABE (KP-ABE) [6] and ciphertext policy ABE (CP-ABE) [2]. In a KP-ABE system, the access policy is set to user's private keys and the message is encrypted with a set of descriptive attributes. In this model, the access tree placed in the private key specifies which ciphertexts the user will be allowed to decrypt. Conversely, in CP-ABE, each user's secret key is associated with a set of attributes and sender determines an access structure on which user is allowed to decrypt the encrypted message.

Until now, several KP-ABE [14,16] and CP-ABE [19,20] schemes have been proposed by researchers, but most of them have major efficiency drawbacks and suffer from high computation overhead in encryption and decryption phases. These schemes are not suitable for mobile cloud computing scenario where users have resource constrained devices with limited computing power. Nowadays, ABE with low computation and communication costs has emerged as a hot topic.

In [27] Zhang et al. improved their previous work [26] and proposed a CP-ABE with constant computation cost and ciphertext size which supports AND-gate access policies with multiple attribute values and wildcards. Bayat and et al. [1] presented an efficient and revocable CP-ABE data sharing structure

which decreases the computation overhead expenses by supporting partial decryption. Their scheme also preserved user privacy by hiding access policy and is immune against DoS attack. Zhang et al. [25] presented a novel scheme called match-then-decrypt. In this scheme an additional matching phase is introduced before the decryption phase. Their technique works by computing special components in ciphertexts, which are used to perform the test that whether the attribute private key matches the hidden access policy in ciphertexts without decryption. A privacy aware smart health access control system (PASH) is proposed in [28], where a large universe CP-ABE scheme with partially hidden access policies is introduced to deal with both data security and user privacy issues.There are also many works proposed to make further improvements on ABE, such as [9, 10, 24]

Unfortunately, all of the above schemes are based on the expensive bilinear pairing operation and because of their high computational costs are not suitable for cloud computing systems.

Recently, Yao and et al. [23] introduced a no-pairing ECC-based KP-ABE data sharing scheme which bilinear pairing operation has been replaced with point scalar multiplication on elliptic curve. This replacement makes their scheme efficient and it is suitable for using in resource constrained devices. However, their scheme does not support user revocation and it is also responsible for user to perform data decryption completely which can take a significant amount of time and resources.

1.2 Organization

The remaining of this paper is organized as follows. Section 2 introduces our system model, definitions, security requirements and the cryptographic preliminaries. Section 3 provides the details of our proposed scheme. In Section 4 and 5, we analyze the security requirements and performance evaluation of our scheme, respectively and finally, Section 6 contains the concluding remarks of the paper.

2 Preliminaries

In this section, we first define the architecture of the proposed data sharing model along with its framework and security requirements and then we will describe the briefly review of some cryptographic primitives. The notations used in our scheme are shown in Table 1.

2.1 System Model

Our system model is composed of four entities: an attribute authority, a cloud service provider, data owners (senders) and data consumers (users). They are shown in Figure 1.

Authorized Authority (AA). The AA is a semi trusted (honest but curious) key entity that is responsible for generating global public parameters and master secret keys. It takes charge of computing corresponding private keys for users and publishes the keys among them. It also is responsible for revoking the users. The AA is assumed to be honest but curious, that is, it will not deny services to any authorized users and it will correctly follow the proposed protocol, but it is curious about the data content and it would like to obtain as much private information as possible.

Cloud Service Provider (CSP). The CSP is a powerful computing entity with unlimited resources and is responsible for collecting and storing data from the owners. It also helps users decrypt the ciphertext by computing a large amount of decryption overhead. Like AA, the CSP is assumed to be honest but curious.

Data owner. The data owner is an entity who wishes to outsource data file to the CSP. Before transmission of data, it first encrypts the data under a set of attributes.

User. It is an entity who can freely query ciphertext from the cloud server. For this purpose, it first generates a token based on its key and requests access to the data by sending that token to the CSP.

Table 1: Notations of proposed scheme

Notations	Description
p	A large prime number.
F_p	A finite field with p elements.
E	An elliptic curve over a finite field F_p.
G	A generator point on the elliptic curve E.
O	The point at infinity.
Z_p	A finite integer set with $\{0, 1, ..., p-1\}$ as its elements.
Z_p^*	Z_p - $\{0\}$.
PS	One point scalar multiplication.
AA	Attribute Authority.
CSP	Cloud Service Provider.
MK	The system master private key.
PK	The system master public key.
GP	The public key parameters of the ABE scheme.
M	The shared data.
MAC	The message authentication code.
U	The number of the possible attributes.
ω	The number of the attributes used to encrypt data.

2.2 Definitions of Our Lightweight KP-ABE

In the following, we present an overview of the algorithms used in the attribute based data sharing system.

Setup(λ , U)$\rightarrow$(MK, GP)

The setup algorithm takes as input a security parameter λ and an attribute universe U and it outputs global public parameters GP and a master key MK for the system.

Encryption(GP, M, ω)$\rightarrow CT$

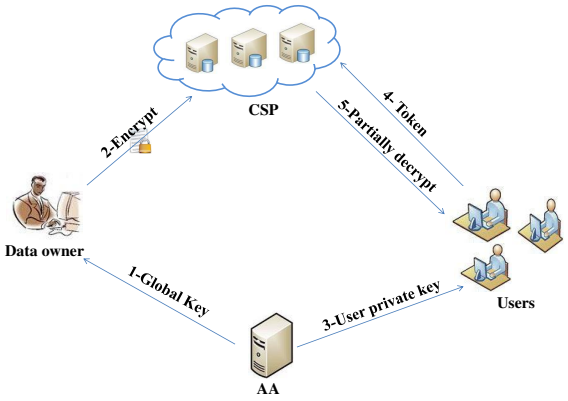

Figure 1: System model of our attribute-based data sharing

The encryption algorithm takes a message M, a set of descriptive attributes ω and the global parameters GP. It outputs a ciphertext CT.

KeyGen$(GP, MK, \Gamma) \rightarrow SK$

The key generation algorithm takes as input the global parameters GP, the master secret key MK and an access tree Γ. It generates a private key SK for each authorized user.

TokenGen$(GP, D) \rightarrow TK$

The user runs this algorithm in order to generate a decryption token TK.

Partial Decryption$(GP, TK, CT) \rightarrow CT_{Partial}$

The partial decryption algorithm takes as input the global parameters GP, the users token and the ciphertext. It outputs a partially decrypted ciphertext.

Decryption$(CT_{Partial}) \rightarrow (M, MAC_M)$

The decryption algorithm runs by user and it takes the partially decrypted ciphertext. It outputs the message M and the message authentication code MAC_M.

2.3 Security Requirements

- **Data cofidentiality** This means that only authorized users is allowed to access the data and unauthorized users, the AA and the CSP are unable to decrypt the message.
- **Data Integrity** This property guarantees the validity and accuracy of data over its entire life cycle. By using it, the user can easily detect any modifications, addition or deletion of data.
- **Denial of service (DoS) attack** Denial of service is a kind of attack which the attackers try to deny services to legitimate users. Such an attack can simply waste cloud's resources and limit the authorized users to utilize the facilities of the data sharing system.
- **Collusion resistance** Collusion resistance means a deterring process of combining keys by two or more unauthorized users in order not to allow them to decrypt a ciphertext that none of them can decrypt it individually. In this article, the CSP and the AA is assumed to be honest and do not engage in any active attack for colluding.
- **Revocation of users** User revocation is a vital issue in data sharing systems. It refers to the act of terminating a previously granted user. The revoked users should not be able to decrypt the ciphertext even if they have the corresponding keys.

2.4 Access Structure

Definition 1. (Access Structure [11]).

We denote $\mathbb{P} = \{P_1, P_2, ..., P_r\}$ be a set of attributes. A collection $\mathbb{A} \subseteq 2^{\{P_1, P_2, ..., P_r\}}$ is monotone if $\forall A_1, A_2$: if $A_1 \in \mathbb{A}$ and $A_1 \subseteq A_2$, then $A_2 \in \mathbb{A}$. An (monotone) access structure is a (monotone) collection $\mathbb{A}$ of non-empty subsets of $\mathbb{P} = \{P_1, P_2, ..., P_r\}$. That is, $\mathbb{A} \subseteq 2^{\{P_1, P_2, ..., P_r\}} / \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.

Definition 2. (Access Tree [6]).

Let T_R be the access tree with a root node R. In the access tree T_R, each non-leaf node is a threshold gate which is defined by its children and a threshold value. Let num_x is the number of children of a node x and k_x is its threshold value, then $0 < k_x \leq num_x$. The threshold gate is an OR gate if $K_x = 1$ and it is an AND gate if $K_x = n$. each leaf node x is corresponding to an attribute value and a threshold value $K_x = 1$.

We also define the parent of node x by $parent(x)$ and the function $index(x)$ returns the number associated with node x that is given by x's parent node. The children of every node x are numbered from

1 *to* num_x. *The function* $att(x)$ *is defined only if* x *is a leaf node and denotes the attribute associated with the leaf node* x *in the tree. If an attributes set* γ *satisfies the access tree* T_R, *then* $T_R(\gamma) = 1$. $T_R(\gamma)$ *is computed recursively. If* x *is a non-leaf node, evaluate* $T_x(\gamma)$ *for all children* x' *of node* x. $T_x(\gamma)$ *returns* 1 *if and only if at least* k_x *children return* 1. *If* x *is a leaf node, then* $T_x(\gamma)$ *returns 1 if and only if* $att(x) \in \gamma$.

2.5 Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is a public key cryptography that was originally introduced by Victor Miller and Neal Koblitz in 1985. Let p be a prime number and let $\mathbb{F}_p$ denotes the field of integers modulo p. An elliptic curve E over the finite field (or Galois Field) GF is defined by a cubic equation $y^2 = x^3 + a \cdot x + b$ where $a, b \in \mathbb{F}_p$ satisfy $4a^3 + 27b^2 \neq (mod\ p)$. Each different value of a and b form a different elliptic curve. The set of all points (x, y) which satisfies the above equation along with a point in infinity which denoted by ∞ lies on the elliptic curve. In the ECC, a random number is chosen as private key and the public key is a point in the curve which is constructed by multiplying the private key with the generator point G in the curve. The set of parameters to be used in ECC is presented in Table 1 .

In [5] you can see a detailed description of the ECC.

3 Our Construction

In this section, based on the preliminaries and system model, we introduce our lightweight KP-ABE scheme in detail. In the general view of the proposed data sharing scheme, the owner U_A generates a set of descriptive attributes and then he/she encrypts a message M by a symmetric cryptographic algorithm and sends the ciphertext to the cloud service provider. Here, we do not use any of the ABE expensive operations such as modular exponent or bilinear pairing. Instead, we use lightweight ECC operations along with a symmetric encryption algorithm. When a user U_B receives an associated key from the authority and wants to decrypt the stored data in the cloud servers, first the user U_B must generate a decryption token and send it to the CSP. While receiving the token, the CSP partially decrypts the stored ciphertext and sends the obtained message to the U_B. The user then can easily decrypt the ciphertext. The proposed data sharing scheme consist of six function module: *Setup*, *Encryption*, *KeyGen*, *TokenGen*, *Partial Decryption* and *Decryption*. They are described as follows:

- **Setup**$(\lambda\ ,\ U) \rightarrow (MK.GP)$

  The setup algorithm runs by the trusted attribute authority which takes as input a security parameter λ and the attribute universe U. It outputs global public parameters GP and the master key MK. For this purpose, the authority first selects a random number α from Z_P^* and it computes $PK = \alpha \cdot G$. Let $U = \{1, ..., n\}$ be a set of all attributes in the system, for each attribute $i \in U$, the authority chooses a random number $s_i \in Z_P^*$ and computes the public key of each attribute i as $P_i = s_i \cdot G$. The CSP selects a random value β as its secret key and computes $PK_{CSP} = \beta \cdot G$ as its public key. The authority does not know β and the CSP proofs the knowledge of β to the authority by using a zero knowledge proof protocol [12].The authority sets $MK = \{\alpha, \{s_1, ..., s_i\}_{i \in U}\}$ as its secret key and publishes the global parameters $GP = \{PK, PK_{CSP}, \{P_1, ..., P_i\}_{i \in U}\}$.

- **Encryption**$(GP,\ M,\ \omega) \rightarrow CT$

  The encryption algorithm takes as input a message M, a set of descriptive attributes ω and the global parameters GP. When the owner U_A wants to encrypt a message under the set of attributes

ω, he/she chooses a value d randomly from Z_P^* and computes K and F as follows:

$$\begin{aligned} K &= d \cdot PK = (k_1, k_2) \\ F &= d \cdot PK_{CSP} = (f_1, f_2) \end{aligned} \tag{1}$$

If $K = O$, the authority re-chooses d randomly from Z_p^* to computes K until $K \neq O$. we consider the point (k_1, k_2) as encryption key and integrity key respectively, and compute ciphertext C and MAC_M for message M as follows:

$$\begin{aligned} C &= ENC(M, k_1) \\ C' &= ENC(C, f_1) \end{aligned} \tag{2}$$

$$MAC_M = HMAC(M, k_2) \tag{3}$$

in Equation (2), $ENC()$ is a symmetric encryption algorithm such as AES. The message M is encrypted under the key k_1 and then it re-encrypted with the key f_1 again, this causes the ciphertext C to be hidden of the authority sight. In Equation (3), $HMAC()$ is a cryptographic hash function which it generates the hash based message authentication code for message M according to the integrity key k_2. Finally, the owner U_A computes $C_i = d \cdot P_i$ for all of the attributes in ω and uploads $CT = (\omega, C', MAC_M, \{C_i\}_{i \in \omega}, H = d \cdot G)$ to the cloud service provider.

- **KeyGen**$(GP, MK, \Gamma) \rightarrow SK$

  Upon the request of a user U_B, the keyGen algorithm computes the decryption key for U_B in following manner. It chooses a polynomial q_x for each node x in the access tree Γ. These polynomials are chosen in a top to bottom manner, starting from the root node R. For each node x in the tree Γ, the authority sets the degree d_x of the polynomial q_x to be one less than the threshold k_x of that node, that is, $d_x = k_x - 1$. For the root node R, it sets $q_R(0) = \alpha$ (note that α is the secret value of authority) and then it sets rest of the points randomly to completely fix q_R. For any other node x, it sets $q_x(0) = q_{parent(x)}(index(x))$ and chooses d_x other points randomly such as $q_R(x)$. $Index(x)$ is the unique index number of x given by its parent.

  Let Y be the set of leaf nodes in the tree Γ and $att(y)$ denotes the attribute associated with the leaf node $y \in Y$. Once the polynomials have been completed, for all leaf nodes y, the KeyGen algorithm outputs the following values:

  $$D_y = q_y(0)/s_i, i = att(y) \tag{4}$$

  Finally, the authority sends $D = (D_x, i = att(x) \text{ and } i \in \omega)$ as the private key for U_B.

- **TokenGen**$(GP, D) \rightarrow TK$

  At this phase, a user U_B generates a token TK_B based on his/her private key and sends it to the CSP to convey most of the decoding computational load to the CSP. For this purpose, U_B first selects a random number b from Z_P^* and computes $D' = \{D_x \cdot b = (q_x(0) \cdot b)/s_i\}_{i \in \omega}$. After that, the user U_B computes the point $Q = b \cdot PK_{CSP} = b \cdot \beta \cdot G = (q_1, q_2)$ and $B = b \cdot G$ and sets the token TK_B as follows:

  $$TK_B = \{B, T_B = ENC_{q_1}(D', i = att(x), i \in \omega, T)\} \tag{5}$$

  Here, we use a timestamp T to defend against DOS attack. Upon receiving the token, the CSP decrypts T_B and checks the time stamp T. If it is valid, the CSP continues the partial decryption phase. $ENC(.)$ is a symmetric encryption function such as AES.

- **Partial Decryption**$(GP, TK_B, CT) \rightarrow CT_{Partial}$

  While receiving the token TK_B, the CSP computes the decryption key q_1 by using its secret key β as $Q = \beta \cdot B = \beta \cdot b \cdot G = (q_1, q_2)$ and decrypts T_B. If validation of the timestamp T is failed, the CSP rejects the partial decryption request. Otherwise, the CSP executes the partial decryption algorithm as follows. Let x be a node of tree Γ, we first define a recursive algorithm $DecryptNode(CT, D', x)$. Let $i = att(x)$, if x is a leaf node, then $DecryptNode(CT, D', x)$ is computed as follows:

$$\begin{aligned} D'_x \cdot C_i &= D_x \cdot b \cdot C_i = q_x(0) \cdot s_i^{-1} \cdot b \cdot d \cdot s_i \cdot G \\ &= q_x(0) \cdot b \cdot d \cdot G. \end{aligned} \tag{6}$$

  This recursive algorithm outputs an element in elliptic curve group or $\perp$.

  If x be a non-leaf node, the algorithm $DecryptNode(CT, D', x)$ calls for all nodes z which are children of x, and the output is stored as F_z. Let L_x be an arbitrary k_x-sized set of child nodes z such that $F_z \neq \perp$. If there is not such a L_x, then the node was not satisfied and the function returns $\perp$ for $DecryptNode(CT, D', x)$. Otherwise, assume that $i = index(z)$ and $L'_x = \{index(z) : z \in L_x\}$, $DecryptNode(CT, D', x)$ can be calculated as follows:

$$\begin{aligned} \sum_{z \in L_x} \Delta_{i, L'_x}(0) \cdot DecryptNode(CT, D', z) &= \sum_{z \in L_x} \Delta_{i, L'_x}(0) \cdot q_z(0) \cdot b \cdot d \cdot G \\ &= \sum_{z \in L_x} \Delta_{i, L'_x}(0) \cdot q_{parent(z)}(index(z)) \cdot b \cdot d \cdot G \\ &= \sum_{z \in L_x} \Delta_{i, L'_x}(0) \cdot q_x(i) \cdot b \cdot d \cdot G \\ &= q_x(0) \cdot b \cdot d \cdot G. \end{aligned} \tag{7}$$

  Based on the above, for the root node R of the access tree Γ, the output of $DecryptNode(CT, D', R) = q_R(0) \cdot b \cdot d \cdot G = \alpha \cdot b \cdot d \cdot G$. After computing the $DecryptNode(CT, TK_B, R)$, the CSP computes $F = \beta \cdot H = \beta \cdot d \cdot G = (f_1, f_2)$ and $C = DEC(C', f_1)$. Finally, the CSP sends $CT_{Partial} = \{\omega, C, MAC_M, N = DecryptNode(CT, D', R)\}$ to the user U_B. As shown in Equation (7), the cloud service provider cannot decrypt the ciphertext because does not know the value of b and it only helps the receiving users to easily decrypt the ciphertext.

- **Decryption**$(CT_{Partial}) \rightarrow (M, MAC_M)$

  Upon receiving the
  $CT_{Partial}$, the user U_B can easily compute the decryption and integrity keys. Since $N = \alpha \cdot b \cdot d \cdot G$, the Decryption algorithm simply divides out b and recovers the keys as Equation (8).

$$\begin{aligned} C'_1 &= N/b = \alpha \cdot b \cdot d \cdot G \cdot b^{-1} \\ &= \alpha \cdot d \cdot G = (k'_1, k'_2) \end{aligned} \tag{8}$$

  The point (k'_1, k'_2) is decryption key and integrity key for message M respectively. Then the user can decrypt the message M as $M' = DEC(C, k'_1)$. the message M is correct if $HMAC(M', k'_2) = MAC_M$.

In order to revoke a user U_B, the authority securely sends all of the attributes of revoked user U_B to the CSP. When receiving the token, the CSP first checks whether all of the attributes of U_B exist in the token or not, if yes it rejects the token.

4 Security Analysis

In this section, we evaluate the security features of the proposed scheme. The security analysis focuses on the security requirements defined in section 2. The correctness of our scheme is based on the two following theorems:

Theorem 1. *A user can correctly decrypt M if and only if he holds an appropriate access structure in his/her key.*

Proof. In our scheme, each user's key is associated with an access tree where the leaf nodes are associated with attributes. A user can decrypt a ciphertext if and only if the access tree in his key is satisfied by the attributes associated with a ciphertext. □

Theorem 2. *Except for the authority, it is hard for any other parties to generate a valid secret key D.*

Proof. According to the Equation (4), in order to generate a valid secret key D, one needs the secret values S_i and α and without these values, no party can compute the valid secret key. Since these secret values are kept only by the trusted authority, other parties cannot compute the valid secret keys. Moreover, by using the property of Discrete Logarithm Problem *(DLP)*, it is almost impossible for the party to calculate the values S_i and α from the public parameters PK_{CSP} and PK respectively. Thus, the party cannot compute a valid secret key. □

In addition, our scheme achieves the following security goals.

- **Data confidentiality**

  In the proposed scheme, the data is encrypted by the data owner before uploading to the CSP. Therefore, only the users with appropriate private keys can correctly decrypt the message M and unauthorized users cannot obtain any information about the encrypted data. As defined in the TokenGen phase, the user blinds his private keys with a secret value b before generating a decryption token. Hence, when a user requests the CSP to partially decrypt the ciphertext by sending the token, the storage server cannot decrypt the ciphertext because it does not have the secret value b and thus unable to calculate the proper decryption key Equations ((6 and 7)). Another attack on the stored data can be occurred by the authority. Since it is a semi trusted entity, data confidentiality against it can be consider as another vital security criteria for secure data sharing. When the sender delivers the ciphertext to the CSP, the authority cannot decrypt it. This is because in accordance with the Encryption phase, the owner re-encrypts the ciphertext with the key f_1 and since f_1 is only computable by the sender and the CSP, the authority cannot decrypt the ciphertext C. Therefore, data confidentiality of the proposed scheme is immune against the curious authority, the CSP and unauthorized users. In addition, key escrow problem against the CSP and the authority is conquered.

- **Collusion resistance**

  This property is one of the most important security features in ABE. It means that, two or more unauthorized users cannot cooperatively act as a valid user and generate the corresponding key and decipher the encrypted data, even if they collude and combine their keys. In the keyGen phase, each user's attribute key is tied with a random polynomial so that users cannot combine their attribute keys to recover the message M. If different users combine their keys, the Equations (6 and 7) do not result in the correct value $\alpha \cdot b \cdot d \cdot G$. Therefore, our scheme achieves fully collusion secure.

- **Data integrity**

  According to the Encryption phase, the owner computes the message authentication code of the message M by using a hash function and sends it along with the other components to the CSP. After decrypting the message, the user computes the MAC again and compares it with the received one. If they both are equal, the message has not been modified. Thus, the assurance of the accuracy and consistency and correctness of data is guaranteed.

- **Denial of service attack** To achieve this, in TokenGen phase we have added a time stamp T to the token TK_B and encrypt it along with other data. When an active eavesdropper listens to the channel and steals the token, he cannot send it to the CSP repeatedly, because after decrypting the token by CSP, it first checks the time stamp and if it is valid, the CSP continues the partial decryption phase. Otherwise, the request is rejected. Thus, our scheme is immune against the DoS attack.

Table 2: The comparison of the security goals

Schemes	Key escrow	Data confidentiality	Data integrity	Collusion resistant	DoS attack resistant	Revocation of users
Yao's scheme [23]	Yes	Yes	Yes	Yes	No	No
proposed scheme	No	Yes	Yes	Yes	Yes	Yes

Table 3: Key length comparison of RSA and ECC [5]

Security level (bit)	RSA key length (bit)	ECC key length (bit)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

- **Revocation of users**

  If the user quits the system, the scheme can revoke his access right from the system. That is, the authority securely sends all of the attributes of a revoked user U_B to the CSP. When receiving the token, the CSP first checks whether all of the attributes of U_B exist in the token or not, if yes it rejects the token. After the revocation process, the revoked user cannot access to the stored data even if the user has valid secret key. Our revocation process is efficient because there is no need to update any parameters such as non-revoked secret key update or data re-encryption.

We compare the security of our scheme with the Yao's data sharing scheme. The results are shown in Table 2. As shown, both schemes can achieve the data confidentiality and data integrity and are resistance to collusion attack. But the proposed scheme, as opposed to the Yao's scheme, is free from the key escrow property and it is immune against DoS attack. Moreover, it supports the user revocation procedure.

Table 4: Symbols and their meanings

Symbols	Meanings
L_T	The bit length of the timestamp.
L_{HMAC}	The bit length of the value derived from a hash function.
L_{SymKey}	The key length of a symmetric cryptography algorithm.
L_{Data}	The size of the data to be encrypted.
L_{PriECC} & L_{PubECC}	The private and public key sizes of the 160-bit ECC.
L_{PriRSA} & L_{PubRSA}	The private and public key sizes of the 1024-bit RSA.
L_{Point}	The size of a point on the elliptic curve.
L_{G_1} & L_{G_2}	The bit length of an element in G_1 and G_2 respectively.
k, l	The size of the attribute set associated with the ciphertext and the private key of a user, respectively.
t	The number of attributes associated with the token.
u	The size of the attributes space.

Table 5: The efficiency comparison

Schemes	**Communication size (bit)**			
	CT	PK	SK	T
Yao [23]	$(2k+2)d$	$(2u+2)d$	$l \cdot d$	–
Hohenberger [7]	$(k+3)6.4d$	$(u+2)6.4d$	$19.2l.d$	–
Bethencourt [2]	$(2k+3)6.4d$	$25.6d$	$(2l+1)6.4d$	–
Waters [21]	$(2k+3)6.4d$	$(u+3)6.4d$	$(l+2)6.4d$	–
Bayat [1]	$(2k+3)6.4d$	$19.2d$	$(3l+1)6.4d$	$(3t+1)6.4d$
Zhou [30]	$19.2d$	$(6u+1)6.4d$	$(2l+1)6.4d$	–
Goyal [6]	$(k+2)6.4d$	$(u+2)6.4d$	$6.4l \cdot d$	–
Our scheme	$(2k+4)d$	$(2u+4)d$	$l \cdot d$	$(t+2.18)d$

D^* is the division operation that is performed by end user in order to compute symmetric key and integrity key.

Table 6: The efficiency comparison (cont.)

Schemes	**Comput. cost (ps)**		Policy	Access structure
	Enc.	Dec.		
Yao [23]	$1+k$	$2k-1$	KP	Tree
Hohenberger [7]	$2k+22$	$4k+40$	KP	$LSSS$
Bethencourt [2]	$4k+22$	$40k+20$	CP	$Tree$
Waters [21]	$6k+26$	$60k+36$	CP	$LSSS$
Bayat [1]	$4k+44$	20	CP	$Tree$
Zhou [30]	26	$60k+20$	CP	$And^*_{+,-}$
Goyal [6]	$2k$	$40k$	KP	$Tree$
Our scheme	$2+k$	D^*	KP	$Tree$

D^* is the division operation that is performed by end user in order to compute symmetric key and integrity key.

5 Performance Evaluation and Implement

In this section, in order to present the efficiency of our proposed scheme, we evaluate the performance of our scheme in two areas of communication cost and computation overhead and compared it with the some of the previous KP-ABE and CP-ABE schemes. Then, we implemented the communication overhead of our work and two other basic schemes [2, 6] under the same condition for the comparison purposes.

5.1 Performance Analysis

For the sake of evaluation, we use the same metrics as [23]. Based on the used operations, ABE can be divided into two categories, RSA based scheme and ECC based scheme. Versus to our scheme that is based on ECC, most of the existing ABE schemes are based on bilinear pairing, which use two cyclic groups G_1 and G_2 of prime order P and a bilinear mapping $e : G_1 \times G_1 \rightarrow G_2$. Due to using of modular exponential operation, these schemes can be called RSA based. Since bilinear mapping and modular exponentiation are expensive operations, most of the existing RSA based ABE schemes suffer from high encryption and decryption overhead. As mentioned in Section 1, ECC is a secure, efficient and scalable public key encryption system which has stronger bit security than RSA. It means that, ECC can achieve the same level of security with smaller key sizes and higher computational efficiency. Table 3 shows a comparison between RSA and ECC key lengths [5]. The security level means the cryptographic strength provided by a symmetric encryption algorithm, using an n bits key. As we can see, on the same security level the key size of ECC is much less than that of RSA.

In order to simplify the comparison process, we assume that all ABE schemes to be compared with are under the same encryption attribute set and at the same security level that is equal to 160 bit ECC. We denote this security level by d. The symbols employed in the comparison are described in Table 4. Based on the above assumption and according to the Table 4, let $|L_T| = 0.18d, |L_{HMAC}| = |L_{SymKey}| = |L_{Data}| = |L_{PriECC}| = d, |L_{Point}| = |L_{PubECC}| = 2d, |L_{PriRSA}| = |L_{PubRSA}| = |L_{G_1}| = 6.4d$ and $|L_{G_2}| = 12.8d$.

The communication overhead depends on the length of the message to be transmitted which is consist of the ciphertext, public key and private key. Thus, we consider the lengths of these three parameters as the communication overhead metrics to measure and compare the communication overhead. In addition, we analyze the computation overhead of the proposed scheme based on the encryption and decryption algorithms. In the proposed scheme, the predominant computation operation involved in encryption and decryption algorithms is point scalar multiplication and the cost of other operations such as HMAC, arithmetic and logic operations can be ignored. In an RSA ABE scheme, the most expensive operations are bilinear mapping and modular exponentiation respectively, and we ignore the cost of all other operations. In order to ease the calculation of computation overhead, the point scalar multiplication can be taken as the unit of computation overhead in ABE schemes. Since the cost of bilinear mapping and modular exponentiation is much more than the point scalar multiplication, let the cost of one bilinear mapping is equal to 20 point scalar multiplication, and one modular exponential operation is 2 point scalar multiplication [23].

Tables 5 and 6 show a performance comparison in terms of the ciphertext size, the size of system public key, private key size, computation overheads of encryption and decryption, kind of policy and the expressiveness of access policy. In this Table, CT, PK, SK, T and LSSS are abbreviation for ciphertext size, public key size, private key size, token size and linear secret sharing schemes, respectively. In comparison to Yao's scheme, the proposed scheme does not increase the private key size and it only increases $2d$ bits (320 bits) to the ciphertext size and public key size and $(k + 2.18)d$ bits to the token which are acceptable for the security enhancements. From Tables 5 and 6, we can see that the computation complexity of encryption algorithm in our scheme is almost equal to Yao's scheme, but we

convey most of the decryption computational load without disclosure of data to the CSP and the end user can decrypt the ciphertext easily and only by performing a division operation that we denote it by D^* in Tables 5 and 6. It is important to note that our scheme does not reveal any useful information to the CSP during partial decryption and thus the CSP is unable to recover the valid private key and access to the plaintext. The performance comparison with other KP-ABE and CP-ABE schemes are shown in Tables 5 and 6. From the Table, we can see that in most cases our scheme is more efficient in both communication and computation costs, as well as it achieves higher performance in privacy and security without increasing computational complexity and is quite suitable for using in computationally limited devices such as smart phones.

5.2 Implementation

We conduct simulation experiments for communication overhead of our scheme in terms of ciphertext size, public key size and private key size and compared them with Goyal et al.'s KP-ABE [6] scheme and Bethencourt et al.'s CP-ABE scheme [2] as the basic current ABE schemes. The comprehensive experiments are conducted by MATLAB on a Windows 7 machine with dual core 2.40-GHz CPU and 4-GB RAM. Figure 2 shows the comparison of ciphertext size versus the number of attributes used to create it. As can be seen, the ciphertext size on the three schemes increases linearly with the number of attributes, but by increasing number of attributes, our proposed scheme is more efficient than those two others. This efficiency is achieved by using ECC operations instead of expensive modular exponentiation and bilinear mapping operations. Figure 3 gives the public key size of our scheme, [6] and [2] versus the all number attributes used in the system. As Figure 3 shows, the public key size in our scheme is always shorter than that [6] and when the number of attributes is more than 11, the public key size in our scheme is longer than [2]. That is because the ciphertext size in [2] is independent of the number of attributes and it is always a constant value $25.6d$. Figure 4 presents the comparison of private key length versus the number of attributes used in access structure. As it is evident from Figure 4, the private key size increases linearly with the number of attributes, but with the increase of attributes, our proposed scheme act more efficient than other schemes. In total, we can say that our scheme outperform other ABE schemes in lightweight.

In addition, we simulate the communication overhead of our scheme in term of owner-to-CSP communication and user-to-CSP communication. Let n is the number of all users in the system ($n \leq 1000$) and the number of all attributes is 100 ($u = 100$). We plot the communication overhead in terms of the number of users n and the number of attributes k. We assume that each user has maximum half of the possible attributes ($k \leq 50$). Figure 5 shows the owner-to-CSP communication, where the owners encrypt the data and upload the encrypted data to the storage servers. As stated in Tables 5 and 6, the size of ciphertext is $(2k + 4)d$ bits, therefor, in the worst condition when all owners encrypt and upload their data Simultaneously, the communication overhead between the owners and the CSP for all owners is $((2k + 4)d) \times n$ bits. Next, we consider the user-to-CSP communication overhead of proposed scheme as shown in Figure 6. According to the token size in Tables 5 and 6, the bit length of a token is $(k + 2.18)d$ bits for one user. Therefore, the overall communication overhead between the users and the CSP is $((k + 2.18)d) \times n$ bits. Note that, these communication overheads are negligible by considering current communication technologies for cloud computing.

6 Conclusion

This paper introduced a revocable and lightweight data sharing system by using KP-ABE and ECC based operation. We convey most of the decryption computational load without disclosing any data to the cloud service provider. In addition, we presented detailed security analysis which shows that

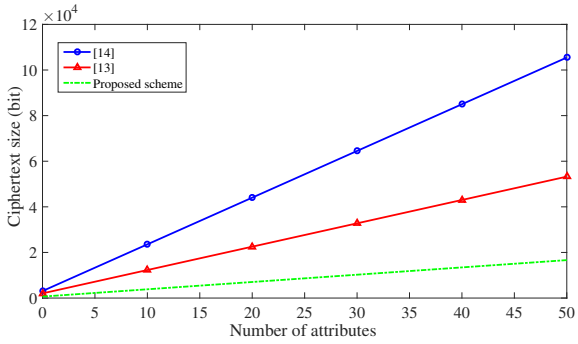

Figure 2: Comparison of ciphertext size

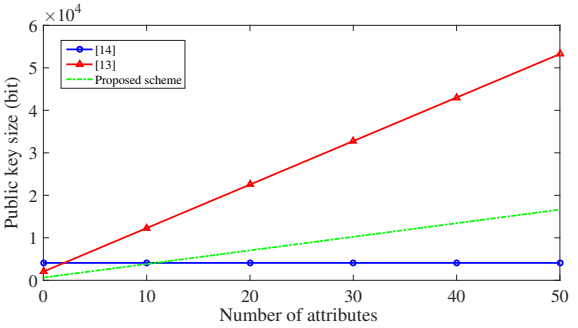

Figure 3: Comparison of public key size

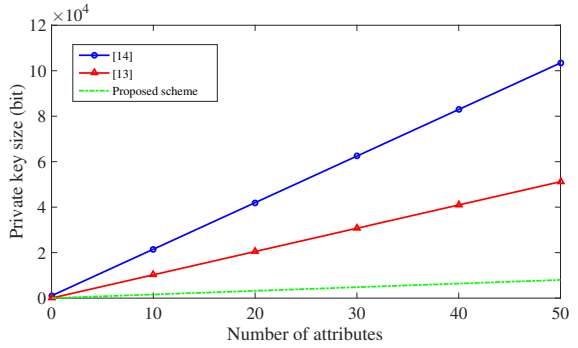

Figure 4: Comparison of private key size

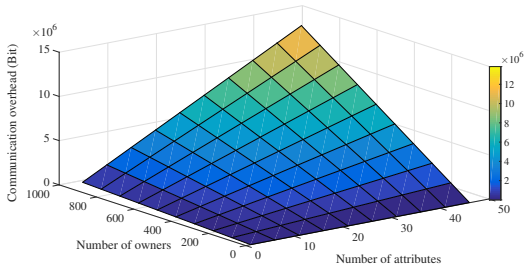

Figure 5: Owner-To-CSP communication overhead of our scheme

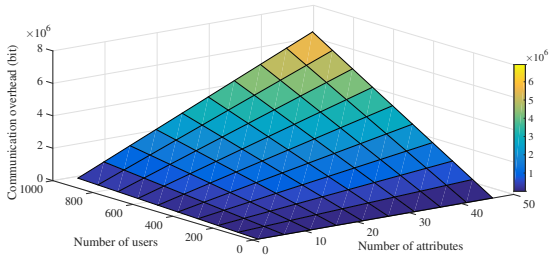

Figure 6: User-To-CSP communication overhead of our scheme

our scheme is secure enough for data sharing in the cloud computing. Moreover, we evaluated the performance of our scheme in computation and communication complexity and compared it with other ABE schemes. From the result we can see that the proposed scheme is low overhead and highly efficient. In our future research work, we will improve the generality of our scheme for using in a multi-authority environment.

Acknowledgments

The authors would like to thank the anonymous reviewers for their thorough reviews and highly appreciated comments and suggestions.

References

[1] Majid Bayat, Hamid Reza Arkian, and Mohammad Reza Aref, "A revocable attribute based data sharing scheme resilient to dos attacks in smart grid," *Wireless Networks*, vol. 21, no. 3, pp. 871–881, 2015.

[2] John Bethencourt, Amit Sahai, and Brent Waters, "Ciphertext-policy attribute-based encryption," in *2007 IEEE symposium on security and privacy (SP'07)*, pp. 321–334. IEEE, 2007.

[3] Dan Boneh and Matt Franklin, "Identity-based encryption from the weil pairing," in *Annual International Cryptology Conference*, pp. 213–229. Springer, 2001.

[4] Xin Dong, Jiadi Yu, Yuan Luo, Yingying Chen, Guangtao Xue, and Minglu Li, "Achieving an effective, scalable and privacy-preserving data sharing service in cloud computing," *computers & security*, vol. 42, pp. 151–164, 2014.

[5] Víctor Gayoso Martínez, Luis Hernández Encinas, and Carmen Sánchez Ávila, "A survey of the elliptic curve integrated encryption scheme," 2010.
[6] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security*, pp. 89–98. Acm, 2006.
[7] Susan Hohenberger and Brent Waters. "Attribute-based encryption with fast decryption,". in *Public-Key Cryptography–PKC 2013*, pp. 162–179. Springer, 2013.
[8] Caihui Lan, Haifeng Li, Shoulin Yin, and Lin Teng, "A new security cloud storage data encryption scheme based on identity proxy re-encryption.," *IJ Network Security*, vol. 19, no. 5, pp. 804–810, 2017.
[9] Jin Li, Yinghui Zhang, Xiaofeng Chen, and Yang Xiang, "Secure attribute-based data sharing for resource-limited users in cloud computing," *Computers & Security*, vol. 72, pp. 1–12, 2018.
[10] Ruixuan Li, Chenglin Shen, Heng He, Xiwu Gu, Zhiyong Xu, and Cheng-Zhong Xu, "A lightweight secure data sharing scheme for mobile cloud computing," *IEEE Transactions on Cloud Computing*, vol. 6, no. 2, pp. 344–357, 2018.
[11] Kaitai Liang, Liming Fang, Willy Susilo, and Duncan S Wong, "A ciphertext-policy attribute-based proxy re-encryption with chosen-ciphertext security," in *Intelligent Networking and Collaborative Systems (INCoS), 2013 5th International Conference on*, pp. 552–559. IEEE, 2013.
[12] Ueli Maurer, "Unifying zero-knowledge proofs of knowledge," in *International Conference on Cryptology in Africa*, pp. 272–286. Springer, 2009.
[13] San Murugesan and Irena Bojanova, "Cloud computing," *Encyclopedia of Cloud Computing*, vol. 13, no. 2, pp. 92–97, 2016.
[14] Yogachandran Rahulamathavan, Suresh Veluru, Jinguang Han, Rongxing Lu, Fei Li, and Muttukrishnan Rajarajan, "User collusion avoidance scheme for privacy-preserving decentralized key-policy attribute-based encryption," 2016.
[15] Amit Sahai and Brent Waters, "Fuzzy identity-based encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 457–473. Springer, 2005.
[16] Yanfeng Shi, Qingji Zheng, Jiqiang Liu, and Zhen Han, "Directly revocable key-policy attribute-based encryption with verifiable ciphertext delegation," *Information Sciences*, vol. 295, pp. 221–231, 2015.
[17] Ilya A Sukhodolskiy and Sergey V Zapechnikov, "An access control model for cloud storage using attribute-based encryption," in *Young Researchers in Electrical and Electronic Engineering (EIConRus), 2017 IEEE Conference of Russian*, pp. 578–581. IEEE, 2017.
[18] Sreeja Cherillath Sukumaran and Mohammed Misbahuddin, "Dna cryptography for secure data storage in cloud.," *IJ Network Security*, vol. 20, no. 3, pp. 447–454, 2018.
[19] Nyamsuren Vaanchig, Hu Xiong, Wei Chen, and Zhiguang Qin, "Achieving collaborative cloud data storage by key-escrow-free multi-authority cp-abe scheme with dual-revocation," *International Journal of Network Security*, vol. 20, no. 1, pp. 95–109, 2018.
[20] Shulan Wang, Kaitai Liang, Joseph K Liu, Jianyong Chen, Jianping Yu, and Weixin Xie, "Attribute-based data sharing scheme revisited in cloud computing," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 8, pp. 1661–1673, 2016.
[21] Brent Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *International Workshop on Public Key Cryptography*, pp. 53–70. Springer, 2011.
[22] J. Wei, W. Liu, and X. Hu, "Secure data sharing in cloud computing using revocable-storage identity-based encryption," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2018.
[23] Xuanxia Yao, Zhi Chen, and Ye Tian, "A lightweight attribute-based encryption scheme for the internet of things," *Future Generation Computer Systems*, vol. 49, pp. 104–112, 2015.

[24] Lo-Yao Yeh, Pei-Yu Chiang, Yi-Lang Tsai, and Jiun-Long Huang, "Cloud-based fine-grained health information access control framework for lightweightiot devices with dynamic auditing andattribute revocation," *IEEE transactions on cloud computing*, vol. 6, no. 2, pp. 532–544, 2018.
[25] Yinghui Zhang, Xiaofeng Chen, Jin Li, Duncan S Wong, Hui Li, and Ilsun You, "Ensuring attribute privacy protection and fast decryption for outsourced data security in mobile cloud computing," *Information Sciences*, vol. 379, pp. 42–61, 2017.
[26] Yinghui Zhang, Dong Zheng, Xiaofeng Chen, Jin Li, and Hui Li, "Computationally efficient ciphertext-policy attribute-based encryption with constant-size ciphertexts," in *International Conference on Provable Security*, pp. 259–273. Springer, 2014.
[27] Yinghui Zhang, Dong Zheng, Xiaofeng Chen, Jin Li, and Hui Li, "Efficient attribute-based data sharing in mobile clouds," *Pervasive and Mobile Computing*, vol. 28, pp. 135–149, 2016.
[28] Yinghui Zhang, Dong Zheng, and Robert H Deng, "Security and privacy in smart health: efficient policy-hiding attribute-based access control," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2130–2145, 2018.
[29] Shungan Zhou, Ruiying Du, Jing Chen, Hua Deng, Jian Shen, and Huanguo Zhang, "Ssem: Secure, scalable and efficient multi-owner data sharing in clouds," *China Communications*, vol. 13, no. 8, pp. 231–243, 2016.
[30] Zhibin Zhou, Dijiang Huang, and Zhijie Wang, "Efficient privacy-preserving ciphertext-policy attribute based-encryption and broadcast encryption," *IEEE Transactions on Computers*, vol. 64, no. 1, pp. 126–138, 2015.

Biography

Saeid Rezaei was graduated from master degree in computer science. He is a researcher in Shahed University. His research interests include Cloud Computing and Storage, cryptography, privacy and digital signature schemes.

Mohammad Ali Doostari received his B.Sc. degree in Computer Engineering from Shiraz University in 1975. He received his M.Sc and Ph.D degrees from Kyoto University of Technology in the field of Information and Electronics Engineering. Upon graduation, he had been employed in Engineering and Technical College of Shahed University as a faculty member. From 2000, he has been involved in research works on IT Security and Smart Cards. His current research interests include areas of E-Voting, E-Payment, Trusted Computing, Smart Cards and cryptography.

Majid Bayat received his Ph.D. from the Department of Mathematics and Computer Sciences at Kharzmi University in Tehran, Iran. He is presently a Research Assistant of Kharzmi university and Information Systems and Security Lab (ISSL) of Sharif University in Tehran, Iran. His research interests include VANETs , smart grids, cryptographic protocols and provable security.