Secure Cloud Storage Scheme Based On Hybrid Cryptosystem

Atanu Basu

ERP System, Indian Institute of Technology, Kharagpur - 721302.
E-mail : atanu@iitkgp.ac.in

Indranil Sengupta

Department of Computer Sc. & Engg
Indian Institute of Technology, Kharagpur 721302.
E-mail : isg@iitkgp.ac.in

Abstract

This paper presents a secure cloud storage scheme based on hybrid cryptosystem, which consists of Elliptic Curve Cryptography (ECC), Advanced Encryption Standard (AES), and one-way hash function. Here, the data owner exports large volume of encrypted data to a cloud storage provider. The exported encrypted data is over-encrypted by the cloud storage provider, and the data is sent to the requesting user. An existing hybrid cryptosystem based dynamic key management scheme with hierarchical access control has been incorporated in our scheme. The key management scheme groups users in various security classes, and helps to derive efficiently, as well as directly the secret keys of the lower order security classes. The incorporated key management scheme in our proposed scheme incurs low computational, communication, and storage overheads for key generation, and derivation purposes. The security analysis, and the simulation results run on the AVISPA tool (formal security verification tool) show that the proposed scheme is protected from the adversaries. This scheme is useful in 'owner-write-users-read' application areas, and the end users may use resource-constrained wireless mobile devices securely in this proposed scheme.

1 Introduction

We have proposed a security scheme for data storage on the cloud [1, 2, 3], where 'owner-write-users-read' application areas in public cloud is considered. The huge volumes of data of the applications are created, updated by the owner, and are stored in cloud storage. The end users having different access rights read the information securely. The subscribed readers with different access rights of a digital library access the contents of a digital library, and the owner of the digital library creates, updates the contents of the digital library hosted on a cloud storage. The research work of a various research groups of a research project, which are located in different countries of the world can be outsourced to a cloud storage. The scientists of the research groups with dynamic security clearance can access information of the cloud securely.

When data is outsourced to a cloud storage, it is a matter of concern that the outsourced data may be compromised. Also, due to increasing recent trend of using mobile devices by the users, any cloud storage scheme should support the use of mobile devices. This motivates us to design a secure lightweight cloud storage scheme, which supports the use of mobile devices by the users.

The objective of the work is to propose an efficient secure cloud storage scheme for 'owner-write-users-read' application areas, and the end users are capable to use resource-constrained wireless mobile devices in secure manner.

In this proposed scheme, the data owner encrypts huge volume of data, and outsources it to a cloud storage. A new user registers to the data owner. The data owner sends the list of valid registered users to the cloud storage provider. When a valid end user requests data to the cloud storage, the cloud storage over-encrypts [6] the requested data with the dynamic session key, and exports it to the end user. The end user decrypts the received data, and gets the requested data. Our proposed hybrid cryptosystem based dynamic key management scheme with hierarchical access control [5] is incorporated into this cloud storage scheme. In this key management scheme, the users of the higher order security class can access the data of the lower order security classes through derivation of the keys of the lower order security classes directly. In this 'owner-write-users-read' application areas, the end users can securely access the required data with their resource-constrained mobile devices, whereas the revoked users are not able to access any updated data. The pictorial diagram of the scheme is shown in Figure 1.

The novelty of this scheme is that the users in higher order security class can access data of its lower order security classes by deriving directly the keys of the lower order security classes. The security of the proposed scheme is verified through formal security verification using AVISPA tool [12, 13, 14]. Section 2 discusses about the related work of the secure cloud storage schemes while the Section 3 discusses in brief about the various key management schemes with access control in user hierarchy. Section 4 discusses about the system, network and adversary model of the proposed scheme. Section 5 discusses the proposed cloud storage scheme in detail. The performance analysis, and security analysis have been discussed in Section 6, and Section 7 respectively. In Section 8, comparison of the scheme with other related scheme has been discussed, and Section 9 concludes the paper.

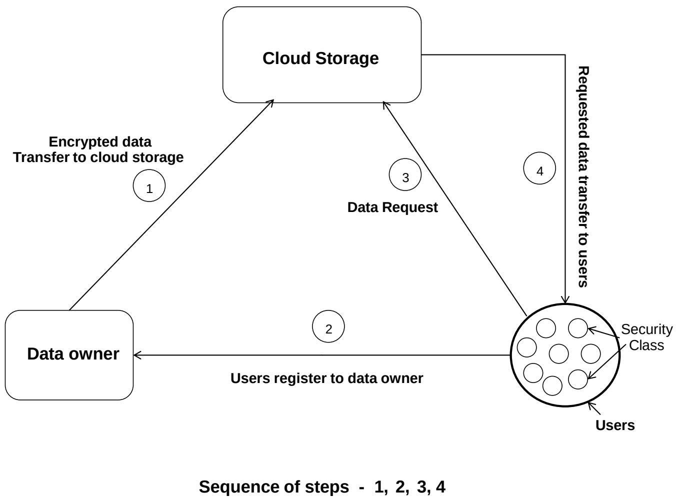

Cloud Storage
Encrypted data
Transfer to cloud storage
1
3
Data Request
4
Requested data transfer to users
Data owner
2
Users register to data owner
Security
Class
Users
Sequence of steps - 1, 2, 3, 4

Figure 1: Proposed cloud storage scheme

2 Related works of cloud storage schemes

In this section, we have discussed recent secure cloud storage schemes.

Vimercati et al. [8] proposed the merging of authorizations, as well as encryption of data, and exports the encrypted data to the cloud storage. This allows incorporation of access control with the exported data, while the data owner needs not be involved with the incorporation of access policy. The data owner uses the BEL (Base Encryption Layer) to the exported data, and the cloud storage server uses SEL (Surface Encryption Layer) termed as over-encryption [6] to the exported encrypted data. The end users use two decryption keys, SEL decryption key, and BEL decryption key for decryption of the requested data. When there are any change of access right of any user or any updation of the data is required, the over-encryption [6] is conducted to the requested data. This scheme is not free of collusion.

Wang et al. [9] proposed a secure cloud storage scheme for 'owner-write-users-read' application areas. In this scheme, the data owner exports huge volume of encrypted data to a cloud storage, where each data block is encrypted with different key. When any user requests data to the data owner, the data owner sends access certificate to the end user. After receiving the access certificate, the end user submits it to the cloud service provider. After verification of the access certificate of that user, the cloud service provider applies over-encryption [6] to the requested data using one time keypad, and transfers the requested data to that end user. But, the data owner may use the lazy revocation [7] technique when any cloud service provider denies to do the over-encryption. The end user derives large number of keys from a root key to decrypt blocks of encrypted data. This scheme is suitable for wireless mobile devices. But, the problem with the scheme is that the data owner will have to be online always for providing access certificates to the requesting users.

Yu et al. [10], and Sahai et al. [11] proposed their revocable cloud storage scheme based on ABE (Attribute Based Encryption). Though the ABE based schemes support fine grained access control, but in these schemes the secret keys of the users, and the ciphertext grow proportionally with increase in the number of associated attributres.

Chen et al. [4] proposed a secure cloud storage scheme based on Bell-LaPadula security model with user revocation feature efficiently. Here, users can upload their own data to the cloud storage, and users have read, and write access to their data. But, a user in higher order security class will have to derive key of any user in lower order security class with $O(d)$ complexity, where d is the distance (node count) between two users who are in the same predecessor-successor chain. This indicates that the key derivation mechanism of this scheme has the scalability problem when the associated users are in long predecessor-successor chain.

3 Related Works of key management schemes with hierarchical access control

The key management schemes [15, 16, 17, 18, 19, 20, 21, 22, 23, 24] with hierarchical access control describe computation of key of lower level security class from higher level security class of same predecessor-successor chain. But, the key computation of higher level security class from lower level security class of same predecessor-successor chain is not possible. We consider the user set SC is grouped into disjoint sets, i.e. $SC = \{SC_1, SC_2, \ldots, SC_n\}$, considering total number of security classes is n. The security classes are arranged into hierarchical structure as per the security clearance as shown in Figure 2. The security keys corresponding to the security classes are $SK = \{SK_1, SK_2, \ldots, SK_n\}$. The security classes are partially ordered by a binary relation '$\leq$'. If the security classes $SC_j \leq SC_i$, $i, j \in n$, then the users in security class SC_i can only access information of the users in security class SC_j. The derivation of secret keys of successor security classes are computed through direct computation or through iterations considering available public information or tokens.

Atallah et al. [22] is an efficient scheme, which solved most of the problems of key management scheme with hierarchical access control. The scheme is provable secure. But, if any node v of higher order security class tries to derive key of any node w of lower order security class in the same predecessor-successor chain, the computation time for the node v is $O(d)$, where d is the distance between nodes v and w. This indicates that the node v cannot directly compute key of node w, and the scheme is not scalable.

We have applied our secure dynamic key management scheme with hierarchical access control [5] to this proposed cloud storage scheme. This scheme [5] is an improvement of Wu et al. [24] scheme. The redundant computations have been excluded, and the scheme emerges as more computationally efficient, takes less storage space compared to Wu et al. [24], and Nikooghadam et al. [23] schemes. This scheme uses hybrid cryptosystem, i.e. Elliptic Curve Cryptography (ECC) [25, 26, 27], AES [28], and one-way hash function [28] for the key generation, and key derivation purposes. The users of the scheme [5] with proper security clearance can directly compute keys of the lower order security classes with the help of public parameters. This key management scheme suports dynamic features (addition, deletion of security classes, establishment of new relationship among security classes, generation of new security keys), and the scheme is secure in random oracle model, as well as prevents attacks from adversaries.

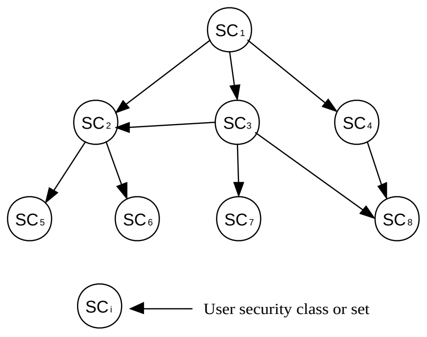


Figure 2: Hierarchical tree for user classes

4 System, network and adversary models

In this section, system, network and adversary models are described in which the proposed scheme works.

4.1 System and network models

The system model of the scheme as shown in Figure 1 consists of three components – Data Owner (DO), Cloud Storage (CS), and end user (U_v). They are described below in brief.

a) Data Owner (DO) : The DO encrypts data, and exports encrypted data to the CS. It classifies the whole data, and user security classes. The DO generates secret keys of the security classes, and publishes public keys, public parameters of the security classes.

b) Cloud Storage (CS) : The CS stores huge amount of data as requested by the DO. It is assumed that the CS is honest but curious entity, and faithfully transfers data to the U_v as requested by the U_v.

c) End user (U_v) : The valid end user U_v or reader requests data to the CS.

The communication between the DO, and the CS is done through a secure communication channel. The U_v may be connected to the DO or the CS through wired or wireless medium using public channel. The U_v may use resource-constrained wireless mobile device.

4.2 Adversary model

The various types of active, and passive adversaries or attackers are considered in this scheme. The adversaries capture or modify data of the DO, CS, and the U_v.

Any adversary or intruder may mount *impersonation* attack between the DO, and the CS, as well as between the DO, and the U_v.

Any adversary may also capture, and modify transmitted data through *replay* attack.

5 Proposed cloud storage scheme

The ECC public key cryptography is used in this scheme as the ECC is suitable for use in resource-constrained devices [27, 33] efficiently. The proposed cloud storage scheme is discussed in detail below.

5.1 Setup phase

The following steps are followed by the DO, CS, and U_v.

Step 1 : All the users $U = \{U_v\}$ where $v = \{1, 2, \ldots, N\}$ registers to the DO where N is the total number of U_v. All the U_vs are grouped into different disjoint security classes as per their hierarchy, i.e. $SC = \{SC_i\}$ where $1 \leq i \leq n$.

Step 2 : The DO randomly selects ECC private key Pri_{DO} from the finite field and generates ECC public key Pub_{DO} as $Pri_{DO}.G$. The CS also randomly selects ECC private key Pri_{CS} from the finite field, and generates ECC public key Pub_{CS} as $Pri_{CS}.G$. Similarly, each U_v selects ECC private key Pri_v from the finite field, and generates ECC public key Pub_v as $Pri_v.G$. The CS, and U_vs transfer their public keys to the DO.

Step 3 : The DO performs classification of whole data into disjoint data sets, i.e. $DS = \{DS_i\}$ where $1 \leq i \leq n$.

Step 4 : The DO selects secret key SK_i (128-bit) for each security class SC_i where $SK = \{SK_i\}$ where $1 \leq i \leq n$, and computes ECC point $Z_i = Pri_{DO}.Q_i$ for each SC_i $(1 \leq i \leq n)$.

Step 5 : Subsequently, the DO computes symmetric key CK_i for each security class SC_i is shown below.

$$CK_i = H([Q_i]_x \mid\mid [Z_i]_x),$$

where CK_i (128-bit) is computed through one-way hash operation ($H()$) after the concatenation of x-coordinates of the public key Q_i, and the ECC point Z_i for each security class SC_i. The related public parameters $m_{i,j}$ of each security class are computed as per its security clearance as described in [5] as $m_{i,j} = E_{CK_i}[SK_j]$ for each security class SC_i. The procedure to generate $m_{i,j}$ for each SC_i are repeated as per the security clearance for $1 \leq j \leq n$. The parameter $E_{CK_i}[SK_j]$ signifies that the AES symmetric encryption (E_{CK_i}) is applied on the SK_j of the SC_j.

Step 6 : Two access control matrices (Table 1 and Table 2) are created by the DO. Table 1 shows relationship between U_v, and SC_i. Table 2 shows relationship between SC_i, and DS_i. The parameter '1' in the tables indicates existence of relationship, and the parameter '0' indicates non-existence of relationship between the entities.

Step 7 : The DO initiates mutual authentication protocol, i.e. follows modified version of well-known Needham-Schroeder authentication protocol [29, 12] while initiating communication between the DO, and CS. The DO performs encryption of each data set DS_i with the corresponding key SK_i of the security class SC_i, and sebsequently exports encrypted data to the CS as shown below.

$$\text{DO} \rightarrow \text{CS} : E_{SK_i}[DS_i].$$

Step 8 : The DO publishes p, q, G, Pub_{CS}, Pub_{DO}, Pub_v, $m_{i,j}$, one-way hash function $H()$. It removes security keys SK_is, and encryption keys CK_is of the security classes.

Table 1: U_v versus SC_i

	SC_1	SC_2	SC_i	SC_n
U_1	1	0	0	0
U_2	0	1	0	0
U_v	0	0	1	0
U_N	0	0	0	1

Table 2: SC_i versus DS_i

	DS_1	DS_2	DS_i	DS_n
SC_1	1	0	0	0
SC_2	0	1	0	0
SC_i	0	0	1	0
SC_n	0	0	0	1

5.2 Data access mechanism of the users

When a U_v tries to access data of the DO, the following steps (Step 1 - 5) are followed -

Step 1 : When U_v initiates to access data of DS_i, U_v requests to the CS as shown below.

$U_v \rightarrow$ CS : $Pub_{CS}[U_v,\ RD,\ RI]$,

the information $[U_v,\ RD,\ RI]$ is encrypted with the public key of the CS. The *replay* attack is prevented by the increment of the request index (RI) each time for next requested data (RD) of U_v.

Step 2 : The CS decrypts request sent by the U_v by using private key Pri_{CS} of the CS. The CS verifies whether the U_v, and its request are valid or not. The CS also checks U_v belongs to which SC_i from Table 1. It also verifies whether the corresponding SC_i has access permission to the DS_i from Table 2.

Step 3 : After checking validity of the request of U_v, the CS randomly selects any element from finite field, and generates session key Ses_v (160-bit) for the U_v using $H()$. The CS encrypts Ses_v with public key Pub_v of that particular Ses_v, and sends it to that U_v.

CS $\rightarrow U_v$: $Pub_v\ [Ses_v]$

Step 4 : The CS over-encrypts RD with Ses_v of that U_v, and sends it to the U_v as shown below.

CS $\rightarrow U_v$: $[Ses_v \oplus E_{SK_i}[RD]]$.

The above XOR ($\oplus$) operation is performed between Ses_v (160-bit), and each 160-bit block of $E_{SK_i}[RD]$.

Step 5 : The U_v after receiving the data, decrypts the over-encrypted ciphertext through the XOR operation using the Ses_v and subsequently decrypts the data with the SK_i. Now, U_v gets the RD. It should be mentioned here that each U_v who is in SC_i retrieves the security key SK_i from corresponding $m_{i,j}$ through derivation of the key CK_i as described in Step 5 of Section 5.1.

Any revoked U_v with the retained SK_i may eavesdrop or sniff updated data destined for another U_v of the same SC_i. The over-encryption technique prevents the revoked U_v to access the updated data. This technique also helps protection of data, as the users may switch among the security classes.

The integrity of the received data may be verified by the U_v through the integrity checking schemes [30, 31, 32].

The pictorial diagram of key generation and derivation of the DO, CS and the users are shown in Figure 3.

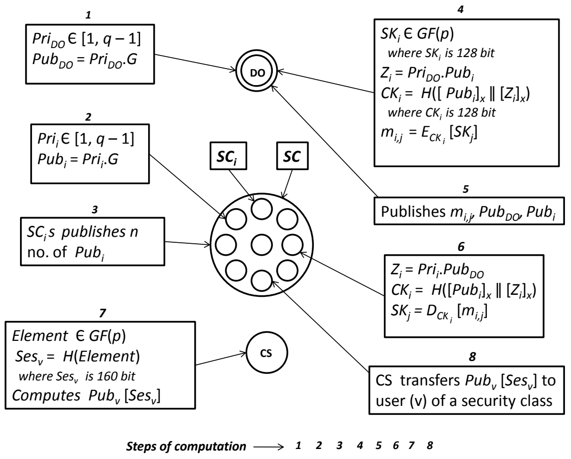

1
$Pri_{DO} \in [1, q-1]$
$Pub_{DO} = Pri_{DO}.G$
DO
4
$SK_i \in GF(p)$
where SK_i is 128 bit
$Z_i = Pri_{DO}.Pub_i$
$CK_i = H([Pub_i]_x \parallel [Z_i]_x)$
where CK_i is 128 bit
$m_{i,j} = E_{CK_i}[SK_j]$
2
$Pri_i \in [1, q-1]$
$Pub_i = Pri_i.G$
SC_i
SC
5
Publishes $m_{i,j}$, Pub_{DO}, Pub_i
3
SC_is publishes n no. of Pub_i
6
$Z_i = Pri_i.Pub_{DO}$
$CK_i = H([Pub_i]_x \parallel [Z_i]_x)$
$SK_j = D_{CK_i}[m_{i,j}]$
7
$Element \in GF(p)$
$Ses_v = H(Element)$
where Ses_v is 160 bit
Computes $Pub_v [Ses_v]$
CS
8
CS transfers $Pub_v [Ses_v]$ to user (v) of a security class
Steps of computation ⟶ 1 2 3 4 5 6 7 8

Figure 3: Pictorial diagram of key generation & derivation

6 Performance analysis

The computational and overhead storage overheads of the proposed scheme are due to generation of keys, and storage of keys respectively. The communication overhead of the scheme is due to transfer of keys to the CS, and the U_v. The computational overhead also depends on number of encryption, and decryption of data requested by the U_v to be performed by the DO, CS and the U_v. Here, for performance analysis, and comparison purposes, it is considered that the elliptic curve is defined over the finite field $GF(p^m)$ where $p = 2$, and $m =$ 163-bit which provides enough security in practical scenario.

The notations used for performance analysis and comparison purposes are shown in Table III.

6.1 Computational overhead

T_{DO} (computational overhead in the DO) : This is due to generation of ECC public key of the DO, generation of symmetric keys CK_i (128-bit) of the n

Table 3: Notations used in the scheme

T_{EC_MUL}	Time to execute ECC modular multiplication
T_{HASH}	Time to execute hash function
T_{AES}	Time to execute AES operation
v_i	The number of successor security classes of a security class
l	$(v_i + 1)$ no. of data sets

numbers of security classes, generation of $(v_i + 1)$ numbers of $m_{i,j}$ public parameters for each security class as per security clearance of each security class, and generation of session keys of N numbers of U_v. Then, T_{DO}
= Time required for generation of $[Pub_{DO}$ + (n no. of $CK_i + m_{i,j}$ of the SC) + (N no. of session keys)]
$= T_{EC\_MUL} + (n.T_{EC\_MUL} + n.T_{HASH} + \sum_{i=1}^{n}(v_i + 1).T_{AES}) + N.T_{HASH}]$
$= (n + 1).T_{EC\_MUL} + (n + N).T_{HASH} + \sum_{i=1}^{n}(v_i + 1).T_{AES}$.

T_{CS} (computational overhead in the CS) : This is due to time required for generation of Pub_{CS}
$= T_{EC\_MUL}$.

T_{U_v}(computational overhead in the T_{U_v}) : The computational overhead in the U_v as a member of a secuirty class, T_{U_v}
= Time required for generation of $[Pub_v$ + derivation of $CK_i] = T_{EC\_MUL} + [T_{EC\_MUL} + T_{HASH} + T_{AES}]$
$= 2.T_{EC\_MUL} + T_{HASH} + T_{AES}$.

6.2 Storage overhead

S_{DO} (storage overhead of DO) : The DO stores its own ECC private key (163-bit), secret key (SK_i) and encryption key (CK_i) of the n security classes, current session key (Ses_v) of the N number of U_v. Then, S_{DO}
= [163 + n.128 + n.128 + N.160]-bit

S_{CS} (storage overhead of CS) : This is due to storage of own ECC private key (163-bit), and N no. of session keys. = [163 + N.160]-bit

S_U (storage overhead of U) : Any U_v as a member of a security class stores own ECC private key (163-bit), session key and maximum $(v_i + 1)$ security keys, S_U
= [163 + 160 + $(v_i + 1)$.128]-bit
= [128.v_i + 451]-bit.

6.3 Communication overhead

The communication overheads of the DO, CS and the U_v have been considered taking into account the overhead of transferring keys only.

The DO transfers the session keys (Ses_v) to the N users and to the DO.

C_{DO} (communication overhead in DO)
= transfer of N no. of session keys to the U_v and the CS

$= N.(160 + 160)$-bit
$= N.320$-bit.

C_{CS} (communication overhead in CS) : This is due to transfer of ECC public key having x and y coordinate.
$= 2.163$
$= 326$-bit.

C_{U_v} (communication overhead in U_v) : This is due to transfer of public key to the DO and downloading of $(v_i + 1)$ no. of $m_{i,j}$
$= [2.163 + (v_i + 1).128]$-bit
$= (v_i.128 + 454)$-bit.

7 Security analysis

The proposed scheme is based on the hybrid cryptosystem, which includes robust secure cryptographic primitives AES, ECC, and one-way hash function. It is assumed that the key stores of the DO, CS, and the U_v will not be compromised by the adversaries.

As the exported data to the CS is kept encrypted during the operation of the scheme, the adversaries (active or passive) cannot compromise the exported data. The CS cannot decrypt or modify any encrypted data because the CS does not have any secret key for decryption of data.

The U_v always receives any data in encrypted form. Any revoked U_v from the same security class cannot eavesdrop any updated data destined for the U_v as the data for each U_v is over-encrypted by each U_v's unique session key by the CS. So, any active or passive adversary cannot compromise any data of the U_v.

7.1 Security analysis using AVISPA tool

In this section, first a brief discussion of the AVISPA tool [12, 13, 14] has been provided. After that, a brief description of the proposed scheme's HLPSL code (AVISPA's high level language code), which has been run on the AVISPA tool is described. Subsequently, the simulation results of the proposed scheme has been expalined.

7.1.1 Brief description of the AVISPA tool -

A well-known automated formal verification tool AVISPA (Automated Validation of Internet Security Protocol and Applications) [12, 13, 14] is being used for security verification of any proposed security protocol. The main advantage of the AVISPA is ability to use different verification techniques in different backends on the same protocol specification. The backends in the AVISPA framework are OFMC (On-the-Fly Model Checker), SATMC (SAT-based Model-Checker), CL-ATSE (Constraint-Logic based Attack Searcher), and TA4SP (Tree Automata based on Automatic Approximations for the Analysis of Security Protocols). The AVISPA uses Dolev-Yao intruder model [34] , where any intruder (active or passive adversary) can eavesdrop any transmitted message, mount masquerading (impersonation attacks), and replay attacks (modify or inject any message), but follows perfect cryptography, i.e. the intruder cannot break the cryptography. A high-level language HLPSL (High Level Protocol Specification Language) is a flexible, expressive, modular, and role based formal language, which is used on the AVISPA for specifying protocols, and their security properties. The HLPSL code consists of two parts - 'role' and 'goal' sections. The 'role' section of HLPSL code defines action of any entity of the proposed scheme, and is described by some states.

7.1.2 AVISPA in our proposed scheme -

The verification process of the AVISPA helps to check whether our proposed scheme is protected from any intruder attack, or not.

Our proposed scheme is defined in HLPSL code of the AVISPA framework, which contains 'role' sections of the DO, CS, and the U_v. Each 'role' section contains modified version of Needham-Schroeder [29, 13] authentication protocol, which is used for mutual authentication between the DO, CS, and the U_v. The 'goal' section shows the intended authentication verification between the DO and the CS, the CS and the U_v.

The modified version of Needham-Schroeder [29, 13] authentication protocol, which is used for mutual authentication between the DO and the CS has been discussed below -

1. DO $\rightarrow$ PKs[IDv, Na] $\rightarrow$ CS
(Encrypted with the public key PKs of the CS)
2. CS $\rightarrow$ PKv[IDs, Na, Nb] $\rightarrow$ DO
(Encrypted with the public key PKv of the DO)
3. DO $\rightarrow$ PKs[IDv, Nb] $\rightarrow$ CS
(Encrypted with the public key PKs of the CS)

where IDv and IDs are identification numbers of the DO, and the CS respectively. The numbers Na, and Nb are the 'nonce' generated by the DO, and the CS respectively.

The 'role' sections of the DO, CS and U_v are shown in Figure 4, Figure 5, and Figure 6 (end of the paper).

Figure 4 shows the 'role' played by the DO. The states 0 to 4 of Figure 4 shows Needham-Schroeder authentication protocol, and state 5 shows message transfer from the DO to the CS. The state 6 shows request message from the U_v.

Figure 5 shows the 'role' played by the CS. The states 0 to 1 of Figure 5 shows Needham-Schroeder authentication protocol, and state 3 shows message received by the CS, which was sent by the DO. The state 4 shows request message from the U_v. The state 5 shows requested message transfer from the CS to the U_v.

Figure 6 shows the 'role' played by the U_v. The states 0 to 3 of Figure 6 shows Needham-Schroeder authentication protocol, and state 4 shows requested message received from the CS.

The 'goal' section in Figure 7 shows required authentication between the DO and the CS, the DO and the U_v, the CS and the U_v.

7.1.3 Simulation result -

The simulation results of the OFMC, and SATMC backend independent components of AVISPA tool are shown in Figure 8 and Figure 9 respectively. The simulation results show that the proposed scheme is 'SAFE', i.e. it prevents attacks from the intruders, and the scheme is protected from the *replay* attack, as well as the *man-in-the-middle* attack.

The security analysis of the scheme, and the simulation results of the AVISPA tool show that the proposed cloud storage scheme is secured from the adversaries.

8 Comparison with other schemes

Here, the proposed scheme is compared with Wang et al. [9] scheme, and Chen et al. [4] scheme. The key derivation mechanism of the key management protocol, and the user revocation mechanism, which are incorporated in secure cloud storage scheme incur major costs in computational overhead of the cloud storage scheme.

The computational overhead of key derivation for Wang et al. [9] scheme is due to one-way hash operation [22], and symmetric key encryption operation. Here, the computational overhead for key derivation is $O(d)$, where d is the distance between predecessor, and succesor nodes. The scheme uses over-encryption technique for preventing access of data to revoked users. Any end user stores two secret keys for data decryption.

The computational overhead of key derivation for Chen et al. [4] scheme is also due to one-way hash operation [22], and symmetric key encryption operation. Here, also the computational overhead for key derivation is $O(d)$, where d is the distance between predecessor, and succesor nodes. The scheme uses proxy re-encryption technique for preventing access of data to revoked users. Any end user stores one secret key for data decryption.

The computational overhead of key derivation for our proposed scheme is $O(1)$, i.e. derivation takes constant time. Here, the predecessor node directly derives the key of the successor node. The scheme uses over-encryption technique for preventing access of data to revoked users, and the end user stores two secret keys for data decryption. The proposed scheme is suitable for end users to use resource-constrained wireless mobile devices as the scheme requires less computational, and storage overheads to access data.

Comparison of various schemes with our proposed scheme has been shown in Table - IV below.

Table 4: Comparison of various schemes with our proposed scheme

	Wang et al. [9]	Chen et al. [4]	Proposed scheme
Computational cost (Key derivation)	$O(d)$	$O(d)$	$O(1)$
Revocation mechanism	Over-encryption	Proxy Re-encryption	Over-encryption
Applicable to lightweight Mobile device (Yes/No)	Yes	Yes	Yes
Number of decryption keys required	2	1	2

9 Conclusion

In the proposed cloud storage scheme, it has been shown that the data requested by the end users are served to them using over-encryption by the cloud storage provider after proper authentication from the data owner, where the data owner already had sent the data to the cloud storage provider in encrypted form. All the required secret keys are generated, and distributed by the data owner to the corresponding users. It has been shown that the incorporation of the hybrid cryptosystem based key management scheme with hierarchical access control enhances the efficiency of the proposed cloud storage scheme. The hybrid cryptosystem uses secure lightweight ECC, AES, and one-way hash primitives. The scheme is secured against the adversaries as validated through formal verification tool AVISPA, and the end users of the scheme may use wireless resource-constrained mobile devices.

References

[1] CSA (Cloud Security Alliance): Security Guidance for Critical Areas of Focus in Cloud Computing V2.1. (October 2009). http://www.cloudsecurityalliance.org.

[2] Mell P., Grance T.: The NIST Definition of Cloud Computing Version 15., Information Technology Laboratory, NIST (National Institute of Standards and Technology), (October 2009). http://csrc.nist.gov/groups/SNS/cloud-computing.

[3] Kamara S., Lauter K.: Cryptographic cloud storage. In: 14th international conference on Financial cryptograpy and Data security, pp. 136-149 (2010).

[4] Yi-Ruei Chen, Cheng-Kang Chu, Wen-Guey Tzeng, and Jianying Zhou : CloudHKA: A Cryptographic approach for hierarchical access control in cloud computing, Proc. of 11th international conference on Applied Cryptography and Network Security (ACNS'13), pp. 37-52, Springer-Verlag, Berlin, Heidelberg, (2013)

[5] Basu A., Sengupta I.: Improved Secure Dynamic Key Management Scheme with Access Control in User Hierarchy. In: 2nd IEEE conference on Digital Information and Communication Technology and it's Applications (DICTAP), IEEE Computer Society, pp. 220-225 (2012).

[6] Vimercati S., Foresti S., Jajodia S., Paraboschi S., Samarati P.: Over-encryption: management of access control evolution on outsourced data. In: International Conference on Very Large Databases, pp. 123-134 (September 2007).

[7] Kallahalla M., Riedel E., Swaminathan R., Wang Q., Fu K.: Plutus: Scalable secure file sharing on untrusted storage. In: 2nd Conference on File and Storage Technologies (FASTâ03), USENIX, Berkeley, CA, pp. 29-42 (2003).

[8] Vimercati S., Foresti S., Jajodia S., Paraboschi S., Samarati P.: A data outsourcing architecture combining cryptography and access control. In: ACM workshop on Computer Security Architecture, ACM, pp. 63-69 (November 2007).

[9] Wang W., Li Z., Owens R., Bhargava B.: Secure and efficient access to outsourced data. In: ACM workshop on Cloud Computing Security, ACM, pp. 55-66 (2009).

[10] Yu, S., Wang, C., Ren, K., Lou, W.: Achieving secure, scalable, and fine-grained data access control in cloud computing, Proc. of the IEEE International Conference on Computer Communications (INFOCOM), IEEE Computer Society, pp. 534-542 2010)

[11] Sahai, A., Seyalioglu, H., Waters, B. : Dynamic credentials and ciphertext delegation for attribute-based encryption, Proc. of Safavi-Naini, R.(ed.) CRYPTO 2012, LNCS, vol. 7417, pp. 199-217, Springer, Heidelberg, (2012)

[12] AVISPA Project. AVISPA protocol library, http://www.avispa-proect.org.

[13] Armando A., Basin D., Boichut Y., Chevalier Y., Compagna L., Cuellar J., Drielsma P.H., He'm P.C., Kouchnarenko O., Mantovani J., M'dersheim S., Oheimb D. von ao, Michael R., Santiago J., Turuani M., Vigan L., Vigneron L.: The AVISPA tool for the automated validation of internet security protocols and applications. In: CAV 2005, pp. 281-285 (2005).

[14] Vigano L.: Automated Security Protocol Analysis With the AVISPA Tool. In: 21st Mathematical Foundations of Programming Semantics (MFPS'05), pp. 61-86 (May 2006).

[15] S. Akl, P. Taylor: Cryptographic solution to a multilevel security problem. In: Advances in Cryptology, Crypto â82, pp. 237-249 (1982).

[16] Akl S., Taylor P.: Cryptographic solution to a problem of access control in a hierarchy, ACM Transaction on Computer Systems, ACM, 1(3), 239-248 (1983).

[17] Chang C., Hwang R., Wu T.: Cryptographic key assignment scheme for access control in a hierarchy, Information Systems, Elsevier, 17(3), 243-247 (1992).

[18] Shen V., Chen T.: A novel key management scheme based on discrete logarithms and polynomial interpolations, Computers & Security, Elsevier, 21(2), pp. 164-171 (2002).

[19] Chang C., Lin I., Tsai H., Wang H.: A key assignment scheme for controlling access in partially ordered user hierarchies. In: 18th IEEE International Conference on Advanced Information Networking and Applications (AINA 2004), IEEE Computer Society, Fukuoka, Japan, pp. 376-379 (March 2004).

[20] Jeng F., Wang C.: An efficient key-management scheme for hierarchical access control based on elliptic curve cryptosystem, Journal of Systems and Software, Elsevier, 79(8), 1161-1167 (2006).

[21] Chung Y., Lee H., Lai F., Chen T.: Access control in user hierarchy based on elliptic curve cryptosystem, Information Sciences, Elsevier, 178(8), 230-243 (2008).

[22] Atallah M., Blanton M., Fazio N., Frikken K.: Dynamic and efficient key management for access hierarchies, ACM Transactions on Information and System Security, ACM, 12(3), 1-43 (2009).

[23] Nikooghadam M., Zakerolhosseini A., Moghaddam M.: Efficient utilization of elliptic curve cryptosystem for hierarchical access control, Journal of Systems and Software, Elsevier, 83(10), 1917-1929 (2010).

[24] Shuhua W., Kefei C.: An Efficient Key-Management Scheme for Hierarchical Access Control in E-Medicine System, Journal of Medical Systems, Springer, DOI: 10.1007/s10916-011-9700-7, Published online on 2011, 36(4), 2325-2337 (2012).

[25] Hankerson D., Menezes A., Vanstone S.: Guide to Elliptic Curve Cryptography, Springer (2004).

[26] Certicom Research: SEC 1: Elliptic Curve Cryptography, Standards for Efficient Cryptography 1, working Draft, Version 1.9, (August 22 2008).

[27] Vanstone S.: Elliptic curve cryptosystem - The Answer to Strong, Fast Public-key Cryptography for Securing Constrained Environments, Information Security Technical Report, Elsevier, 12(2),78-87 (1997).

[28] Stallings W.: Cryptography and Network Security : Principles and Practices, Fourth Edition, Pearson Prentice Hall (2006).

[29] Needham R., Schroeder M.: Using Encryption for Authentication in Large Networks of Computers, Communications of the ACM, ACM New York, USA, 21(12), 993-999 (1978).

[30] Xie M., Wang H., Yin J., Meng X.: Integrity auditing of outsourced data. In: International Conference on Very large Data Bases (VLDB 2007), ACM, pp. 782-793 (2007).

[31] Goodrich M., Papamanthou C., Tamassia R., Triandopoulos N.: Athos: Efficient authentication of outsourced file systems. In: International conference on Information Security, LNCS, Springer-Verlag Berlin Heidelberg, pp. 80-96 (2008).

[32] Bowers K., Juels A., Oprea A.: HAIL: A High-availability and Integrity Layer for Cloud Storage. In: 16th ACM Conference on Computer and Communications Security, ACM, pp. 187-198 (2009).

[33] Wong D., Fuentes H., Chan A.: The performance measurement of cryptographic primitives on palm devices. In: 17th Annual Computer Security Applications Conference (ACSAC 2001), IEEE Computer Society Washington, DC, USA, pp. 92-101 (2001).

[34] Dolev D., Chi-Chih Yao A. : On the security of public key protocols. In: Proc. of FOCS, IEEE Computer Society, pp. 350-357 (1981).

```
% HLPSL code
% DO authenticates CS
% CS authenticates DO
% DO sends encrypted message M, i.e. ({M}_Kv}) to CS where Kv is the symmetric key of DO
    known to DO & requesting user
% CS receives encrypted message M, i.e.  ({M}_Kv}) from DO
% Requesting user authenticates to DO and requests message to DO
% CS sends over-encrypted ({{M}_Kv}}_Kus}) message to user where Kus is the session key of
    CS & user.
% User receives over-encrypted ({{M}_Kv}}_Kus} message from CS
%  DO's role

role do (V, S   : agent,
        Kv : symmetric_key,
        PKu, PKv, PKs : public_key,
        IDv, IDs, IDu, RD : text,
        M: message,
        Snd, Rcv: channel(dy))

played_by V def=
  local State   : nat,
            U : agent,
            Na, Nb, Nc : text
            const  do_cs,  cs_do, do_user, skv : protocol_id

   init  State := 0

  transition
   1.  State    = 0  /\  Rcv(start) =|>
       State'   := 1  /\  Na' := new()
                      /\  Snd(IDv.IDs.{IDv.Na'}_PKs)        % Na' sent to S
                      /\  secret(Na', {IDv, IDs}, skv)
                      /\  witness(IDv, IDs, do_cs, Na')

   2.  State   = 2   /\  Rcv({Na'.Nb'.IDs}_PKv) =|>        % Nb'  received from V
       State'  := 3   /\  Snd({Nb'}_PKs)
                      /\  request(IDv, IDs, Nb')

    3. State   = 4   /\  Snd(IDv.IDs.Na'.{M}_Kv) =|>       % Encry. M sent to S
       State' := 5   /\  secret(Kv, {IDv, IDs})

    4. State   =  6   /\  Rcv(IDu.IDv.{IDu.RD.Nc}_PKv) =|>   % Request received from U
       State' :=  7   /\  secret(Nc', {IDv, IDu}, skv)
                      /\  witness(IDv, IDu, do_user, Nc')

end role
```

Figure 4: Role section of the DO

```
%HLPSL code – CS's role

role cs (S,V : agent,
        Kv, Kus : symmetric_key,
        PKs, PKv : public_key,
        IDu, IDv, IDs : text,
        M: message,
        Snd, Rcv: channel(dy))

played_by S  def=

      local State  : nat,
      U : agent,
      Na, Nb, Nd : text
      const  do_cs,  cs_do, do_user, cs_user, skv : protocol_id
      init  State := 0

  transition
   1.  State    =  0  /\  Rcv(IDv.IDs.{IDv.Na'}_PKs) =|>     % Na' received from V
      State'   :=  1  /\  Nb' := new()
                      /\  Snd({Na'.Nb'.IDs}_PKv)              % Nb' sent to V
                      /\  Rcv({Nb'}_PKs)                      % Nb' received from V
                      /\  secret(Nb', {IDv, IDs}, skv)
                      /\  witness(IDv, IDs, cs_do, Nb')

   2.  State   =  2   /\  Rcv(IDv.IDs.{M}_Kv) =|>             % Encry. M received from V
       State' :=  3   /\  secret(Kv, IDv)

   3.  State   = 4    /\  Snd(IDu.IDs.{IDu.Nd'}_PKs) =|>      % Request received from U
       State' := 5    /\  secret(Nd', {IDs, IDu}, skv)
                      /\  witness(IDs, IDu, cs_user, Nd')

   4.  State   =  6   /\   Snd(IDs.IDu.Nd'.{{M}_Kv}_Kus)  =|>   % Encry. M sent to U
       State' :=  7   /\   request(IDs, IDu, cs_user, Nd')

end role
```

Figure 5: Role section of the CS

```
% HLPSL code – Users's role

role user (U,V : agent,
           Kv, Kus : symmetric_key,
           PKu, PKv, PKs : public_key,
           IDv, IDu, IDs, RD : text,
           M : message,
           Snd, Rcv: channel(dy))

played_by U def=
   local State : nat,
   Na, Nc, Nd : text
   const do_cs, cs_do, do_user, skv : protocol_id
   init State := 0

transition
      1.  State   =  0  /\  Rcv(start) =|>
          State' :=  1  /\  Nc' := new()
                          /\  Snd(IDu.IDv.{IDu.RD.Nc'}_PKv)    % Request sent to V

      2. State   = 2  /\  Rcv(start) =|> % Request sent to s
         State' := 3  /\  Nd' := new()
                      /\  Snd(IDu.IDs.{IDu.Nd'}_PKs)            % Request sent to s

      3. State  =  4  /\  Rcv(IDs.IDu.Nd'.{{M}_Kv}_Kus) =|>    % Encry. M received from S
         State' := 5  /\  secret(Nd', {IDs, IDu}, skv)

end role
```

Figure 6: Role section of the end user

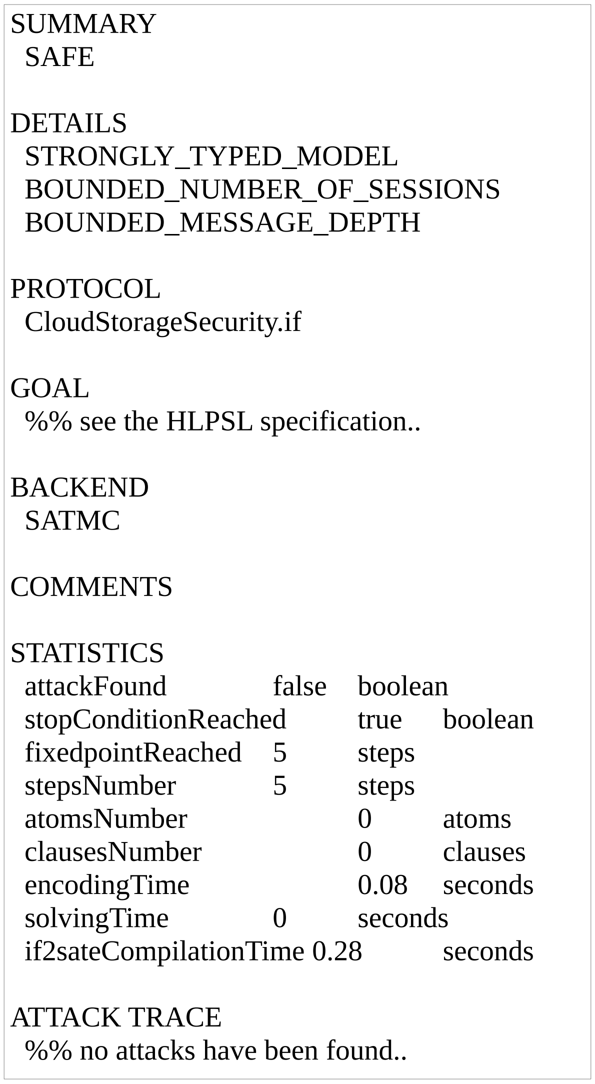
```
% HLPSL code  - goal section

goal
   secrecy_of skv
   authentication_on do_cs
   authentication_on cs_do
   authentication_on do_user
   authentication_on cs_user
end goal
```

Figure 7: Goal section

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /users/people/atanub/AVISPA/span/testsuite/results/CloudStorageSecurity.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 3.93s
  visitedNodes: 3349 nodes
  depth: 9 plies
```

Figure 8: OFMC result



```
SUMMARY
  SAFE

DETAILS
  STRONGLY_TYPED_MODEL
  BOUNDED_NUMBER_OF_SESSIONS
  BOUNDED_MESSAGE_DEPTH

PROTOCOL
  CloudStorageSecurity.if

GOAL
  %% see the HLPSL specification..

BACKEND
  SATMC

COMMENTS

STATISTICS
  attackFound             false    boolean
  stopConditionReached             true     boolean
  fixedpointReached       5        steps
  stepsNumber             5        steps
  atomsNumber                      0        atoms
  clausesNumber                    0        clauses
  encodingTime                     0.08     seconds
  solvingTime             0        seconds
  if2sateCompilationTime 0.28               seconds

ATTACK TRACE
  %% no attacks have been found..
```

Figure 9: SATMC result