Abstract

Wireless Sensor Nodes in military situations, for example, a war zone or an unfriendly locale are liable to experience the ill effects of discontinuous system network. **Objective:** To design an efficient data recovery scheme for Decentralized DTNs in military environments, based on CP-ABE methodology, making Security the utmost prioritized characteristic. **Analysis:** Latest Innovations in Disruption Tolerant Networks (DTN) are getting to be effective arrangements, that permit mobile devices carried by military personnel to correspond with one another and access the classified data or to command effectively by manipulating the independent-autonomous repository nodes. Out of these, the most difficult issues in the situation are the authorization strategies and those approaches’ improvisation for secure data recovery. **Methodology:** Cipher text Policy-Attribute Based Encryption (CP-ABE) is a promising cryptographic technique to address the access authority issues. The issue of applying CP-ABE in decentralized DTNs presents a few security and protection challenges like attribute revoke, key security, and regulation of attributes issued from distinct authorities. So, we implement CP-ABE along with 2PC (Two Phase Commit) protocol and where numerous key authorities deal with their attributes autonomously to overcome the above mentioned challenges. **Findings/Improvements:** We implement the proposed System as a stand-alone application in Eclipse IDE. We’ve been successful in implementing this system to achieve the security requirements we’ve been discussing at length in the paper and we’ve demonstrated how to recover the data in very hostile military environments in an efficient and a secured manner.

**Keywords:** Attribute Based Encryption, Collusion Protection, Data Confidentiality, Decentralized Key Authorities, Disruption Tolerant Networks, Forward and Backward Secrecy, Military Networks, Privileged Access Permissions, Secured Data Sharing, Wireless Sensor Networks

1. Introduction

Disruption Tolerant Network (DTN) is a network architecture that deals with the issues in heterogeneous networks that are defective in consistent availability of the network, which operates in very hostile and unfriendly environments. DTNs are described by their absence of connectivity, bringing about an absence of prompt end-to-end routing paths. In these tricky situations, mainstream ad hoc routing conventions, for example, AODV and DSR fail to set up courses. This is because of these conventions attempting to first build up a complete routing path and after that, only after the course has been set up, the actual data is forwarded. Then again, when spontaneous end-to-end routing paths are troublesome or difficult to set up, routing conventions must adapt a “store and forward” methodology, where information is incrementally moved and put all through the network with a hope that it will ultimately reach its destination. A typical strategy used to amplify the likelihood of a message being effectively communicated is to repeat numerous duplicates of the message with the hope that at least one of them will end up reaching its destination.

Consigning the security issues has been one of the major focuses in DTNs. Security establishment
in DTN range from reckoning upon the back ground circumstances and application, however authentication and privacy are fundamentally critical. These security assurances are hard to build up in a network which is without consistent availability, on the grounds that the network prevents sophisticated cryptographic protocols, obstructs key exchange and every mobile device must recognize other irregularly visible devices. Solutions have regularly been adjusted from versatile ad hoc networks and distinguished security analysis, for example, the utilization of decentralized certificate authorities. Unique solutions from the Delay Tolerant Networking Research Group\(^2\) include: Identity based encryption\(^3\)\(^4\) and usage of Gossip protocol\(^5\).

Coming to Military DTN’s, the problem here is not just the lack of continuity in network but also the other external disturbances such as jamming, inconsistent mobility of the nodes, damage causing agents, eavesdropping by the opponents and many other adversaries.

Military applications require enhanced assurance of classified information including access control approaches that are cryptographically implemented. There are already developments for storage nodes in DTNs, where data is reserved and reproduced such that only the privileged and authorized nodes can access/share the required data amongst themselves in an effective and a rapid way\(^6\)\(^7\).

Much of the time, it is apt to give distinguished access approaches such that information access arrangements are characterized over client attributes or designations, which are controlled by the key authorities. Therefore, it is a Logical Presumption that numerous key authorities are likely to deal with their own particular attributes for troopers in their deployed areas or echelons, which could be very dynamic in nature. We advert this DTN schematic where numerous authorities issue, furthermore, deal with their own attribute keys autonomously as a decentralized DTN\(^8\).

The scheme of Attribute Based Encryption (ABE) is an assuring methodology that appeases the prerequisites for secure information recovery in DTNs\(^8\)\(^9\)\(^10\). Attribute Based Encryption is a kind of public-key encryption methodologies where in the secret key of the client and the encrypted (cipher) text looks upon to the attributes (e.g. Battalion of the soldier, the region he belongs to, or the kind of subscription he has with the military base). In such a framework, the decoding of a cipher text is conceivable if and only if the attributes of the client key matches the attributes of the cipher text\(^11\). An essential security part of Attribute-Based Encryption is to incorporate collusion resistance: An enemy that holds different keys ought to only have the scope to get to information if no less than one individual key grants access. ABE highlights a methodology that empowers an access control over encoded information utilizing access arrangements also, credited attributes among private keys and cipher texts. Particularly, Cipher text Policy-ABE (CP-ABE) gives an adaptable method for encoding information such that the encryptor characterizes the attribute set that the decryptor needs to maintain for decrypting the cipher text. Therefore, diverse clients are permitted to decode diverse bits of information as per the security methodology in force.

On the other hand, the issue of enforcing the ABE to DTNs presents a few security and protection problems. Since a few clients may change their related attributes sooner or later (for instance, transferred from one battalion to the other), or some private keys may be compromized by the attacker, key updating for every attribute is vital with a specific end goal to make frameworks secure. But, this issue is much more problematic, exceptionally in ABE frameworks, since every attribute is possibly shared by diversified clients (from this time forward, we call such an accumulation of clients as an attribute group). This suggests that revocation of any attribute or any particular client in a such a group would influence the remaining clients in the group. For instance, on the off chance that a client joins or leaves an attribute group, the related attribute key ought to be modified and redistributed to the various individuals in the same group for in forward or backward secrecy. It may bring about a hindrance amid rekeying method, or security corruption preceding attribute key is not redesigned quickly.

One more complication is the key security issue. In CP-ABE, the key authority produces private keys of clients by incorporating the authority’s master secret keys to clients’ related set of attributes. Subsequently, the key authority can decipher each cipher text intended to distinct set of users by producing their attribute keys. If at all the key authority is brought down by attackers when deployed in the unfriendly situations, this could be a potential risk to the data secrecy or protection particularly when the information is exceptionally confidential. The key security is an inborn issue indeed, even in the multi authority frameworks as long as every key authority has the license to create their own particular attribute keys with their own particular master keys. As such a key

---

\(^{11}\) An essential security
protection can be cryptographically upheld against any inquisitive key authorities or information storage nodes.

2. Security Requirements

In this section we'll look into the three crucial security aspects of CP-ABE policy, which our proposed scheme should be able to incorporate into it. Our system should be able to handle all of the three requirements. They are as follows:

2.1 Data Confidentiality

Unauthorized clientele who do not satisfy the predefined access policy methodology viz., not in possession of all the set of attributes that are required for access grant should be detained from accessing the data from the external mobile storage node. Apart from this, the same methodology should also be applied to the storage node and key authorities, that is, unauthorized access of both these entities should also be deterred.

2.2 Collusion-Attack Protection

There is a possibility if some clientele in the system decided to conspire against the defence mechanism by merging their attributes to decipher the data, even though each of these clients cannot do it individually. Let's have an insight to this scenario for better understanding. Assume a Soldier who had attributes (“Battalion 2”, “Region 3”) and another Soldier with attributes (“Battalion 3”, “Region 1”). There is a possibility that if these soldiers work together they can compromise the system’s security and successfully access the classified information under the access policy (“Battalion 2” AND “Region 1”), even though they cannot be doing this security breach individually. A robust system cannot let this collusion attacks happen.

2.3 Forward and Backward Secrecy

As far as CP-ABE is concerned there comes two type of secrecies needed to be definitely satisfied, no matter what. They are generally termed as Forward Secrecy and Backward Secrecy. Forward secrecy means once if any client annuls any of his respective attribute should be blocked from accessing the data transferred after the attribute nullification, unless and until his other valid attributes satisfy the access privileges. The data secrecy and protection can be cryptographically upheld against any inquisitive key authorities or information storage nodes.
client possessing an attribute that fulfils the access policy, should be blocked from accessing the data of the information transfer before he got possession of the attribute.

3. Proposed Methodology

We’re proposing a multi authority CP-ABE scheme to recover data from a decentralized DTN in a secured manner. Each and every one of the multiple local key authorities distributes partially customized attribute keys to the clients by executing a robust 2PC protocol with one another where every attribute key of the client can be updated instantaneously. Therefore, security and scalability of the system can be optimized.

3.1 Key Components

CP-ABE scheme maintains two components for client secret key. They are a Personal unique key and multiple attribute keys. The personal keys are unique to each client for avoiding the Collusion attacks, which were discussed in the previous section. In our proposed scheme, a personal key is produced succeeded by multiple attribute keys. We can overcome the key security problems like no single key authority can be able to access the complete key privileges of clients all by themselves.

3.2 Data Encryption and Decryption

In our scheme plain text is encrypted via Advanced Encryption Standard (AES) encoding scheme, which can be used at both the sender and receiver sides as it employs symmetric key algorithm, same key is used for both ciphering and deciphering the plain text. In our simulation, client also employs the same AES scheme to decrypt the encrypted data. AES is based on Rijndael cipher. Rijndael cipher comprises with a fixed block size of 128 and three variable key sizes viz., 128, 192 and 256. AES is variant of this fixed size with in Rijndael cipher thought it follows the minimum key length of 128 and maximum of 256. The key length employed for the AES scheme is based on the cycles of transformation of plain text into cipher text, input into output. The number of iterations are as follows:

- 128 bit keys: 10 iterations
- 192 bit keys: 12 iterations
- 256 bit keys: 14 iterations

This AES scheme can implemented with ease on both hardware and software, as it is designed on the strategy of amalgamating successive substitution and permutation called, substitution-permutation methodology.

3.3 Key Updating and Revocation

Key revocation is a tricky part in the implementation. To counter the Collusion attacks in the system, the normal scheme employed in key revocation is, Complete set of attribute keys of a client are needed to be revoked to facilitate the revoking of a particular attribute key of the client. This particular methodology becomes very troublesome for large DTN's as it causes high overhead in the communication.

We propose that instead of rekeying the entire set of attribute keys, we identify the particular attribute key that need to be revoked, we re-encrypt the cipher text with every attribute key and re issue the same to the authorized users who are in possess of the valid attribute key. Key updating activity must be carried out by the key authority each and every time a client drops his attribute to withhold the user from accessing the preceding or succeeding data for satisfying one of the three security requirements stated in the previous section: forward and backward secrecy.

This key updating mechanism is carried as follows: A join/leave request it sent for the attribute group from which the client wants to drop/hold the attribute. On successful reception of this request, the local key authority notifies the external storage node and this prevents the user from unauthorized access of the plain text at the storage node.

4. System Design

In this section, we’ll be dealing with System Design and Development.

4.1 System Architecture

Figure 1 depicts the System Architecture. The architecture comprises of the following modules

4.1.1 Sender

Sender is responsible for sending/communicating the information with his clients. In our case Sender can be any military authority say, like a commander. He possess classified information which he wishes to share only with trusted clients (for instance, soldiers in his battalion). The classified information possessed by the sender is stored in the external mobile storage node. Whatever the data the sender wishes to communicate with the clients, it is done
in very hostile environments and is expected to be done reliably. Before placing the data in the mobile storage node, sender describes the access policies (attribute-based) and implements these policies on the data so that only those clients who are in possession of such privileges can access the data.

### 4.1.2 Mobile Storage Node

This particular entity is responsible for storing data which is placed by the sender and facilitates the access to the corresponding clients. In our scheme, we consider this particular entity to be partially loyal, which is genuine-but-inquisitive.

### 4.1.3 Decentralized Key Distribution Authorities

These authorities generate public keys or secret keys for CP-ABE based on the parameters. We are talking about Decentralized key authorities, that is there is no one central key authority, there are multiple local authorities that generate secret keys based on the attributes and distributes those generated attribute keys to the individual clients (for instance, soldiers in a client). These decentralized key authorities allot distinguished access privileges to the clients, based on their attributes. We assume these authorities to be genuine-but-inquisitive. They will loyally perform the tasks designated to them, but would like to grasp the information of encrypted data to the maximum extent.

### 4.1.4 Clients

Clients are those who wish to access the information reposited in the mobile storage node, who in our case are soldiers of a battalion. If a soldier owns the attributes that fulfil the constraints of access policy that is in enforcement, satisfy the privileges set by the key authorities and is not annulled in any set of attributes, then such a client can access the data, decipher the encrypted data and attain the data.

### 4.1.5 CP-ABE Methodology

As already stated, our decentralized key authorities are in a state of “genuine-but-inquisitive” and are partially loyal, we should make sure that they are detained from attaining the actual data that sender wishes to deploy in the external mobile storage node by themselves, but still they must be capable of distributing secret keys to the clients. To rationalize this quite a bit tricky requirement, we are implementing 2PC protocol on the key authorities, who have their own master secret keys and can distribute autonomous keys to the clients. The 2PC protocol restricts these local key authorities from gaining access to each of the other local key authorities’ secret keys, therefore, none of them can achieve the ability to produce the complete set of user secret keys by themselves. Therefore, we can assume that these key authorities cannot access the secret keys of all other users and this leads to conclusion that keys authorities do not collude with one another.

### 4.2 Use Case Diagram

Figure 2 depicts the use case diagram we’re implementing in the system. There are three actors in this use case diagram, viz., Commander, Soldier and Key Authority. Each of these actors has specific set of tasks to achieve the CP-ABE based secured data recovery in our proposed system. First of all, both the Commander and
the Soldier must register with the external storage node in order to perform their respective roles in the system. Now Commander logs in to the storage node, encrypts the message/data that he wishes to communicate with his Soldiers (clients) and upload it to the external storage node. The Key distribution authority is decentralized here and there are multiple local key authorities, in which one of them generates the secret key for the file that the commander had uploaded (here the selection of local key authority is done by `r and() method). During the uploading process, the required access policy (attribute based) is enforced onto the key authority by the commander.

Soldier who wishes to access the information communicated by the commander will log in to the system, gets his all credentials, viz., the access policy privileges, that is, the set of attributes defined by the Commander, which should be satisfied in order to access the data (for instance, “battalion number”, “region”, etc.), verified and only after all of them have satisfied the prerequisites, then only can attain the encrypted copy of the message, decipher it and access the original data.

5. Implementation

5.1 Simulation

Simulation of our proposed scheme is carried out on eclipse IDE in Java. We simulate the Decentralized DTN schema in Java networking as a standalone application in the IDE. The database employed is Microsoft’s access MDB and user interface is coded in java applets and AWT. We employ the java's default RMI server (Remote Method Invocation). Client-Server communication can be established over different machines by incorporating this RMI and thus enabling the remote access to the system. We’re going with RMI model because our application is a standalone application and clients need to interact with the server to access the services of the system, in our case clients being the soldiers and server being the commander. The main advantage of employing RMI server is that it sustains the system’s objective model at the same transmission time of both client and server is saved as they can cope up with the exceptions and mistakes quite quickly with the aid of RMI server.

Figures 4.1–4.7 shows the scenario of how a sender encrypts the data using AES scheme, determines the attribute keys for facilitating access privileges to the user and finally uploads the encrypted data to the storage node.

4.1: Sender uploads the plain text for encryption. 4.2: Plain text is encrypted using AES scheme. 4.3, 4.4, 4.5, 4.6: Sender defines the attribute keys to grant access privileges

![Figure 3](image-url)

Figure 3. Home page of the System in simulation.

![Figure 4.1](image-url)

Figure 4.1. Sender uploaded the plaintext.

![Figure 4.2](image-url)

Figure 4.2. Plain text encryption.
Figure 4.3. Entering the file name (attribute 1).

Figure 4.4. Entering the IP address of storage node (attribute 2).

Figure 4.5. Determining the Battalion number (Attribute 3).

Figure 4.6. Determining the Region (Attribute 4).

Figure 4.7. Keys received to Sender from Key Authorities.

viz., name of the file, IP address of the node, Battalion of the soldier, Region of the Battalion. Figure 4.7: Secret key received from the key authority, in this simulation test case Key Authority 1. The Key Authority to generate the secret key is selected randomly at each instance using the rand() method. After these steps the file is successfully uploaded to the storage node.

Now registered clients (Soldiers) in the storage node can access the data after the Sender determines the Privileges for the each client associated with him. To access the data uploaded to the storage node, the client should register with the same attribute set as the file got uploaded while registering. In this test case, after entering his username and password, the remaining attributes
such as IP ADDRESS, BATTALION NUMBER & REGION should be exactly same as that of the uploaded file (Here IP Address is 127.0.0.1, Battalion number should be B2 and Region should be R3) Only then the client can access this particular file uploaded by the Sender. During registration if any of the attribute is mismatched, then the client is prevented from accessing the data as per CP-ABE. Here it should be noted that, if a client is already registered with the same attributes of the file, before the actual uploading takes place, that particular client can access the file as all the attributes are matched as per CP-ABE.

After successful registration, the client logs in to the system and access the required data as per the CP-ABE scheme with valid attribute keys and personal key.

Figures 5.1–5.8 depicts how a user access the data with his valid attribute keys and personal key from the storage node and process captured in eclipse IDE

Figure 5.1: User starts accessing the data, first step here is to enter the IP address of the node from which the user trying to retrieve the data. It is considered to be one of the attribute keys.

Figure 5.2: User next enters the second attribute, name of the file and its extension. It is to be noted that extension of the file is quite important. If user cannot give the proper extension of the file while retrieving, file access is blocked and such a user is placed in attacker list.

Figure 5.3: User now enters his unique personal key of the data issued to him by the key authority. In this simulation it is issued by key authority 1.

Figure 5.4: User enters another attribute “Battalion Number”. It is to be noted here that it should be the same battalion the sender had intended the file access and to which the user had registered during the registration process.

Figure 5.5: User enters the final attribute “Region”. It should also be the same region to which the sender had intended the access privileges and to which the user belongs, to which he had registered during the registration process.

Figure 5.6: The encrypted file is successfully received by the user as all the attribute keys and unique personal key are valid and satisfied according to the access privilege policy.
Figure 5.1. End user entering another attribute: battalion.

Figure 5.2. End user entering final attribute- Region.

Figure 5.3. File successfully retrieved by the end user.

Figure 5.4. End user entering another attribute: battalion.

Figure 5.5. End user entering final attribute- Region.

Figure 5.6. File successfully retrieved by the end user.

Figure 5.7. End user saving the recovered data.

Figure 5.8. Process captured in eclipse IDE.

5.2 Evaluation

5.2.1 Data Confidentiality

We've already seen how a system can be guaranteed. Client should have the constraints we've defined.

5.2.2 Collusion-Attack Protection

Our system is designed in such a way that none of the local key authorities and external storage node are completely trusted, enabling the plain text to be kept secret from both of them as well as the unauthorized clients.

We've already defined the security requirements in the previous sections. Now let us evaluate our System as per the constraints we've defined.
components of the attribute keys are also re-encrypted. Therefore, backward secrecy is achieved.

Coming to forward secrecy, when any of the clients annuls his attributes that satisfies the data access policy, this is notified to all the remaining clients in the group and the attribute set is updated. Thus, the client who had revoked his attribute set cannot access the plain text from the storage node as the attribute keys are newly updated.

6. Conclusion

Secured mobile device communication in the military environment which is very much hostile and unfriendly can be successfully achieved through the implementation of Decentralized DTNs. In this paper, we've designed an efficient data recovery scheme for decentralized DTNs based on CP-ABE methodology, making security the utmost prioritized characteristic. We've been successful in implementing this system to achieve the three security requirements we've been discussing at length in the paper and we've demonstrated how to recover the data in very hostile military environments in an efficient and a secured manner.

7. Acknowledgements

The constant support and encouragement of Koneru Lakshmaiah University (K L U), is gratefully acknowledged. This work wouldn't have been completed without the University Support and we're forever thankful. Our special thanks to, “Embedded Systems and Sensor Networks (ESSN)” research group of K L U.

8. References

1. Kate A, Zaverucha GM, Hengartner U. Anonymity and security in delay tolerant networks. IEEE, Security and Privacy in Communications Networks and the Workshops, 2007, Third International Conference at Nice, France, Secure Comm. 2007. p. 504–13.
2. Available from: https://sites.google.com/site/dtnresgroup/home (Date Accessed, 2015 Nov 18)
3. Oliveira et al. Identity-based encryption for sensor networks. Pervasive Computing and Communications Workshops, Fifth Annual IEEE International Conference at White Plains, NY, PerCom Workshops ’07. 2007. p. 290–4.
4. Yang G, Rong C, Veigner C, Wang J, Cheng H. Identity-based key agreement and encryption for wireless sensor networks.
In this section the remaining part of the system which we didn’t consider in the simulation. We’ve designed our system with three key authorities. As one of these authorities is selected randomly, the remaining two authorities have not much role to play in the further proceedings. But we cannot know which one of them will be selected as it is selected by rand() methodology. We are providing here the remaining two key authorities that are not selected for generating keys in this test case. We also provide here the storage node of the system which represents both the list of files associated with it and the list of attackers who tried to access the data in an unauthorized fashion, say it may be with annulled attributes, without proper privileges, etc.,Figures 6.1 and 6.2 depicts the local key authority 1. Every key authority displays all the registered users, their privileges and the keys the authority has issued. It should be observed here that even though the key authority 1 knows the personal key of the user, it still doesn’t know the encrypt-decrypt AES key, which it can never have access to. So, our first requirement of security, Data confidentiality has been achieved.

Figure 6.1: Key authority 1 displaying the list of registered users. We can see the user “tester” we’ve created for this simulation.

Figure 6.2: Key authority 1 displaying the list of secret keys it has generated for the files. As the file we’ve uploaded for this simulation fell under key authority 1 via rand() method, key authority 1 has issued the secret key for our file.

Figures 7.1, 7.2 and 7.3 depicts the key authorities 2 and 3 same like that of the key authority 1. As they haven’t generated keys for this test case simulation, we’re not bothered to go into deep about them. It is to be noted that

APPENDIX

In this section the remaining part of the system which we didn’t consider in the simulation. We’ve designed our system with three key authorities. As one of these authorities is selected randomly, the remaining two authorities have not much role to play in the further proceedings. But we cannot know which one of them will be selected as it is selected by rand() methodology. We are providing here the remaining two key authorities that are not selected for generating keys in this test case. We also provide here the storage node of the system which represents both the list of files associated with it and the list of attackers who tried to access the data in an unauthorized fashion, say it may be with annulled attributes, without proper privileges, etc.,Figures 6.1 and 6.2 depicts the local key authority 1. Every key authority displays all the registered users, their privileges and the keys the authority has issued. It should be observed here that even though the key authority 1 knows the personal key of the user, it still doesn’t know the encrypt-decrypt AES key, which it can never have access to. So, our first requirement of security, Data confidentiality has been achieved.

Figure 6.1: Key authority 1 displaying the list of registered users. We can see the user “tester” we’ve created for this simulation.

Figure 6.2: Key authority 1 displaying the list of secret keys it has generated for the files. As the file we’ve uploaded for this simulation fell under key authority 1 via rand() method, key authority 1 has issued the secret key for our file.

Figures 7.1, 7.2 and 7.3 depicts the key authorities 2 and 3 same like that of the key authority 1. As they haven’t generated keys for this test case simulation, we’re not bothered to go into deep about them. It is to be noted that

1. M. S. Sarath Chandra, K. Raghava Rao and Mohammed Ali Hussain

APPENDIX

In this section the remaining part of the system which we didn’t consider in the simulation. We’ve designed our system with three key authorities. As one of these authorities is selected randomly, the remaining two authorities have not much role to play in the further proceedings. But we cannot know which one of them will be selected as it is selected by rand() methodology. We are providing here the remaining two key authorities that are not selected for generating keys in this test case. We also provide here the storage node of the system which represents both the list of files associated with it and the list of attackers who tried to access the data in an unauthorized fashion, say it may be with annulled attributes, without proper privileges, etc.,Figures 6.1 and 6.2 depicts the local key authority 1. Every key authority displays all the registered users, their privileges and the keys the authority has issued. It should be observed here that even though the key authority 1 knows the personal key of the user, it still doesn’t know the encrypt-decrypt AES key, which it can never have access to. So, our first requirement of security, Data confidentiality has been achieved.

Figure 6.1: Key authority 1 displaying the list of registered users. We can see the user “tester” we’ve created for this simulation.

Figure 6.2: Key authority 1 displaying the list of secret keys it has generated for the files. As the file we’ve uploaded for this simulation fell under key authority 1 via rand() method, key authority 1 has issued the secret key for our file.

Figures 7.1, 7.2 and 7.3 depicts the key authorities 2 and 3 same like that of the key authority 1. As they haven’t generated keys for this test case simulation, we’re not bothered to go into deep about them. It is to be noted that
only key authority 1 knows the secret personal key of the user for this simulation case, both the remaining authorities are ignorant of the personal key, this implements the collusion-attack prevention mechanism.

Figures 8.1 and 8.2 depicts the storage node displaying the files stored in it and also the attackers (unauthorized) list. Here it is noteworthy that not only the user who tries to access the data in an unauthorized manner is placed in

**Figure 6.2.** KA1 list of keys (we can see the newly uploaded file sample.txt key is issued from this authority).

**Figure 7.1.** Key authority 2.

**Figure 7.2.** Keys in KA2.

**Figure 7.3.** Key authority 3.

**Figure 8.1.** List of files stored in the node.

**Figure 8.2.** List of attackers.
this attacker list but also those users who entered wrong attribute keys or expired/revoked attribute keys during the data access, thus achieving the forward secrecy and backward secrecy.

Figures 9-11 represents the client registration scenario, client log in and Sender assigning privileges to the registered clients.

**Figure 9.1.** Client registration home page.

**Figure 9.2.** Client enters his details and also enters the storage node IP address. Here it is to be noted that client registers himself based on the attribute set defined by the sender. In this case with IP address of the node, Same battalion number and region the sender intended for the file he uploaded into the node.