ISSN:0975-9646

Ultimate Cloud

(a scheme which uses all formidable technologies culminated together to form a most secure mechanism for cloud data fortification)

K. Naga Sumanth, (M.Tech)
Department of Information Security and Cyber Forensics,
SRM University, Kattankulathur,
Chennai, India.

Dr. Magesh Sriramula, (Ph.d)
Department of Information Security and Cyber Forensics,
SRM University, Kattankulathur,
Chennai, India.

Abstract: - One of the supreme perplexing problems of cloud service provisioning is to motivate users to believe and to trust the security of cloud service and upstream their sensitive data. Even though cloud service providers can say that their services are heavily protected by extravagant encryption methodologies, modern cloud systems still cannot convince the users that even if the cloud servers are exploited, the data is still strongly protected. This study proposes Ultimate Cloud, a scheme which uses all formidable technologies culminated together to form a most secure mechanism for cloud data fortification, in every way that is possible. Ultimate Cloud employs AES, RSA and indirectly encrypts users' data by their public keys, but stores their private keys on nothing; instead, they are stored on any mobile devices and accessible via 2-dimensional barcode images such as QR codes. And they are employed to decrypt users' sensitive data. In this way, users' data are securely protected even if the cloud servers are compromised. Also, Ultimate Cloud provides users with the knowledge of managing private keys by storing the keys into mobile devices and displaying them via QR codes. Moreover, three Modules exist such as: personal, home and enterprise level scenarios are proposed to present the achievability of Ultimate Cloud. In addition, a classified structure is designed for key randomization and data sharing in the suggested structure.

Key words: - UC or uC (Ultimate Cloud), QR (Quick Response)

I. INTRODUCTION

Benefit from the idea of cloud and related technologies, cloud services are popular in recent years. More and more users and enterprises tend to upload their data onto cloud servers so that the data can be maintained properly with the scalability, ubiquity and accessibility properties. However, since users usually do not know or trust the security level of external cloud services, risk management of out-sourcing data storage service is a dire issue here.

Conventionally, enterprises manage their trade secrets by themselves. Although uploading these trade secrets to cloud servers can reduce the cost of data management, the risk of secret leakage is greater. Business rivals, or even governments, may try to steal the trade secrets by employing hackers or sending insiders to the cloud service operator. Moreover, since the ownership of enterprise data usually belongs to the enterprises, not creators, the data must be accessible to the creator's direct or indirect bosses according to the enterprise hierarchy structure but not to other unauthorised employees.

The most common and trusted solution to protect data are to encrypt and decrypt data by encrypting and decrypting keys. However, the design of key management of these keys is the greatest challenge of this solution if both the data confidentiality and sharing requirements should be satisfied. Traditionally, the keys are designed to be stored on PCs where the data are encrypted and decrypted. In this manner, although the data can be safely protected, it is difficult to be shared with other users who are authorised to access it. Also, the keys are not portable and not convenient to be used on other PCs.

II. BACKGROUND WORK

A. Cloud data fortification

Commonly in conventional cloud service providers such as Amazon, google users' data are managed and secured by them. Suppose those cloud servers are compromised in one way or other, then the users data may suffer from leakage problem. Several researches include the idea of client application which is responsible for encrypting the data before sending to the cloud service provider and decrypting after downloading it from it. Also, the client application is coupled with user's computer here and the decryption keys of data are difficult to be shared with other users or another client application.

B. Facial Recognition

Recently there has been introduction of many alternatives for the authentication and authorization mechanisms such as biometric, palm-vein technology and so on. But the best and lightweight technology more over efficient one is the facial recognition. Now as far as we know each facial recognition software has an algorithm and they try to match using that. The one which we are currently available can be easily bypassed or in some cases due to the light variations they may not be able to even recognize the face in front of the camera. So, this factor is to be resolved and combined with any other authentication mechanism would provide tight security.

C. QR codes

Quick Response codes are normally used to scan the URLs' but on the bright side they can store large amount of data unlike the barcodes. The recognition speed for QR codes for an ordinary 0.3 Mega pixel camera is 0.35 seconds which is quite impressive. But then again every mobile device now a day have QR scanners so we do not want to store any sensitive info in the QR codes. Or we have to find

a mechanism to randomize the QR code for a single user in a period of time, such that if any one manages to acquire it, it is useless for him.

III. SYSTEM OVERVIEW AND ARCHITECHTURE

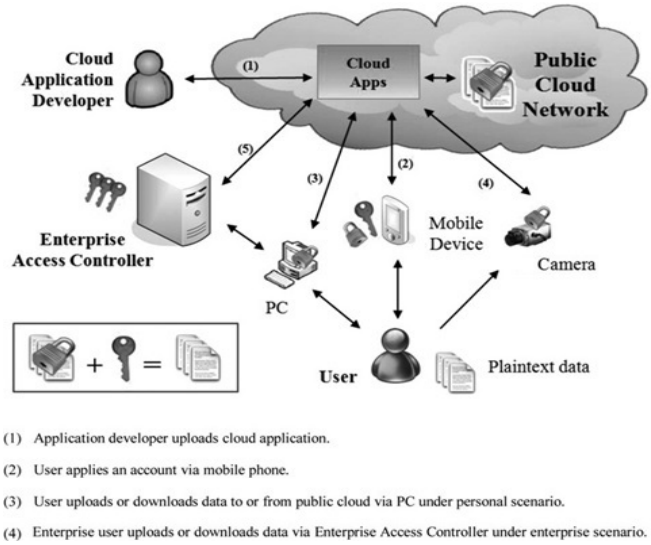

Fig 1: Basic Overview

Fig. 1 shows the system overview of UC. In this figure, the key and lock represent a pair of private and public key, respectively. First of all, the cloud application developers can upload and register their applications onto an un-trusted public cloud. Second, the users can generate their own public and private key pairs by mobile applications, and register their accounts of the public cloud applications. For the personal usage scenario, users can encrypt and upload their files by PC applications and decrypt these files by showing 2D barcode images, which include the users' private keys, on mobile devices to PC applications. For the home surveillance scenario, IP cameras can encrypt streaming frames indirectly with the users' public keys. The encrypted streaming frames can be decrypted and displayed by PC applications after the private keys are extracted from 2D barcodes. Finally, for the enterprise scenario, the Enterprise Access Controller (EAC) provides the access control service to decide whether a user can access another user's secret files.

Fig. 2 shows the high level architecture of UC public cloud. Similar to traditional cloud service, cloud applications of UC are deployed upon cloud platforms (such as the Hadoop platform) in application servers. Based on this architecture, the UC module is included as a middleware between applications and cloud platform to manage security issues.

In the UC module, whenever a cloud application is registered by application developer, the Application Registration Manager registers the application, and returns the ID and password of this application. If any cloud application utilises any functionality of UC module, its ID and password must be verified by the application authenticator. Although the applications can maintain their own user accounts, the User Registration Manager manages a global identifier space where each account in each application is mapped to a unique global UC identifier. Finally, the Relationship Manager maintains the relationship or enterprise architecture between different users.

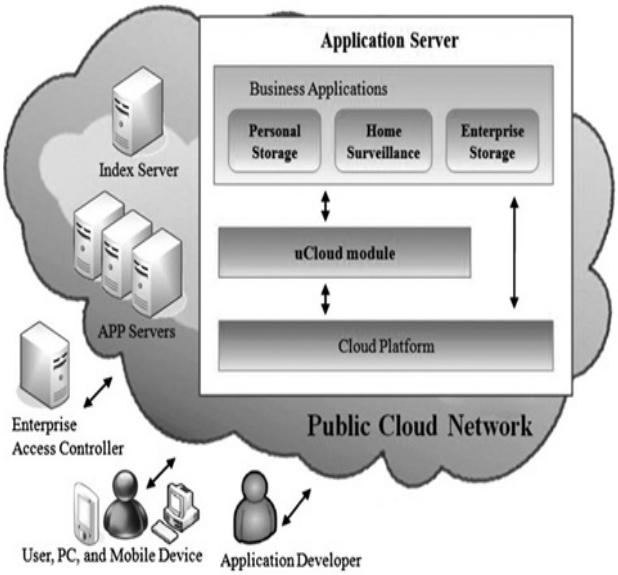

Fig 2: High level Architecture

Whenever an enterprise user sends a request to access other user's data, the EAC must decide that whether this request is permitted via the decision maker module based on access control rules. If this request is permitted, the data manager module is executed to generate the encrypted owner's private key. All the private keys of enterprise users must be stored in the key storage of EAC to fulfil these requests and the backup requirement.

IV. SYSTEM ASSUMPTIONS

To define the scope of uC, several assumptions are defined. Four main parties are included in uC: mobile phone, PC application, enterprise access controller and public cloud. The mobile phone keeps the user's root secret (private key), so it is the root of trust of uC and is not compromisable. One user account can be associated with only one mobile phone. The PC and PC application are not compromisable; if somehow they are to be compromised, we are implementing pseudo random QR generation on both mobile device and cloud server which acts as a token key for a certain period of time and so prevention access to attackers to the files (This is just authentication concern). Although PC is not compromised and private keys can be safely stored on PCs, multiple copies of keys must be maintained by users if the data are required to be decrypted on multiple PCs; it is the usability concern rather than security concern to store keys on mobile phones rather than PCs. In addition, we assume that the public cloud is compromisable but the enterprise access controller is not. Since the enterprise access controller only manages the private keys of enterprise users, access control can be provided without maintaining the large files. Therefore users can feel safe to upload their sensitive data to public cloud. Also, the public cloud service provider must already have a public-key infrastructure (PKI), ⟨EKuC, DKuC⟩, to

use, and the public key EKuC is included directly into the PC and mobile phone applications and maintained in the enterprise access controller. Since numerous researches have proposed various revocation mechanisms for PKI, such as creating certificates via CA, this paper does not focus on this issue. In addition, the mobile and PC applications must not be modified or replaced by adversaries before they are downloaded and installed by users; otherwise, they can be controlled by attackers. Moreover, users can delegate the access rights of sensitive files to the delegates who they trust. Finally, we encrypt the user files using AES or Rijndael algorithm and only use RSA to encrypt and decrypt Session keys.

V. DETAILED SYSTEM DESIGN

The design of uC includes three scenarios: the personal usage, home surveillance and enterprise scenarios. To describe the protocols of them, numerous notations are defined in Table 1 and employed throughout this paper.

To register an account, the user-selected account UIDAPP and π are encrypted with a random nonce R1 by the uC's public key EKuC. If this registration request is approved by cloud application, the cloud application logins to uC module by providing APPIDuC and APPKEYuC to register the account. The registration result is sent back to user with a signature sig1 of uC module, where sig1 = Da (result||UIDAPP||π, DKuC). Then, the mobile application can generate a pair of public and private keys, encode the public key into a QR code image and send the public key to PC application. Finally, the PC application forwards the user's public key to the uC module.

Table 1: Notation Description

UID_{APP}, π	The user account and password used to login the cloud application
UID_{uC}	The global user identifier registered by cloud application and maintained by uC
$APPID_{uC}$, $APPKEY_{uC}$	The account and password used by cloud applications to login the uC module
Rx	Random numbers (can be used as session keys) (x = 1, 2, 3, …)
EKuC, DKuC	The public and private keys of uC module
EKEAC, DKEAC	The public and private keys of Enterprise Access Controller
EKuser, DKuser	The user's public and private keys
EKcam, DKcam	The public and private keys of IP camera
Ea(d, k), Da(d, k)	Asymmetric encryption and decryption functions for data d and key k
Es(d, k), Ds(d, k)	Symmetric encryption and decryption functions for data d and key k (Es = Ds)
Me(d), Md(d)	Matrix code encoding and decoding functions for data d
Path UID 1APP, UID2APP	The hierarchical path from UID1APP to UID2APP

A. Personal Usage Scenario

The personal usage scenario is designed for users to upload and backup their files on one PC or laptop, and download them to other trusted computers. In this scenario, regular files are protected by using the PC applications on users' computers and a personal storage cloud application. Fig. 3 shows the sequence diagram to encrypt and upload files. First, users enter their accounts and passwords to login the Personal Storage Application, and choose a random number R2 to be the file encryption session key. The uploaded file is encrypted by this session key by symmetric encryption function, and this key is encrypted by user's private key; therefore users can extract it by using their private keys in the future.

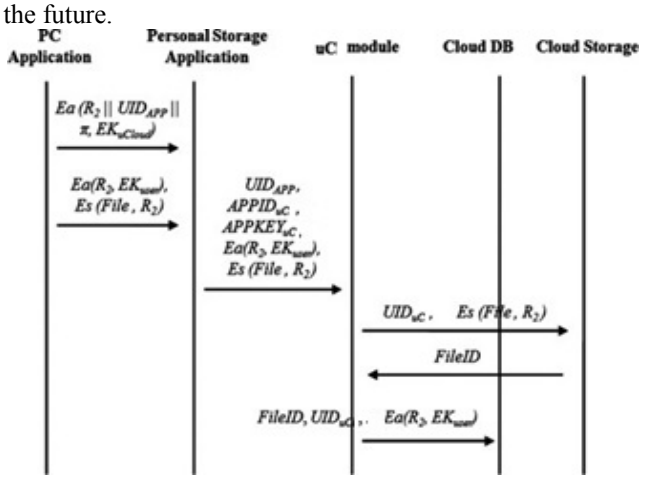

Fig 3: Sequence diagram to encrypt and upload files.

Fig. 4 shows the sequence diagram of downloading the previously uploaded files. After logins to Personal Storage Application, the PC application sends a request to retrieve the files and related metadata by providing the file names. Then, uC module sends back the encrypted file and session key together with the signature sig2 = Da(h(Es (File, R2)||Ea(R2, EKuser)), DKuC), where h is a hash function for reducing the size of signed message. After the encrypted file and metadata is downloaded, the user is asked to present a QR code containing user's private key. Therefore the PC application is able to decrypt the session key R2, and use it to decrypt the file further.

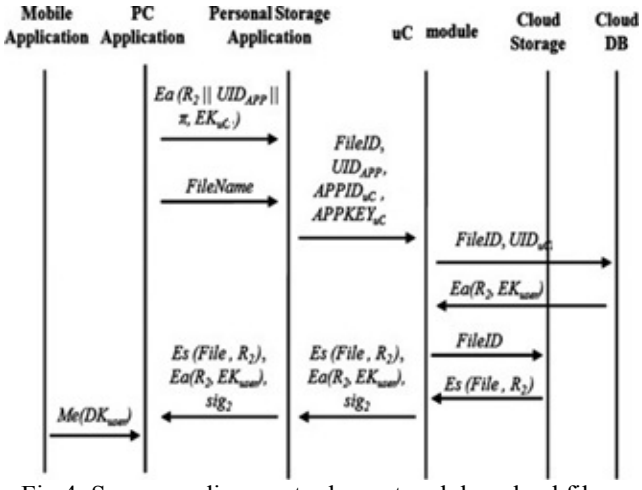

Fig 4: Sequence diagram to decrypt and download files.

B. Enterprise Scenario

For enterprise users, uC module provides a hierarchical key management scheme for basic data sharing. In general, a management hierarchy is established in enterprise with multiple levels. For example, Fig. 5 shows a hierarchy containing chairman, CEO, customer service and RD staffs. To protect their data, a pair of public and private keys is selected by each one of themselves. However, enterprise data are not personal property; at least, the owner's boss must also be able to access the owner's data. Therefore a copy of the owner's private key encrypted by the public key of the owner's direct boss must be maintained. Moreover, users can also delegate their keys to other users who they trust in case they are not familiar with computers or they want to back up their keys. In this case, the data owner's private key is also encrypted by the public key of the trusted person and maintained in uC.

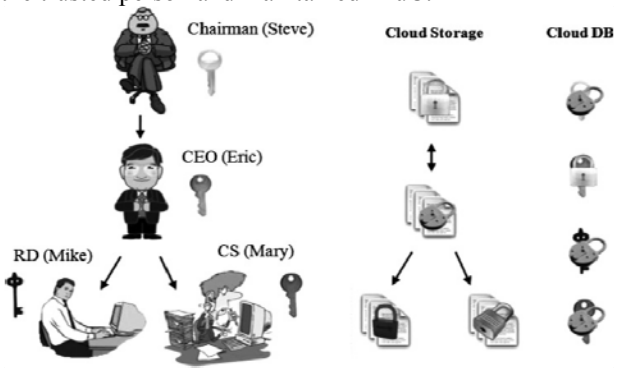

Fig 5: General enterprise scenario

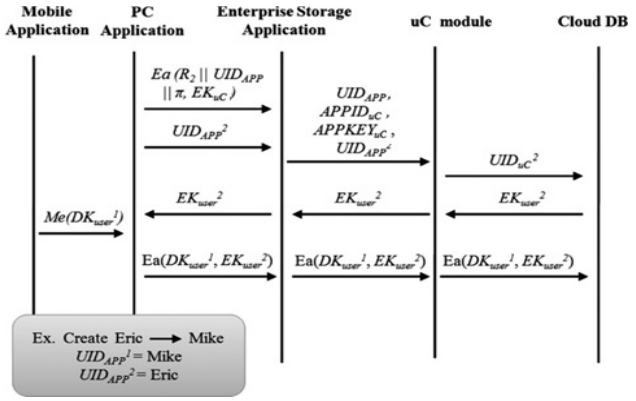

Fig: 6 hirrerarcy path construction

Fig. 6 shows an example of hierarchy path construction. In this case, the RD Mike wants to specify that the CEO Eric is his direct doss. To achieve this goal, Mike logins to the Enterprise Storage Application first to obtain the Eric's public key. Then, the Mike's private key is provided by the mobile application, encrypted by Eric's public key and uploaded to uC module.

Fig. 7 shows an example that the data owner's indirect boss wants to obtain the owner's data. In this case, the chairman Steve wants to access the RD Mike's data. According to the file ID, uC module finds out the file's owner and the encrypted session key Ea R 2, EK_{user}^3 of this file. Then, any path from Steve to Mike is searched by depth-first search. If any path exists from Steve to Mike, the encrypted private keys of all users along with this path are gathered. The path, encrypted private keys, encrypted session key, encrypted file and signature sig_4, are sent back to the PC application together, where

sig_4 = Da (h(Path UID^1_{APP}, UID^3_{APP}Ea $DKuser^2$, EK_{user}^1
3 2 3
Ea DK user, EK_{user}⋯Ea R 2, EK_{user} EsFile, R_2, DK_{uC}

After the private key DK_{user}^1 is obtained, the DK_{user}^2 , DK_{user}^3 , R_2 and the file can be decrypted sequentially.

Although uC module provides the hierarchical data sharing mechanism, more complex access control mechanism such as sharing data between users belonging to the same group but not superiors to each other is not supported. To solve this problem, enterprises can maintain their access control servers, EACs, to cooperate with uC module. Fig. 8 shows the sequence diagram of file sharing via EAC. In this case, the RD Mike wants to obtain the CS Mary's file. First, Mike encrypts his account and password by EAC's public key, sends the result to EAC, and requests for obtaining Mary's file. If this request is permitted, the EAC uses Mike's account and password to login the Enterprise Storage Application, and the Enterprise Storage Application logins to uC module. According to the file ID, uC module gathers the encrypted session key Ea R 2, EK_{user}^2 and the encrypted file, and sends the results and sig_5 to EAC, where sig_5 = Da(h)

UID^2_{APP}Ea R 2, EK_{user}^2 EsFile, R_2DK_{uCloud}. Since EAC maintains Mary's private key, DK^2_{user}, this key is encrypted by Mike's public key and sent to the PC application. Finally, Mike can decrypt the file by using his private key, the DKuser2 , and R2.

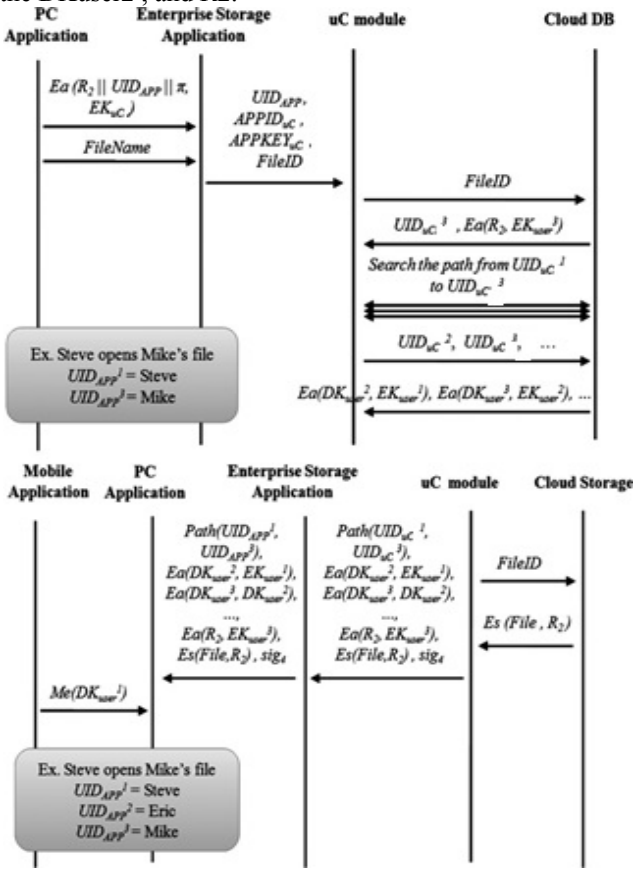

Fig: 7 Sequence diagram of file sharing

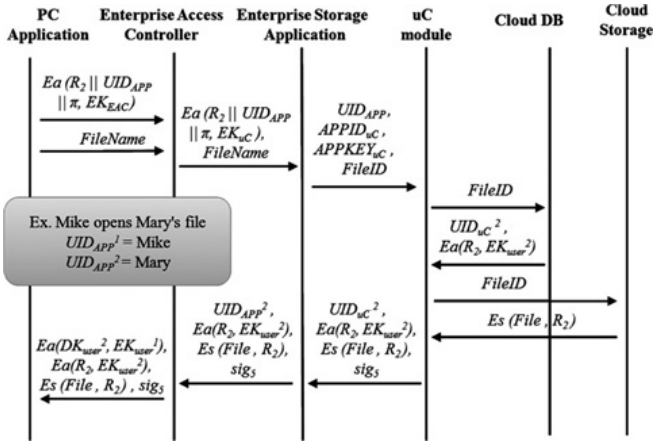

Fig: 8 Sequence Diagram of File Sharing via EAC

C. Security Analysis

As the assumptions mentioned previously, the cloud service, including cloud applications, uC module, cloud storage and cloud DB, are compromisable. However, since user's data are properly encrypted by user's private key which is not stored on cloud, adversaries cannot access the plaintext of data. Although attackers can only delete the encrypted data and session keys, this problem can be solved by self-maintained backups; it is out of scope of this paper. Also, if the hierarchy maintained by uC module is modified, the protected data are still secure since the session keys are only encrypted by the public keys of other trusted users. Moreover, since the mobile device is the root of trust, it is not compromisable. If user's mobile phone is stolen, users can login to the cloud application and delete their data as soon as possible, since users usually bring their phones with them and use them frequently. Therefore only if the mobile phone is stolen and the cloud service is compromised, the protected data are leaked. Finally, although the delegatees have the private keys of delegators, they cannot access to all the files of delegator without the grants of delegator. The reason is that the delegatee and public cloud operator do not collude (as defined in assumptions); therefore the access control and file decryption operations are separated.

As the messages transferred between different parties of PC application, cloud service and EAC, since all the messages and data are properly the encrypted by public keys or session keys, attackers cannot extract protected information. Also, the channel established between mobile and PC application is a constrained channel; 2D barcode images are difficult to be eavesdropped. Finally, almost all the returned messages are properly signed, so attackers cannot forge or modify them without having the private keys.

VI. SYSTEM PROTOTYPE

To demonstrate the proposed system, the uC module, the personal storage and enterprise storage scenarios are implemented in the uC prototype. In the uC prototype, the mobile application is deployed on HTC x using Android Programming. In the PC application, the data protector and Face recognition, QR creation and Scanning is implemented in C# the EAC client is implemented in Java. In the EAC, the user authenticator and decision maker are implemented in JSP and deployed on Tomcat server, and the key storage and access control rules are stored in MySQL server. Finally, a User Interface with High level Security will be made available for the people to use.

VII. CONCLUSION

In conclusion, we propose the uC to provide better management of cloud data fortification, which includes a hierarchical structure for key backup and data sharing, and the EAC server to extent the capability of difficult access control. In uC, the private keys are stored on users' mobile devices and presented via QR barcode imageries when they are utilised to decrypt users' data. In this routine, even if the cloud services are compromised, the data are still safely secured. Therefore individual, family and enterprise users can feel save to upload their sensitive data up to cloud. Finally, the uC prototype including the uC module, these two scenarios, and EAC, is implemented and evaluated to show that it is convenient to be used.

ACKNOWLEDGEMENT

This Project was supported by the Department of Information Security and the Forensics division and prototype tests are conducted in Cloud Central Laboratory and Information security Laboratory and of SRM University, Chennai.

REFERENCES

[1] Behl, A., & Behl, K. (2012). An Analysis of Cloud Computing Security Issues. IEEE, 109-114.
[2] Thanks to the highly useful informations regarding QR codes in the website https://qrstuff.com/.
[3] Top 5 security risks detailed in the webpage http://blogs.cisco.com/smallbusiness/the-top-5-security-risks-of-cloud-computing/.
[4] Face recognition with learning base descriptor (IEEE) paper proposed by Zhimin Cao, Qi Yin, Xiaoou Tang, Jian Sun.
[5] Architechtural cloud design patterns provided by 'Amazon web services'.
[6] Methodology of network security design proposed by Donald Graji,Mohnish Pabrai ,Uday Pahrai .
[7] Integrated design of AES (Advanced Encryption Standard) encrypter and decrypter proposed i Application-Specific Systems, Architectures and Processors, 2002. Proceedings. The IEEE International Conference.
[8] Wikipedia.