Searchable Encrypted Mobile Meter
Senior Thesis
Clayton Quinlan
Computer Science, University of Illinois, May 2014

Abstract. Securing data and ensuring privacy is one of the most difficult challenges as technology becomes increasingly more mobile. Mobile devices allow us the convenience of recording and accessing data from anywhere but, when mismanaged, can be a major source of privacy violation. Furthermore, mobile devices tend to have restraints that limit their computing power which leads to sacrificing security for power. The rest of this paper will focus on securing mobile data as related to Electric Vehicles. Specifically, this paper takes a look at securing a mobile meter that is to be placed in an electric vehicle. This device will use modern encryption techniques while still allowing private queries to the system using Role Based Access Control. This method provides strong security against an adversary who has physical access to the mobile meter as well as allowing multiple users to have access to the same data set while restricting or elevating their privileges. Securing the relatively computationally weak mobile meter in this method also provides the added benefit that most of the computation can be done on a more computationally fit computer. Securing mobile meters in electric vehicles is an area of substantial interest as the amount of information one can potentially gleam from a mobile meter is surprisingly high. This paper also discusses the information that could potentially be gained from an insecure mobile meter.

1. INTRODUCTION

Electric Vehicles are becoming increasingly popular, by 2018 the expected market share of electric vehicles is 25 percent [1]. Part of this popularity is due to the fact that electric vehicles have reduced energy emissions and also provide a mean to energy independence. As electric vehicles become increasingly popular the need for more efficient and convenient charging methods arises. One such method that is being proposed is an on-road dynamic charging system with wireless power transfer technology [2]. In this method the electric vehicle would need to perform some sort of authentication to the system in order to transmit certain data such as its battery type and other necessary charging configurations. In addition, prior to obtaining time on the charging system, the electric vehicle would need to negotiate its method of pay to the charging system such as a credit card or some sort of utility account information. Most of this data will necessarily be stored in the car on the proposed mobile meter. Having a mobile meter presents several unique challenges: securing the data from theft, storing all of the necessary information on it, having high performance while minimizing power consumption and providing access to the data to those who need it, when they need it.

1.1 RELATED WORK

The proposed mobile meter is an adaption from the current smart meters that are being used on the power-grid [3]. Smart meters are devices used by utility companies to keep track of electricity use. It contains more information than older methods [3] as well as provides a way for utility companies to wireless collect energy usage. This fits the goals of the proposed mobile meter with the exception that the mobile meter will not be static. This means that the mobile meter will have to use a different method of communication from the user to the utility company. It will also have to measure the electric input and output differently as electric vehicles play a critical role in load balancing on the power-grid [1].

As previously mentioned, the smart meter is only ever accessed by the utility company (unless they provide you with an interface to it) so they don't have to worry about multiple users in the same sense that the mobile meter will. The mobile meter may be accessed by the owner of it, various utility companies, mechanics, a friend of the owner or even the police. This classification of users into groups makes Role Based Access Control (RBAC) a good fit as the central notion of RBAC is that permissions are associated with roles, and users are assigned to appropriate roles. This greatly simplifies the management of all the permissions. Roles are created for the various job functions in an organization and users are assigned roles based on their responsibilities and qualifications [4]. This separation of job and

users allow different people to be given access to the system without having to duplicate the data excessively and it also allows one to limit the amount of data some user can view or manipulate.

1.2 Approach and Contributions

The proposed mobile meter is an adapted smart meter with RBAC. In the case discussed in this paper the owner of the mobile meter is responsible for generating the various roles. Each user will have to have a private key *K*, which they use to authenticate to mobile meter and decrypt their role token. The mobile meter is essentially a MYSQL server. It contains an AES (or your favorite encryption algorithm) hardware module to encrypt the data that the sensors of the car are reporting to it. This data may consist of GPS coordinates, engine temperatures, battery life, charging location, etc. The mobile meter then stores this information encrypted for the various roles until it is queried for using the index based query method proposed in this paper.

In this paper the problem of having secure searchable data being stored on a potentially insecure database is addressed. In this approach RBAC is implemented which, in an insecure environment, sacrifices space for security. Having an increased space requirement also corresponds to an increased query (search) time. While an increased query time is not desired it is a sacrifice that has to be made in order to maximize security. To minimize this increased query time certain optimizations can be made that trade security for speed and are discussed in section 4. The following outline provides insights to the other various contributions of this paper.

Section 2 contains various background information that introduces those unfamiliar to common security methods such as RBAC, the notion of security and privacy, public private key infrastructure and authentication.

Section 3 discusses the idea of the *Searchable Encrypted Mobile Meter (SEMM)* in much greater detail. In particular section 3.1 discusses the need and motivation for such a device; section 3.2 discusses the the data models and the assumptions that were made while designing SEMM. This includes the following the type of data being stored, the type of roles, user models, key generation, and physical device model. Section 3.3 contains all of the services, functions and components of SEMM.

Section 4 contains the implementation and experimentation validation. This consists of the data structures used, major libraries, pseudo code of key methods, a description of the tests ran as well as the results.

Section 5 contains the conclusions and lessons learned as well as future work.

2. Background

A major purpose of SEMM is to secure and facilitate data acquisition between a user and the utility company. In this section the bigger picture as to why SEMM is designed to be secure and also how it helps facilitate data acquisition from user to utility company is explained. The primary technique used so secure SEMM is encryption. SEMM uses AES / ECB / PKCS5Padding. AES is an encryption scheme that was selected by the United States National Institute of Standards in 2001 as the official encryption method of the United Sates government, replacing DES. What is this magical encryption and how does it make SEMM more secure? Encryption is a way of encoding data in such a way that only authorized sources can decode it in a way that is meaningful. To simplify this method you can think of it like this, a user enters in a key $K°$, this key is then entered into a matrix (we'll say 4x4). The first round of encryption simply XORs [5] the first block of data with the key block. Next lots of different keys are generated for subsequent encryption rounds based off of the first key. These keys are generated subsequently from $K°$ in the following manner: take the last column of the matrix (in this example it would be the 4^{th} column) swap its last byte with its first byte and run it in a substitution box. This simply maps it to some other character, for example, a → c and b → z . You then XOR this data with some 'round constant' that is different for each round. Finally you XOR it with the first column of the previous rounds key to

generate the new first column. To generate the rest of the columns in the new round key just XOR it with the previous columns in the current round. So say the last column of the first 4x4 matrix was [M,A,T,H] and the first column was [T,R,I,P]. First you would swap M with H to get [H,A,T,M]. Next you would substitute each letter maybe leaving you with [G,Z,S,L]. Next you would XOR with some constant like [01,00,00,00] (note that is in hexadecimal) leaving you with [F,Z,S,L]. Lastly you would xor this value with [T,R,I,P] resulting in [12,08,1A,1C] (also hexadecimal). And viola you have your second key's first column. In the end all of that work was to add confusion and diffusion making it increasingly difficult to guess what the key is. These keys are then used to encrypt the data.

Another technique that is implemented into SEMM is Role Based Access Control. RBAC was designed around the basis of security administration and review [4]. RBAC provides three of the main security stigmas: least privilege, separation of duties, and data abstraction. Least privilege is supported because the administrator has the power to give only members of a role permissions to only do that role and nothing else. Separation of duties can be implemented by ensuring roles that involve sensitive tasks are mutually exclusive. Data abstraction is provided by means of abstract permissions (you could assign a charge as positive or negative instead of read or write). The main benefit of a RBAC in SEMM is that it creates a mean for different persons with the same profession to be able to access the same data, the same way, while not having to change the data's architecture in anyway. In doing so you allow different utility companies or different car mechanics to be able to access the system.

The last security technique I will discuss is public key cryptography. Public key cryptography is a method of sharing keys between two sources in a secure manner over an insecure channel. The way it works is first the user generates two keys, $PRIV_{key}$ and $PUB_{key.}$ Next, the user generates two large prime numbers p and q and computes the following:

$$n = p \times q$$

$$\Phi(n) = (p - 1) * (q - 1)$$

Alice then computes an integer d that is coprime to $\Phi(n)$. After which she computes:

$e = d^{-1} \bmod \Phi(n)$

Alice now sets her $PUB_{key} = (n, e)$ and her $PRIV_{key} = (n, d)$. Now for two users, we'll call them Alice and Bob, to share a message who have both public and private keys the following sequence occurs: Bob first obtains Alice's public key, $Pub_A = (n, e);$ next Bob takes the message m and raises it to the power $e;$ then he takes it modulus with n and transmits this.

$$c = m^e \bmod n$$

To decrypt Bob's message, Alice takes her private key $Priv_A = (n, d)$ and raises c to the d power and then takes the modulus n.

$$m = c^d \bmod n$$

This completes the message transfer between Bob and Alice. This is important to this paper because eventually the SEMM will need a way to communicate securely with the aforementioned charging stations. The following table holds the variables and equations used in these calculations:

Equation	Variable Explaination
$n = p \times q$	p,q are larger prime numbers because it is difficult to factor large prime numbers n is used in both public and private key
$\Phi(n)=(p-1)*(q-1)$	$\Phi(n)$ is used in the generation of the public key
$e = d^{-1} \bmod \Phi(n)$	e is used in the public key d is coprime to $\Phi(n)$ is used in the private key generation
$c = m^e \bmod n$	C is the encrypted message m is the plaintext message

3. SEMM

As previously mentioned SEMM stands for Searchable Encrypted Mobile Meter. The bigger picture of SEMM is to have a device placed in an electric vehicle that is capable of quickly and securely storing and accessing data

generated by the car. This information may need to be accessed by the utility company later in order charge or pay certain amounts of money to its customers.

3.1 NEED AND MOTIVATION

Electric Vehicles have the potential to cause a big impact on both reducing emissions and energy independence in the United States [1]. In addition to being eco-friendly they also can have a huge impact on our smart-grid and preparations need to be made. Electric Vehicles are extremely power intensive and it is expected that there will be a massive increase in the usage of night time power [6]. They are also unique in that they roam from place to place while simultaneously being both a load and a source to the grid [6] [7]. In order to manage the increase of electric vehicles it will be necessary to have methods in place to be able secure this dataset.

3.2 MODELS AND ASSUMPTIONS

There are many models and assumptions that needed to be considered and made in SEMM. The first question that needed to be answered is what type of platform will this device be running on? This led to the question who will be supplying SEMM? Will the car manufactures be incorporating SEMM into electric vehicles or will users be going out and buying SEMM as an add-on to their vehicle? For the purpose of this paper it is assumed that the car manufactures will be incorporating SEMM into the future electric vehicles as it provides a convenient method for everyone to gain information about how people are driving as well as a really powerful diagnostic tool. With that assumption made I can assume that the final device will be running on the cheapest piece of hardware that the car manufactures can make it work on. To that end it can be speculated it will be running some sort of raspberry pi spin off (linux based) with a hard drive that is just large enough to store all of the data. So now the question arises what is this data that is being collected?

There appears to be no standard in the data collected from electric vehicles [8] [9]. However, there are a lot of overlaps between sets. In almost all of the sets of data pertaining to electric vehicles the following fields tend to show up: Charging event (unique key generated by charging station), user id, charging station id, start date, start time, end date, end time, total kWh, Cost, Site Name, Longitude, Latitude, Lot Operator, Charger Type, Charge Level, and Access Type (public, private). This data is generated by the interaction between the car and the charging site. The car will also generate its own data such as engine temperature, engine status, fluid levels, gas levels and GPS locations. So there needs to be some way of collecting this data and sending it to a SEMM. Sensors will be used to feed data into the SEMM model which will ultimately be responsible for polling (polling means periodically reading) these sensors and encrypting / maintaining the data in a MYSQL database.

The next problem that needs to be discussed is how does SEMM handle roles and users in a RBAC fashion. SEMM has essentially two tables, a data table and a user-role table. The user-role table contains the columns:

User—Password—Role 1—Role 2—Role n

The user column contains the user's authentication name encrypted using his private key. The Password column contains the user's password hashed and encrypted. The *Role x* column contains the role key (encrypted with the users private key) or random data if the user does not have access to that role. The reason the random data is necessary is to prevent a malicious adversary who can look at the data to be able to trivially determine what users have access to what roles. So when a user authenticates to the system as a role he submits the following query to receive the appropriate token:

SELECT Role x FROM table WHERE
user=E(user_name, key) AND
password=E(h(password), key)

Note: that E(*user_name, key)* means to encrypt the text user_name using the private key, key and that h(password) means to hash the text password.

From this query the user will receive the

encrypted role token which will be decrypted using their private key leaving them with the token for the role queried for. This method requires that you trust the users that have accounts on your system as once they have the role token they could potentially share it with malicious users. As far as anticipated roles in the RBAC model there will have to be an administrative role (to generate all the other roles), utility reading role, mechanic role, and a police role. Each of these roles can be further broken down into weeks to make query sets smaller.

Since the computer on the car will most likely be computationally slow our model seems to be coming along nicely. So far all it has to do is encrypt the data once (which can be optimized via hardware) and then store it on a hard drive. The next step to consider is the generation of private keys and key management. When the car is running it has to be able to access all of the tokens in order to encrypt the data to the database. So when the car is running it has to have admin privileges as that should be the only user to have access to all of the different roles. The admin key then needs to be stored somewhere on the car so it can appropriately retrieve and decrypt all of the tokens. One idea for this is to have it encoded within the SEMM's hardware. One foreseeable problem with this is that if SEMM is stolen from the car this module will be included with it and if its not properly secured then it could leak the administrative private key to the adversary. Another more novel solution is to have the key to the car actually store the private key to the car and then send SEMM the administrative key when it turns the car on.

The last and perhaps most crucial assumption that this model is making is that the computer connecting to the car is of average computational strength. With that being said I would classify average computational strength to be at least 4 gigabytes of memory, a minimum processor of the Intel Celeron family 2.17 GHz (although i3 processor is much more preferred) and a hard drive with at least 20 GB free. The reason for these requirements is that most of the encryption / decryption that will be done during the querying phase of SEMM will be done client side so its important that the client has a computer that is capable of handling these processes. While this does not rule out the use of computationally weak client computers, this method is not tailored around that ideology.

3.3 SEMM ARCHITECTURE

This section will discuss the architecture of SEMM as related to services provided, components of SEMM and protocols used. SEMM provides a storage method of private data using AES / ECB / PKCS5Padding encryption. It contains methods to collect data from the various sensors in the car via serial connection as well as a component that can wireless connect and gather charging information from charging stations. Both of these methods are outside the scope of this research paper. SEMM must also have a hardware module whose sole purpose is to perform encryptions on the incoming data as doing it in software would be computationally inefficient.

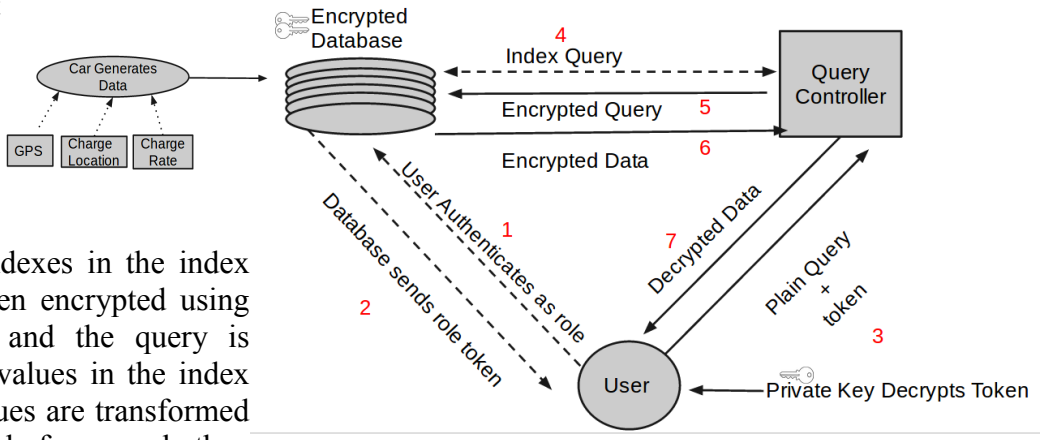
Their are several key protocols for data handling that are used in SEMM. The first is the storage of encrypted data on a MYSQL database. Whenever SEMM decides to poll for data it generates a random string of a predefined length and concatenates this random string to all of the data that is to be encrypted. This data is then encrypted and sent to the MYSQL database along with random string concatenated to the each piece of data. The purpose of this random string is to ensure that the data in the database is not repetitive as one could determine if two rows contained the same piece of data but they would be unable to determine what that data is.

The second protocol is how queries are created and sent to the MYSQL database. First and foremost, the only queries that this encryption method is capable of are queries in the form of:

SELECT some_value FROM table WHERE value = A

The way one queries the database is they first query for the random indexes that were concatenated to the strings. They store this in an index list. Next they take their search value *A* and prepend it to the indexes in the index list. The index list is then encrypted using the current role token and the query is generated for all of the values in the index list. These encrypted values are transformed into the aforementioned form and then queried on the database. The information is then retrieved in list format and decrypted on the client side.

Figure 1.



4

After evaluating the models and assumptions, as described section 3, figure 1 was created to best represent SEMM. As we can see the car generates data and stores it to the encrypted database using the index based method discussed in section 3. There are essentially seven steps to performing a query on this data set. Step 1 is to generate and send the authentication query to the database using the method discussed in section 3. Step 2 is to receive and decrypt the role token. Step 3 is to create the plain query and send it to the query controller. The query controller is a state that translates the plain query into the many encrypted queries and sends it to the database. Step 4 is to receive the index list from the database. Step 5 is to generate and send the many encrypted queries from the received index list. Step 6 is to receive the encrypted data back from the server. Step 7 is to decrypt the data using the role token.

4.1 System Implementation

This system was implemented in Java on two Ubuntu 13.04 boxes, one being the client, the other the server. These boxes were on a local network and were using the nursery data set [10] to test the encryption scheme. The testing methods will be discussed in detail in section 4.2. Some of the major Java libraries that were used were the Javax.crypto for performing encryption / decryption, the java.security for key generation, and the java.sql for easy sql access (external library to java).

One of the more difficult tasks in implementing this solution was generating the query strings. We can only do queries in the form of *Select * were column=value.* If we look in section 3 or look at figure 1. we can see that we have to create *n* queries for each value that we want search for because of the way the random indexes work. This is where we have to optimize based on both the server/client computer's memory. If we were to send 1 query for each random index we would send *n* query's which is very slow but uses very little memory. Alternatively we could send 1 large query containing *n* elements in it. This requires a lot of memory but very little network usage. The ideal solution is shown to be somewhere in between in section 4.2. The code that generates this ideal solution is the following:

```
function query( String val, Key key ):
    queryCount = 0
    MAGICNUMBER = 100
    queryString = “”
    for string rand in randIndexList:
        queryString += client.encryt( val, key, rand)
        queryCount += 1
        if count % MAGICNUMBER == 0
            database.query( queryString )
            queryString = “”
```

Observe that MAGICNUMBER was define to be 100 in the pseudo code but that number is actually dependent on both the client and server computer. The reason this code speeds optimizes the system is that it generates queries that are large enough to make the most use of the memory available which in turn means larger query sizes which means less network latency. This strategy is also used when submitting large amounts of data to the server at once.

The original implementation used the ArrayList Java data structure but as it turns out the ArrayList data structure is not thread safe which prevents one from optimizing the query times. This lead me to discover the copyOnWriteArrayList data structure which has the exact same functionalities as an ArrayList but is thread safe. This safeness comes at the cost of additional memory overhead.

4.2 Experimentation Validation

The testbed in which the experiments were performed on will first be defined. All of the experiments were conducted on two computers on a local network, one computer acted as a mock server and the other computer acted as a mock client / data source. By data source I mean it would generate the data and send it to the server as described in Section 3 and shown in figure I. Both computer had 8 gigabytes of memory and had i7 processors with over 20 gigabytes of hard drive space. Its important to note both computers are much better than that of the target goal of SEMM.

The metric that is most concerning is time. Speed is often what is sacrificed when one add security. The first experiment that was performed was the encryption time of large sets of data based on the number of roles (figure 2). As one can see from the figure, as the number of roles the encryption time increases. This measurement is important because it shows how this system scales with increasing number of roles. One thing to note is that the system should never have to encrypt 12,960 rows of data at the same time. The second experiment that was performed was the optimization problem. The original bottleneck of the SEMM was sending massive amounts of insert queries to the MYSQL database. After some experimentation it was found that there is

Figure 2.

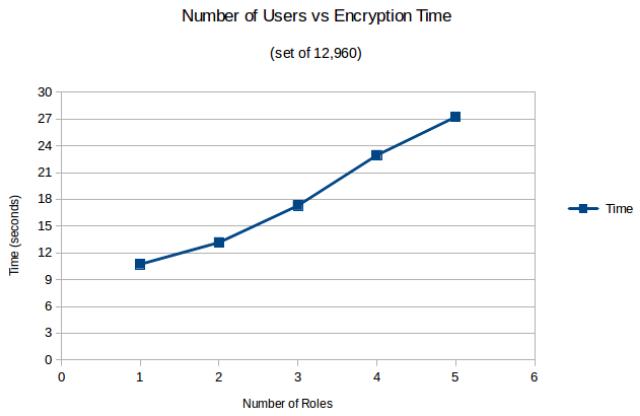


Figure 3.

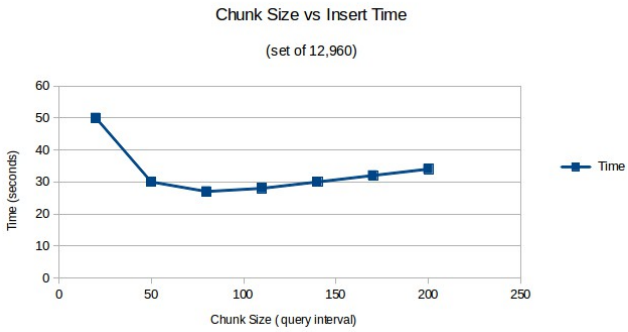
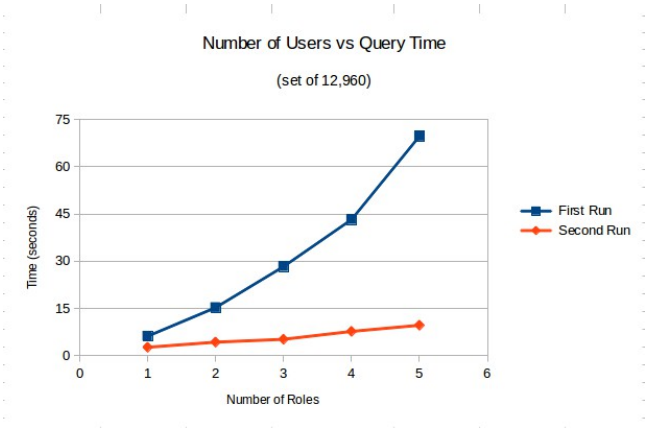


Figure 4.



a sweet spot between the number of queries sent and the size of the query. The reason that we can't send one massive query is that there are server side limits and there are memory limits on the client side that cause for slower behaviors when we fill that memory. The data suggests that the sweet spot for the testbed system is about 80 inserts per query. The third test that was performed was the query time versus the number of roles. This is important because the system should scale very smoothly with a large number of roles. The data suggests that the query time increases almost quadratically for each role we add. A possible reason for this is we still have to perform *n* queries even if only searching for data that is owned by our role. The reason for this is that if one doesn't do this a side channel is created where a user could see what rows you looked at and then say OK these rows are utility rows which isn't the biggest side channel but it still revels information that doesn't have to be revealed. This experiment also shows that subsequent queries are very quick because of caching and essentially the query time for successive queries on an increased number of roles grows very slowly.

5. CONCLUSION

Presented has been an extremely precise SEMM model that successfully allows the Searching of an Encrypted Mobile Meter. Many of the assumptions and models that were used leading to the design of this SEMM provide a common frame of reference for other various research and development topics in this area. While the index based model used in SEMM is extremely easy to implement, the index based model may not be the fastest method to be used in these privacy preserving queries on an RBAC system [11]. It would be interesting to see how the 'checksum' method compares to the index based method over network connections. The other more open areas of research related to this topic is the key space generation. How are keys distributed to and from users? Are the users just given one key with the car at the beginning of time? What happens when the key becomes compromised? How does SEMM partner up with these newer wireless charging method? Does it allow everything to connect to it? How does it decide what charging sites need to see its data and how does it determine when it needs to send its data to the utility companies. Answering these questions will hopefully further encourage the production of public charging stations as well as the creation of more electric vehicles.

6. ACKNOWLEDGEMENTS

This research paper was made possible through the help and support of many people including: funds from Information Trust Institute, Illinois Cyber Security Scholars Program, TCIPG, Andrew Vojak and Adam Hitches. I would like to especially thank Professor Klara Nahrstedt and everyone in her research group for providing copious amounts of feedback and idea generation on this research topic.

.

References

[1] Department of Energy, “Communications Requirements of Smart Grid Technologies”, October 2010
[2] In-Soo Suh; Jedok Kim, "Electric vehicle on-road dynamic charging system with wireless power transfer technology,"*Electric Machines & Drives Conference (IEMDC), 2013 IEEE International*, vol., no., pp.234,240, 12-15 May 2013
[3] Depuru, S.S.S.R.; Lingfeng Wang; Devabhaktuni, V.; Gudi, N., "Smart meters for power grid — Challenges, issues, advantages and status,"*Power Systems Conference and Exposition (PSCE), 2011 IEEE/PES*, vol., no., pp.1,7, 20-23 March 2011
[4] R. S. Sandhu, E.J. Coyne, H.L. Feinstein, C.E. Youman (1996), "Role-Based Access Control Models",IEEE Computer29(2): 38-47, IEEE Press, 1996.- proposed a framework for RBAC model
[5] Davies, Robert B (28 February 2002)."Exclusive OR (XOR) and hardware random number generators". Retrieved 28 August 2013.
[6] J. Gorman, Y. Glick, R. Hourdouillie, “Smart-grid communications: enabling next-generation energy networks”, EBR 2012
[7] Bradley, Frank, “Design, demonstrations and sustainability impact assessments for plug-in hybrid Electric vehicles”, Renewable and Sustainable energy reviews, January 2009.
[8] “City and County of Denver, Public Works - Traffic Engineering Services / Parking Operations”, Electric Vehicle Charging Stations, May 2014
[9]http://data.glasgow.gov.uk/dataset/f0c608a1-f272-4f97-86a2-0e724ee2e273/resource/319883ce-937a-41d9-97ec-c4578adc999c
[10]http://archive.ics.uci.edu/ml/datasets/Nursery
[11] Zhiqiang Yang, et al. “Privacy-Preserving Queries on Encrypted Data”