AN EFFICIENT SOLUTION FOR PRIVACY-PRESERVING, SECURE REMOTE ACCESS TO SENSITIVE DATA

Kalpana Singh[1], Jian Zhong[2], Lynn M. Batten[3] and Peter Bertok[4]

[1,3] School of Information Technology, Deakin University, Burwood Campus,
Melbourne, Australia
[1]kalpana@deakin.edu.au, [3]lynn.batten@deakin.edu.au
[2,4] School of Computer Science and Information Technology, RMIT University
Melbourne, Australia
[2]jian.zhong@rmit.edu.au, [4]peter.bertok@rmit.edu.au

ABSTRACT

Sharing data that contains personally identifiable or sensitive information, such as medical records, always has privacy and security implications. The issues can become rather complex when the methods of access can vary, and accurate individual data needs to be provided whilst mass data release for specific purposes (for example for medical research) also has to be catered for. Although various solutions have been proposed to address the different aspects individually, a comprehensive approach is highly desirable. This paper presents a solution for maintaining the privacy of data released en masse in a controlled manner, and for providing secure access to the original data for authorized users. The results show that the solution is provably secure and maintains privacy in a more efficient manner than previous solutions.

KEYWORDS

Privacy, Security, Sensitive Data

1. INTRODUCTION

Sharing and exchanging of large amounts of data can be essential in responding efficiently and effectively to critical situations such as bush fires, earthquakes or medical emergencies. However, privacy and security issues continue to be major barriers, since revealing sensitive data over the Internet can present a serious threat. Digitally stored medical data accessed by an unauthorised person, or uncontrolled transfer of medical data related to an individual's health condition and medications can lead to major problems with both short term and long term implications. For example, if the information is collected by an insurer or by the patient's employer, the patient may subsequently find that she is unable to obtain insurance or has lost her job.

There may be many interpretations of 'data privacy and security'. Here, by privacy we mean the 'inability to deduce the identity of an individual from the data presented about that individual', while in case of security we focus on confidentiality.

In the research literature, the existing solutions for maintaining data privacy are based on determining how to reduce the probability of an individual's identity discovery when a large amount of private data is shared in an online environment. But, there is always a trade-off

Natarajan Meghanathan, et al. (Eds): SIPM, FCST, ITCA, WSE, ACSIT, CS & IT 06, pp. 173–191, 2012.

DOI : 10.5121/csit.2012.2318

between privacy loss and information loss, and to preserve the accuracy of results and to reduce loss of information, a number of techniques have been developed. Most existing solutions focus on data modification [1], [2], [3], [4], [5] and [6], which requires that the data be altered before its release. However, this may consequently reduce data accuracy. In addition, such solutions overlook the possibility of security breaches in which whole sets of (modified) data may be intercepted in transmission and read or altered.

Our objective is to provide data confidentiality and privacy in an efficient protocol. While a number of widely-used communication protocols are available to provide security in the transport layer and below, upper-layer security protocols need to be tailored to the application, and no general solutions are available. However, solutions in the application layer can protect data within the user equipment as well.

This paper presents an efficient application-layer solution for data privacy and confidentiality along the path from the database to the user application. It can be combined with lower-layer techniques to provide additional features, such as authentication, but at the expense of efficiency, as that may require more resources, such as certificate handling or public-key infrastructure (PKI) in general. Efficiency is further improved by selective encryption: only privacy-protected data is given additional, confidentiality protection. This prevents access by unauthorised users, while reduces the probability of subject identification by authorised users. The solution is intended for health-care data, and the target of protection, medical information, is clearly identified in the application.

We describe a data privacy architecture which, when compared to existing solutions, improves on or matches previous data privacy research both in the retention of privacy and in the efficiency of the protocol, and which in addition guarantees security against some of the more common attacks on data.

1.1.Contributions

This paper proposes a method which efficiently reduces the overhead and maintains levels of security and privacy of sensitive information in an online environment without compromising data accuracy. We demonstrate that the architecture proposed in this paper overcomes the drawbacks of existing solutions by effectively maintaining levels of security while reducing the time complexity of other solutions. In particular, we show that the proposed method demonstrates increased speed of encryption/decryption and reduces overhead when compared to several recent privacy-preserving solutions; it does this without compromising on data privacy, accuracy or security of transmissions between sender and receiver. We also show that this privacy model outperforms the best competing privacy models in its ability to maintain data privacy.
Our specific contributions are:

- A privacy preservation method for personally identifiable data, which provides low identification probability with low overhead cost.
- A demonstrably efficient encryption method for releasing sensitive data.
- Proofs that our method is secure against several standard attacks on data.

The rest of the paper is organised as follows. In Section II, we give the background on contributions made by existing solutions. Section III details the proposed architecture. Section IV presents the experimental setup, analysis, simulation results of the proposed architecture, comparison with analogous data on existing solutions and security analysis; Section V presents proofs of security. Section VI concludes the paper with a summary and discussion of future directions of our work.

2. BACKGROUND

In the literature, several methods have been described for preserving the privacy of individuals when publishing data; simulation [1], noise addition [2], micro aggregation [3], randomization and perturbation methods [4], [5] and [6] are some prominent examples. These algorithms modify the initial data in order to preserve privacy while still providing usable data. However, modification to the original data needs to be performed carefully, as large modifications may lead to loss of important information, while minor changes may not be sufficient to protect privacy [7].

One of the oldest solutions to protecting privacy of digital data is known as *k-anonymity* [8]. The basic principle involved is that for any piece of data released, there are at least *k-1* other pieces of data released at the same time which are indistinguishable from the initial one in terms of attributes. Hence, the probability of correctly identifying an individual in the data set is at most 1/*k*. It is very rare that all personally identifiable information in any given data set satisfies *k-anonymity*, and data may need to be adapted before being released in order to achieve this status. In [8], the ideas of 'generalization' and 'suppression' [9] are used to achieve *k*-anonymity. Generalization replaces values with values that are semantically consistent but different from the original. For example, a specific age might be replaced with an age range. In suppression some data is not released at all.

In [10], data linkage is made more difficult by the addition of fake patient data to the original data set. The amount of fake data to be added is a function of the diversity of the data to be privatized; the more diverse the data, the more fake data is added. This approach has the disadvantage of having to manage or transmit larger data sets than necessary.

Since *k*-anonymity and its generalizations were deemed to work only with certain data attributes, and failed to maintain the privacy of others, researchers began to look for better methods. The algorithm in [11] numerically transforms the original data first by grouping and then by applying a method that keeps relationships between data items while obfuscating the real values. In the transformation process the entire data value domain is split into categories and each category is assigned a number. Each data item is put in a category and a membership value within the category is assigned. Then a new, transformed value is calculated by concatenating the category number and the membership number. However, the applicability of this method is limited to cases when the actual value of the original data is not important, such as in some data mining situations.

In [12], the authors manipulate the subject identifiers to prevent information leakage, while the associated data remains unchanged. The method generates new subject identifiers and attaches them to the data, so no personally identifiable information is revealed when the data is published. To make the de-identification process reversible for authorised users, the parameters of the algorithm generating the new identifier is communicated via a secure channel to those users. A significant advantage of this solution is that it preserves data utility. The computational costs of de-identification are reasonable, but the secure communication channel for reversing the de-identification is more expensive.

While the data privacy mechanisms presented in [8], [10], [11] and [12] present a broad range of the research approaches published in the literature over the last ten years, with the exception of [12], none of these papers considered security when the data is transmitted on-line. Thus, these protocols are susceptible to a number of security attacks and so cannot be considered semantically secure.

In this paper, we propose an efficient, secure protocol while also maintaining privacy. We benchmark our work on all three levels: privacy, security and efficiency, against the papers in [8],

[10], [11] and [12]. Section IV explains the experimental setup and gives the comparative results across these three features.

3. PROPOSED ARCHITECTURE

The architecture we propose in this paper is designed to maintain the best levels of privacy as discussed in the previous section, but also to improve on existing data security as well as efficiency by using selective encryption [13], [14] based on appropriate asymmetric and symmetric encryption techniques. The privacy-preserving techniques employed are Salt Generation, Random ID and Random Grouping along with symmetric encryption. These privacy techniques were all employed in the paper [12] and the model here is based on that given there. We show in a subsequent section that in fact, this privacy model outperforms the best competing privacy models in its ability to maintain data privacy.

In choosing an appropriate method of securing data while in transit, we need to decide between systems embedded into network protocols such as SSL and TLS [15] in which a master key is established and subsequent uses of it are managed by the protocol, or introduce our own cryptographic method where the keys are owned and managed by the users. In making this decision, the most important feature for us was efficiency as it may be necessary in an emergency to send data to a small, low-resourced device.

Elliptic Curve Cryptography (ECC) allows for efficient implementation due to shorter underlying bit lengths key sizes (160...256 bit) when compared to RSA or DLP in finite fields (1024...4096 bit) at an equivalent level of security [16]. This results in faster computations and lower power consumption, as well as memory and bandwidth savings [17].

Modern security protocols such as SSL and TLS are widely used in e-commerce applications. These are arithmetic-intensive public-key primitives; the use of SSL increases computational cost of transactions by a factor of 5-7 [18] in comparison with various parameters such as throughput, utilization, cache sizes, file access sizes and network load on the server. In addition, several weaknesses of these protocols exist and are well-known including the possibility of session hijacking [19], [20]. Thus we opt for establishing our own protocols and key management. In fact, we adopt recommendations of [16] and encrypt sensitive data by using asymmetric techniques ECC [21] and RSA [22], and encrypt non-sensitive data by using the symmetric encryption technique AES for maintaining security and efficiency [23], [24].

We use the architecture in [12] as the basis of the current proposal as that architecture is the only one in the literature (to our knowledge) which provides a high level of data privacy and also tackles the issue of data security.

In our model, we assume that the data to be sent contains sensitive information about individuals. Our objective is to maintain the privacy of these individuals by ensuring that any attacker who obtains the data cannot connect this sensitive information with the individual it describes.

Thus, in the setup, a sender S wants to send data M to a receiver R in a secure and efficient manner in such a way that the privacy of the data is preserved. S first determines which portion of M should be considered to be ‘sensitive’ or in need of privacy maintenance. For instance, patient data may be contained in the sender’s database; from this, S may select the name and address as sensitive data, but regard the age of the patient along with the patient’s medical condition as non-sensitive data. We use P to refer to sensitive data, as determined by S, and M-P to refer to the non-sensitive data.

In advance of sending the data, *S* and *R* each first establish three secret keys. Two are asymmetric, or public, keys based on Elliptic Curves (EC) and RSA; the third is an asymmetric key based on AES [23], [24] and needs to be agreed on by both parties.

In addition to establishing in advance the keys to be used, *R* and *S* also establish a common algorithm for use in the privacy management process. This is the 'Salt Generation' algorithm. Salting is a method of adding some random digits to a plaintext before it is encrypted and is done in a standardized manner so that someone knowing the decryption key and the salt digits (unencrypted) can recover the original plaintext. See [25] for an example of how this is implemented. The original purpose of salting was to prevent fast password attacks using look-up tables; however, the method works well as a data privacy technique as the salt digits can be used to 'perturb' sensitive data. An advantage of salting is that it can effectively perturb the target string without a high computational cost.

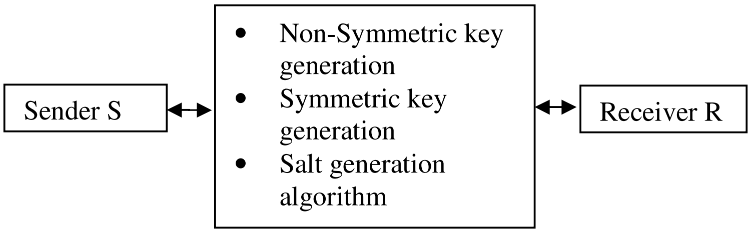


Figure 1. Key generation process in proposed architecture

The total initial interaction between *S* and *R* is described in Figure 1 where key set-up for the EC and RSA asymmetric keys, the symmetric key, and for the salt information is done over an insecure channel. The key generation is described in detail in the next sub-sections. Note that RSA keys are permanent while the other keys are session keys only and must be changed at each data transfer session.

3.1. Key Generation Set-up

Step 1. RSA key set-up (permanent keys). *S* and *R* jointly select an integer n_r which is the product $p_r q_r$ of two large prime numbers. *S* and *R* now each separately choose an integer *e* where gcd$((p_r\text{-}1)(q_r\text{ -}1),e)=1$ and calculate *d* as e^{-1} modulo $(p_r\text{-}1)(q_r\text{ -}1)$. Thus *S* has constructed a public key pair (e_S, n_r) and private key d_S, and similarly *R* has public key (e_R,n_r) and private key d_R.

Step 2. Elliptic Curve asymmetric key set-up (sessional keys). *S* and *R* jointly choose an elliptic curve *E* over a finite field *GF(p), p* a prime number, and a point P_E on *E*. The prime *p*, the elliptic curve E and the point P_E are the public domain parameters. *S* generates a secret random number K_S in the interval [*1, p-1*] and then performs an elliptic curve scalar multiplication to get the corresponding public key $Q_S = K_S.P_E$. Similarly, *R* generates the key pair (K_R,Q_R). Now K_S and K_R are secret keys for *S* and *R* respectively.

Step 3. Elliptic Curve symmetric key set-up (sessional keys). To generate the symmetric key for use with the AES cipher, R and S use the public keys they generated in Step 2. Each can easily generate $K= K_S.K_R.P_E$ independently without the need to send any additional information over a channel. K is now available to each of them for use as a symmetric key for the AES encryption and decryption process.

Since the EC keys are sessional, K is also a session key.

Notation: We use (K_S^n, Q_S^n) and (K_R^n, Q_R^n) respectively to denote the EC asymmetric session keys for S and R for the n^{th} session.

3.2. Encryption Process

Step1. Asymmetric encryption/decryption for sensitive data P. The process for sending encrypted sensitive data to a receiver uses both the EC and RSA asymmetric keys established in Steps (1) and (2) of A. Recall that the EC keys are sessional and so new keys will be established when a new set of sensitive data is to be sent.

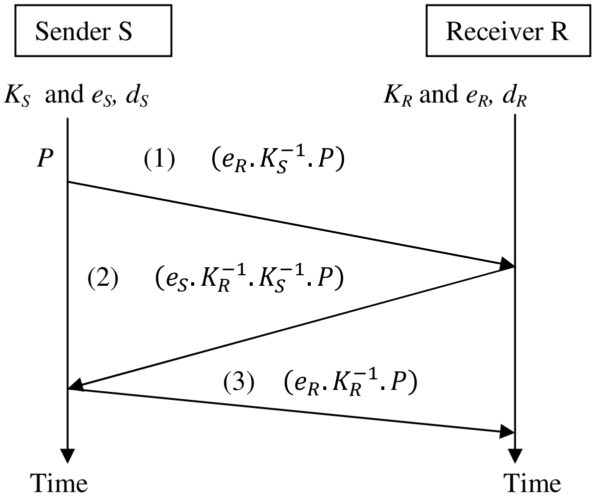


Figure 2. A sender sends sensitive data *P* to a receiver

In Figure 2, K_S , K_R and d_S, d_R are secret, while e_S and e_R are public. To send P to R, S uses R's RSA public key e_R and the inverse of S's secret key K_S^{-1} and sends $e_R.K_S^{-1}.P$. When R receives this, he multiplies by d_R to obtain $d_R(e_R\ K_S^{-1}P) = K_S^{-1}P$. Then, R multiplies this by $e_S\ K_R^{-1}$, and sends $e_S.K_R^{-1}.K_S^{-1}.P$ to S, as shown in step (2) of Figure 2.

Upon receiving this, S calculates $K_R^{-1}P$ by decrypting with S's RSA secret key and EC secret key as follows, $d_S\,(e_S\,K_R^{-1}K_S^{-1}P) = K_R^{-1}K_S^{-1}P$ and $K_S\,K_R^{-1}K_S^{-1}P = K_R^{-1}P$.

Finally, S sends $K_R^{-1}.P$ multiplied by R's public key e_R (as shown in step (3) of Figure 2), $e_R\ K_R^{-1}P$. Upon receiving this, R obtains P by using R's RSA and EC secret keys as follows, $K_R\ d_R\ (e_R\ K_R^{-1}P) = P$.

In each part of the above procedure, the (sessional) secret keys of the participants are required in order to obtain the correct information. In order to verify that what R receives is indeed *P*, *R* now sends $e_S.P$ to *S* who checks $d_S.e_S.P$ and sends a confirm message if *P* is actually obtained.

Step 2. Symmetric encryption/decryption for non-sensitive data M-P. The AES standard for symmetric block ciphers offers variable plaintext block length and key length. The three key lengths available are: 128, 192 and 256; we chose length 128 bits for both the key length and plaintext length in our implementation. The AES encryption/decryption process is shown in http://www.cs.bc.edu/~straubin/cs381-05/blockciphers/rijndael_ingles2004.swf.

Our architecture is now described in Figure 3. The data P is first processed on the sender side by the privacy-enhancing protocol of [12] and the use of sessional asymmetric ECC and RSA encryption keys in the Privacy Enhancement stage.

The sensitive data P is passed into the 'Random ID' box in Figure 3, and the salt algorithm is applied to it. In [12], identifying information is then hashed using a standard hash function such as SHA-1 [26], however, this makes it necessary for S to send R ID information separately over an insecure channel so that R can reconstruct the original IDs by checking hashes against a look-up table. In the current architecture, we eliminate this because R now knows the parameters of the salt algorithm which permits it to reconstruct the IDs.

In the final stage of the privacy-enhancement process, this salted data is sent to the 'Random Grouping' box which also accesses data from P. In this box, the data in P is classified into types such as 'gender', 'postcode' etc., and the data passed to the box from the 'Random ID' box is distributed into files corresponding to its type, but in a random order.

Now S encrypts the contents of the 'Random Grouping' box, again using the sessional ECC and RSA asymmetric keys, while the non-sensitive data M-P is encrypted using the EC sessional symmetric key. The two results are then sent to R across an insecure channel. In Figure 3, the bar above P and above M-P indicates the encrypted versions of P and M-P respectively.

When *R* receives these two pieces of data, it first decrypts both using the respective keys and then reconstitutes the set *P* by reversing the privacy techniques used. To reverse the effects of the 'Random Grouping' box, the attributes defining the types are first identified. The salt algorithm is reversed by using the known salt algorithm and parameters. Thus, both *P* and *M-P* are finally in *R*'s hands.

In the next section, this architecture is compared with several other privacy architectures based on privacy, security and efficiency.

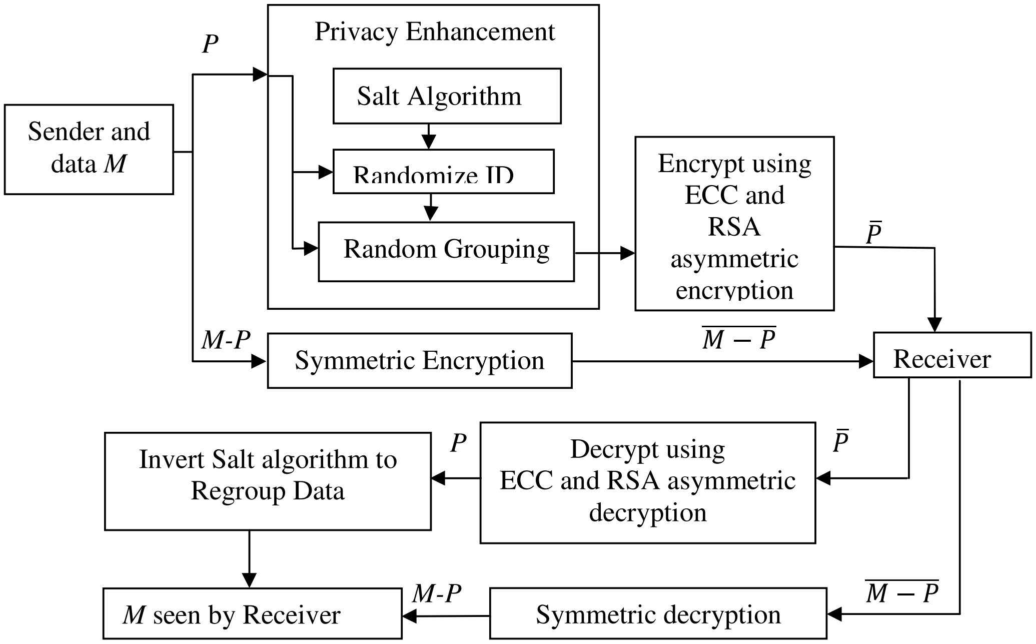


Figure 3. Architecture of proposed privacy and security model

4. Simulation Results of Proposed Architecture and Comparison Study

In this section, we compare a number of solutions in the research literature to the proposed solution. In order to make fair comparisons, we simulated each method based on a common data set but adapted the data format as needed for each method. The basis of comparison was three-fold: privacy, security and efficiency.

The tests were conducted on a machine with Intel E5200 CPU, 4GB RAM and running Fedora 15. All results were averaged over three runs.

4.1. Privacy Comparison

In the tests, four recent, existing solutions [8], [10], [11] and [12] were selected for comparison with the proposed solution; each representing a commonly adopted privacy scheme. The one in [8] implements a generalization technique, [12] presents a random reordering technique, [10] describes a dummy data inserting technique and [11] gives details of a data transformation technique. Test conditions and the environment were the same for all examined solutions, including the same sample data set. We assumed no background-related attacks. The parameters were chosen based on the papers. As the proposed method improves on security aspects of the existing approach of [12], the privacy performance of the two methods was expected to be similar. In the figures PM stands for the proposed method.

The same micro table [27], a matrix containing personal information, is used in all our privacy tests, and it was generated from the experimental data in [8], [10] and [11]. It contains 16 patients (rows) and 5 attributes (columns); two of the attributes are sensitive data and three of them are non-sensitive.

In our evaluation tests, five metrics commonly used in privacy research papers were used: data accuracy, identification probability (ip), overhead, entropy and distribution. Data accuracy measures the changes, or lack thereof, when processing the data. The identification probability is the probability of identifying a particular patient from a given micro table. We assume that the identification probability for the original data is 100%, while for the dummy data it is 0%. The formula used in the calculations is

$$\mathrm{ip} = \left(\mathrm{Math.floor}\left(\left(\sum_{i=1}^{MP}\left(\frac{1}{MP}\times \mathrm{ip_i}\right)\right)\times 100\right)\right)/100 \qquad (1)$$

where Math.floor is the conventional truncating function that cuts the digits after the decimal point; MP is the number of patients and $\mathrm{ip_i}$ is the identification probability for each patient. Overhead is the data added to the original data set, and includes decryption keys, dummy data and so on. More privacy usually entails higher overhead. Entropy in privacy protection measures data uncertainty; higher entropy means that additional background information is needed to identify a particular person. Distribution indicates how much original information has been kept.

Figure 4 shows how the identification probability changes when data accuracy is set at different values. The figure shows that data accuracy in the proposed method and also in that of [12] is always 100% regardless of identification probability, while for [8], the identification probability is rising linearly when data accuracy is increasing. The identification probability result for [10] fluctuates between 18 and 33.5 and data accuracy is stable at around 77%. The approach of [11] encrypts data and so its data accuracy is considered to be 0%. Hence, its result is not drawn in the graph.

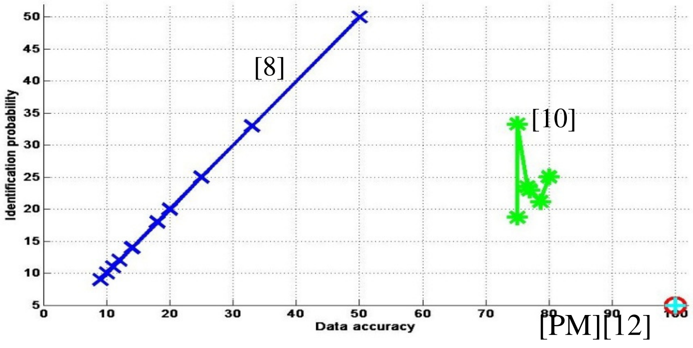


Figure 4. Data accuracy and identification probability

Figure 5 illustrates the trend when the number of patients is increasing. In this test, lower identification probability indicates better performance. The figure shows that the proposed method is able to keep the identification probability under 5% while the methods of [8] and [10] are significantly higher than that. Because of using encryption, [11] has a 0% identification probability in this test.

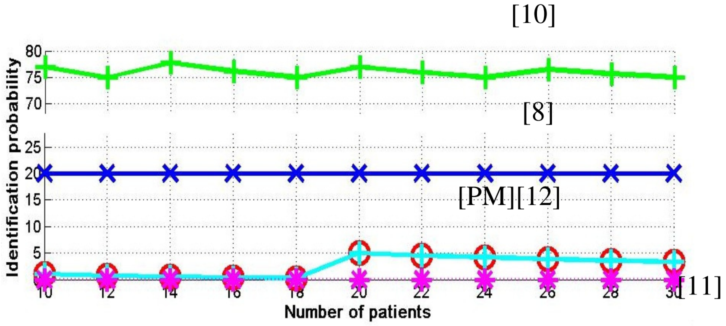


Figure 5. Number of people and identification probability %

Figure 6 shows the low overhead of the proposed solution when compared to the others and illustrates how reducing the probability of identification can increase data overhead. Identification probability (ip) was calculated according to the formula presented earlier, while overhead was measured as the number of additional micro table data entries (units).

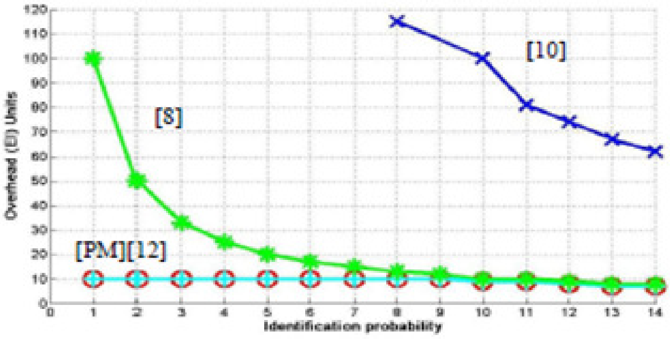


Figure 6. Identification probability % and overhead unit

A common expectation is that better privacy entails higher overhead, and the behavior of methods [8] and [10] are consistent with this. The solution of [10] has a much higher overhead than the others. In case of algorithm [8], the overhead increases sharply when the identification probability drops below 10%, and it gradually decreases when the identification probability increases. However, it is still higher than that of solution [12] and of the proposed method; the latter two can keep the overhead at a constant low level, even when the identification probability is low.

The overhead of [11] is only related to the number of the patients and does not involve identification probability, and thus it is not shown in the figure.

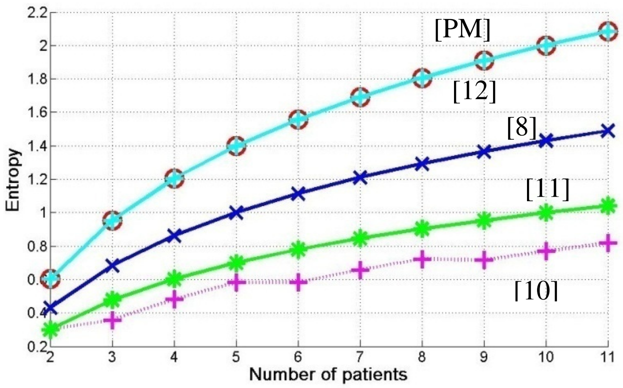


Figure 7. Entropy

Figure 7 shows the entropy of the processed data. Higher entropy gives higher uncertainty about the original data. In the test result, the proposed method is around 40% better than the second best solution.

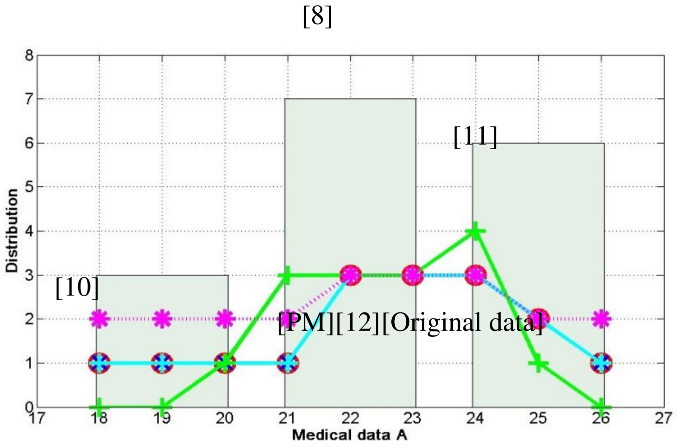


Figure 8. Distribution

Figure 8 shows the data distribution for different approaches. The solution described in [12] and the proposed method [PM] do not change the data. Hence, a single line starting at (18, 1) represents three sets of values: these two solutions and the original data. The approach proposed in [10] approximates the original data quite well, as indicated by the line starting at (18, 2). K-anonymity [8] arranges the data in three groups and the actual values within each group are not known; this is indicated by the three shaded blocks in the figure. The solution described in [11] is also a grouping method, but it maintains the order of data within a group without revealing the real values; the corresponding line starts at (18, 0) in the figure.

4.2. Security Comparison

The architectures presented in [8], [10] and [11], which were discussed in Section IV (A), did not consider the security aspects of electronic data during transmission. Only [12] uses a cryptographic approach to maintaining the privacy level and providing security. In this section, we compare our proposed architecture with that in [12] on the basis of security.

Data encryption. Since a part of the data being sent is considered to be sensitive, encryption is a recommended tool for providing both privacy and security. In our proposed architecture, we therefore encrypt all data before sending it to the receiver. In the interest of efficiency, since encryption carries an overhead cost, we propose the use of two different encryption methods, a cheaper one for non-sensitive data and a more expensive one for sensitive data; in general, the size of the sensitive data set is much smaller than that of the whole set and symmetric encryption is a cheaper solution for large data sets.

Use of session key. If a secret key is captured by an attacker and has not been changed, the attacker may use the key to pretend to be someone else, or to read confidential data. (This is the case in the proposal of [12].) Thus, frequent changes of a secret key are an additional recommendation for security. The EC (asymmetric and symmetric) sessional keys described in Section III allow a user to reserve a single secret key but use it to generate new session keys each time data is to be sent to a receiver. This same session key is used in the salt generation component of the architecture as described in Section III and so needs to be changed each time the sender and receiver transmit and receive data.

4.3. Efficiency Analysis of Proposed Architecture

Experimental setup. The experiment was performed on a 3.20 GHz CPU with i5 processor using the Windows 7, 64-bit operating system. Four tests were run on 256, 512, 1024 and 2048 bits as the total data set size respectively. For each test, we chose sensitive data to be multiples of 64 bits. Table 1 gives the actual numbers used as sizes of *M-P* and of *P*. We used elliptic curve and RSA encryption for the sensitive data *P*, and AES encryption for *M-P*.

In assessing efficiency, we need to consider the two data sets *M* and *P*, and the different encryption algorithms applied to them. An initial approach to preserving data privacy might be to apply the privacy enhancement method to the entire data set and encrypt this with the elliptic curve algorithm. Our aim in this section is to demonstrate that much greater efficiency can be achieved by designating only a part of the data set to be sensitive and separating encryption methods based on this. Hence, in the two tables in this section, we measure performance based on encryption time and speed and separate the data into *P* and *M-P*.

The parameters used to assess performance are:

- *Encryption time (and speed) for the sensitive data set.*
- *Encryption time (and speed) for the non-sensitive data set.*

where the encryption time (CPU time) is considered as the time for the encryption algorithm to produce a cipher text from the original plain text. The speed of encryption is calculated by,
speed of encryption = (size of plain text (bits))/ (encryption time (secs))

The results are discussed in the next section.

Experimental analysis. When applying the architecture proposed in Section III to a set *M* of data, the two different encryption paths are used in parallel. Thus, the total time to run the complete process is the maximum time needed to run either one separately.

From Table 1, we see that AES encryption is always faster than EC encryption, and hence the smaller *P* is chosen to be, the faster we can run the entire protocol; the table shows that the smaller *P* is relative to *M*, the faster the encryption time. Table 2 shows the worst case scenario (most time and slowest speed) where all data is encrypted using the elliptic curve method applied to sensitive data.

The best case scenario occurs when all data is encrypted using AES, however, in this case, no privacy enhancement was applied. From a privacy perspective, the research challenge here is to find the smallest set of sensitive data *P* from a given data set *M* as this will provide greatest efficiency.

Table 1. Encryption speed and time of *p* and m-*p* using a combination of EC and AES

Test No.	**Selected bits (*P*)**	**Encryption time for *P* (secs)**	**Encryption speed (*P*; bits/sec)**	**Unselected bits (*M-P*)**	**Encryption time for *M-P* (secs)**	**Encryption speed (*M-P*; bits/sec)**
1	0	0	0	256	0.00072	355556
	64	0.005	12800	192	0.00048	505263
	128	0.0062	20645	128	0.00028	457143
	256	0.007	36571	0	0	0
2	0	0	0	512	0.0008	640000
	64	0.005	12800	448	0.00077	581818
	128	0.0062	20645	384	0.00074	518919
	256	0.007	36571	256	0.00072	355556
	512	0.0086	59535	0	0	0
3	0	0	0	1024	0.0014	731429
	64	0.005	12800	960	0.0010	960000
	128	0.0062	20645	896	0.0009	995556
	256	0.007	36571	768	0.00089	862921
	512	0.0086	59535	512	0.0008	640000
	1024	0.010	102400	0	0	0
4	0	0	0	2048	0.0029	706207
	64	0.005	12800	1984	0.0025	793600
	128	0.0062	20645	1920	0.0021	914286
	256	0.007	36571	1792	0.00197	909645
	512	0.0086	59535	1536	0.0018	853333
	1024	0.010	102400	1024	0.0014	731429
	2048	0.021	97524	0	0	0

Table 2. Encryption speed and time of *M* using EC

Test No.	**File Size in Bits (*M*)**	**Encryption time for M (secs)**	**Encryption speed (M; bits/sec)**
1	256	0.018	14222
2	512	0.021	24381
3	1024	0.034	30118
4	2048	0.046	44522

5. PROOF OF SECURITY

In this section we analyze the added security provided by our scheme which is missing in [12]. In an online environment, when data is transmitted between two or more parties, then a number of attacks are possible. Here we consider three attacks which are particularly important when sensitive data is being transmitted. These are the Man-In-The-Middle attack [28], the Isomorphic attack [29] and the Denial of Decryption attack [30]. The Man-In-The–Middle attack is designed to allow an attacker to intercept, read, and possibly change information being sent between two parties. The Isomorphic attack is designed to allow an attacker to capture the secret key of a receiver when a certain type of elliptic curve [31] has been chosen. The Denial of Decryption attack is designed to convince a sender to use an incorrect public key or to introduce garbage into the transmission from S to r, thus denying the receiver the opportunity to decrypt the message. We show that these attacks can be used against the architecture described in [12], while none works in our architecture.

5.1. Man-In-The-Middle Attack (Key Establishment)

Often during symmetric key setup a Man-In-The-Middle attack is possible. Here we show how our architecture prevents it.

DEFINITION. Where two parties S and R exchange information over an insecure channel in order to establish a common key, a person can intercept and pretend to be R to S and pretend to be S to R, thus establishing a common key with each separately. The interceptor can then communicate seamlessly with R and S making them believe that they are communicating with each other. This is known as the *'man-in-the-middle' attack for key establishment.*

THEOREM 1. Our architecture is secure from the man-in-the-middle attack during key setup of the symmetric key.

PROOF. In Section III, Part (A), S and R jointly choose an elliptic curve E over a finite field $GF(p)$, p a prime number, and a point P_E on E. The prime p, the elliptic curve E and the point P_E are public domain parameters. S generates a secret random number K_S in the interval [$1, p\text{-}1$] and then performs an elliptic curve scalar multiplication to get the corresponding public key $Q_S = K_S.P_E$. Similarly, R generates the key pair (K_R,Q_R). Now K_S and K_R are secret keys for S and R respectively.

To generate the symmetric key for use with the AES cipher, R and S use their public keys and each easily generate $K= K_S.K_R.P_E$ independently without the need to send any additional information over a channel. Thus, no data needs to be exchanged.

5.2. Man-In-The-Middle Attack (Data Theft)

The use of an ECDH [32] key exchange does not prevent an interceptor from stealing data because it does not authenticate the public key of receiver. Here, however, we show that our architecture prevents this attack.

DEFINITION: Once S sends data to R, an interceptor can pretend to be R and enter into a protocol with S to reveal the sensitive data destined for R. This is known as a man-in-the-middle attack for data theft. The protocol is played out as in Figure 2.

THEOREM 2. Our architecture is secure from the man-in-the-middle attack for data theft.

PROOF. S wants to send sensitive data P to R. As in Figure 2, S firstly sends $\mathrm{e_R. K_S^{-1}. P}$ to R. An attacker A intercepts the message $\mathrm{e_R. K_S^{-1}. P}$ and tries to modify it by randomly choosing an element α and then sending $\alpha.\mathrm{e_R. K_S^{-1}. P}$ encrypted with $\mathrm{e_S}$ to S. Thus S obtains $\alpha.\mathrm{e_R. P}$ by multiplying by $\mathrm{K_S^{-1}}$ and the secret key $\mathrm{d_S}$. S then sends $\alpha.\mathrm{e_R. P}$ to R, which is intercepted by A, from which A gets $\mathrm{e_R P}$. Here, A cannot obtain the sensitive data P due to the fact that it is computationally infeasible to derive P from $\mathrm{e_R P}$.

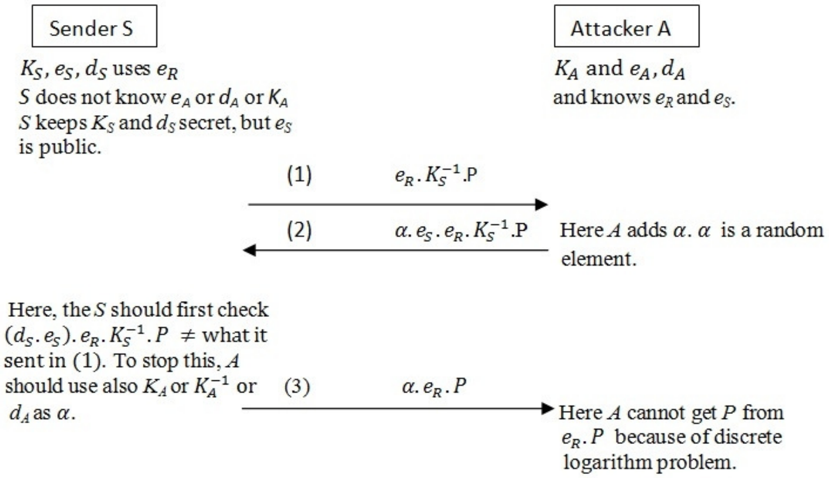

Sender S
Attacker A
K_S, e_S, d_S uses e_R
S does not know e_A or d_A or K_A
S keeps K_S and d_S secret, but e_S is public.
K_A and e_A, d_A and knows e_R and e_S.
(1) $e_R . K_S^{-1}$.P
(2) $\alpha. e_S . e_R. K_S^{-1}$.P
Here A adds α. α is a random element.
Here, the S should first check $(d_S. e_S). e_R. K_S^{-1}. P$ $\neq$ what it sent in (1). To stop this, A should use also K_A or K_A^{-1} or d_A as α.
(3) $\alpha. e_R . P$
Here A cannot get P from $e_R. P$ because of discrete logarithm problem.

Figure 9. Man-in-the middle attack (data theft)

5.3. Isomorphic Attack

When considering this attack, we use the mathematical notations which are described in Section III. In the Isomorphic attack [29] an attacker A wants to get R to sign a message generated by A pretending that it came from S, and use this to obtain R's secret key. We show here how this can be achieved when the elliptic curve E chosen has the special form

$y^2 + axy = x^3$ where, $a \neq 0$
over a finite field GF(p) where p is a prime.

In this attack, A wants R to sign the message $M = (m_x, m_y)$ chosen by A. A obtains the public key, Q_R, of R, and generates a random number u from the set [2, p-1]. A then generates the new message

$$M' = \left(u^2 m_x (\mathrm{mod}\ Q_R), u^3 m_y (\mathrm{mod}\ Q_R)\right) \quad (2)$$

Now, A sends M' to R pretending to be S and asks R to sign it. R signs using its secret key, K_R, obtaining S' = $K_R . M' = K_R . (u^2 m_x (\mathrm{mod}\ Q_R), u^3 m_y (\mathrm{mod}\ Q_R))$ and sends this to A.

Because of the special form of the elliptic curve, it has been shown [31] that the set of points of E is isomorphic to the set of non-zero elements of GF(p). Hence, A can re-compute this equation in GF(p), invert the corresponding value for M' there and calculate K_R, thus obtaining the secret key of R.

THEOREM 3. Our architecture is secure against the Isomorphic attack.

PROOF. Given that the elliptic curve E has the special form

$$y^2 + axy = x^3 \quad (3)$$

over a finite field GF(p), p a prime, A chooses the message M = (m_x, m_y), obtains the EC asymmetric public nth session key, Q_R^n, of R, and generates a random number u from the set [2, p-1]. A then generates the new message

$$M' = \left(u^2 m_x (\mathrm{mod}\ Q_R^n), u^3 m_y (\mathrm{mod}\ Q_R^n)\right) \quad (4)$$

and sends it to R pretending to be S, and asks R to sign it.

R will only sign this message if R has a current session key in use with S. If it does not, it requests A for identification before it runs a new session. In this case, A will be detected. However, if there is still a current session key available between R and S, R signs using its session key, K_R^n, obtaining $S' = K_R^n . M'$ and sends this to A.

As described above, A can now solve this equation for R's sessional EC asymmetric key. However, A is only able to use this key until it expires. If A requests new sensitive data from S it will be required to establish a new session key.

Hence, the Isomorphic attack is not possible in our architecture.

5.4. Denial of Decryption Attack

During elliptic curve encryption/decryption, the sender S wants to send encrypted sensitive data P to the receiver R. S takes R's public key and his identity (or personal information) as input to the encryption function. However, attacker A, has replaced R's public key by someone else's public key (A's own public key or a public key of another person). S is unaware of this replacement and continues to execute the encryption algorithm using R's identity and public key not belonging to R. Although A cannot decrypt the cipher text, neither can R. This is known as a denial of decryption attack on R. While we are not able to prevent such an attack, our architecture can detect it as shown in the next theorem.

THEOREM 4. Our architecture is able to detect a Denial of Decryption Attack.

PROOF. S wants to send sensitive data to R. S sends $e_R.K_s^{-1}.P$ to R but an

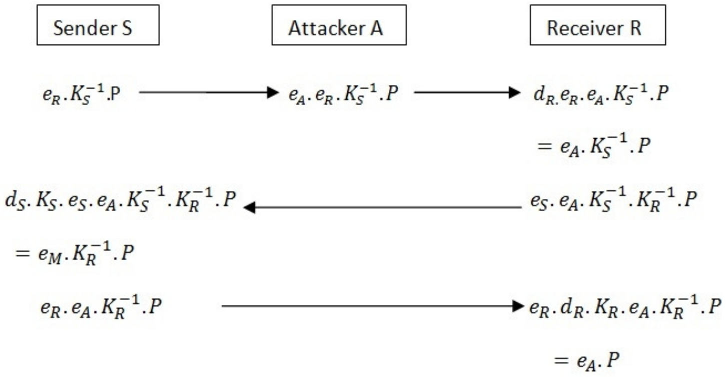


Figure 10. Denial of decryption attack

adversary A intercepts it, includes e_A and then sends $e_A.e_R.K_s^{-1}.P$ to R. Now, R applies her RSA secret key and encrypts the message with the RSA public key of S and sends $e_A.e_S.K_s^{-1}$. $K_R^{-1}.P$ back to S. Now, S calculates $e_A.K_R^{-1}.P$ using his RSA secret key and multiplying by his ECC secret key. Now S uses R's RSA public key and sends $e_A.e_R.K_R^{-1}.P$ to R. So R obtains $e_A.P$ by applying her two secret keys.

However, R cannot make sense of this message and sends $e_A.e_S.P$ to S for verification (If R sends $e_A.P$ to S over an insecure channel, then A can intercept and compute $e_A^{-1}.e_A.P = P$.) If S or R suspects A of an attack, they obtain e_A and compute $e_A.P$, verifying that A was the attacker. However, A can avoid detection by using a random value not his public key. In any case, such an attack is detected in our architecture.

6. Summary and Discussion

We have proposed an efficient and secure privacy-preserving architecture in an online environment. The proposed method addresses efficiency, security and privacy issues for online access of sensitive data. The main results are that the proposed method demonstrates increased speed of encryption/decryption and reduced overhead when compared to other recent privacy-preserving solutions, without compromising on data privacy, accuracy or security of transmissions between sender and receiver. From the privacy aspect, the method proposed here keeps identification probability under control while maintaining data accuracy. On the other hand, to keep identification probability low, usually communication overhead is increased, either because dummy data is added or because of extra parameter processing. However, the overhead of the proposed method depends only on the number of patients and the number of attributes to be protected. The use of selective encryption in the proposed architecture provides both security and efficiency. We also provided proofs of security against some of the classic attacks on transmission of sensitive data. In future work, we plan to develop a prototype of our architecture to handle large data sets as well as complex data such as images. We will also consider the efficiencies available at the receiver by examining resource-constrained (mobile) devices and bandwidth limitations.

Acknowledgements

This project was partially supported by ARC grant LP0989756.

REFERENCES

[1] N. R. Adam & J. C. Worthmann, (1989) “Security control methods for statistical databases: a comparative study”, ACM Computing Surveys, Vol.21, No.4, pp 515-556.

[2] J. J. Kim, (1986) “A method for limiting disclosure in microdata based on random noise and transformation”, American Statistical Association, Proceedings of the Section on Survey Research Methods, pp 303-308.

[3] J. Domingo-Ferrer & J. Mateo-Sanz, (2002) “Practical data-oriented microaggregation for statistical disclosure control”, IEEE Transactions on Knowledge and Data Engineering, Vol. 14, No.1, pp 189-201.

[4] K. Muralidhar & R. Sarathy, (1999) “Security of random data perturbation methods”, ACM Transaction on Database Systems, Vol.24, No. 4, pp 487-493.

[5] P. Kooiman, Leon, Willemborg & J. Gouweleeuw, (1997) “PRAM: a method for disclosure limitation for microdata”, Report, Department of Statistical Methods, Statistical Netherlands, Voorburg.

[6] A. Evfimievski, R. Srikant, R. Agrawal, & J. Gehrke, (2002) “Privacy preserving mining of association rules”, Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 217-228.

[7] C. M. O'Keefe, M. Yung, L. Gu & R. Baxter, (2004) “Privacy-preserving data linkage protocols”, Proceedings of the ACM Workshop on Privacy in the Electronic Society, New York, NY pp 104-102.

[8] L. Sweeney, (2002) “Achieving k-anonymity privacy protection using generalization and suppression”, International Journal on Uncertainty, Fuzziness, and Knowledge-based Systems, Vol. 10, No. 5, pp 571-588.

[9] P. Samarati, (2001) “Protecting respondents identities in microdata release”, IEEE Transactions on Knowledge and Data Engineering, Vol. 13, No. 6, pp 1010-1027.

[10] W. Choi, J. Ryu, W. Kim & U. Kim, (2009) “Simple data transformation method for privacy preserving data re-publication”, IEEE Symposium on Web Society (SWS), pp 209 – 212.

[11] E. Poovammal, M. Ponnavaikko, (2009) “Task independent privacy preserving data mining on medical dataset”, In Proceedings of the International Conference on Advances in Computing, Control, and Telecommunication Technologies (ACT '09), IEEE Computer Society, Washington, DC, USA, pp 814-818.

[12] J. Zhong, V. Mirchandani, P. Bertok, (2010) “Identity protection against data linkage in mHealth”, In proceedings of ATIS 2010, Deakin University, pp 17-24.

[13] K. Singh & S. Samaddar, (2010) “Selective encryption technique in RSA based singular cubic curve with AVK for text based documents: enhancement of koyama approach”, International Conference on Networking and Information Technology (ICNIT 2010) Manila, Philippines, pp 343-349.

[14] C. T. Bhunia, (2006) “Application of avk and selective encryption in improving performance of quantum cryptography and networks”, United Nations Educational Scientific and Cultural Organization and International Atomic Energy Agency, retrieved 10/12/20010, from http://users.ictp.it/ pub off/preprints-sources/2006/IC2006045P.pdf.

[15] R. Oppliger, R. Rytz & T. Holderegger, (2009) “Internet banking: client-side attacks and protection mechanism”, IEEE Journals & Magazines, Volume: 42, Issue: 6, pp 27-33.

[16] ECRYPT II Yearly Report on Algorithms and Keysizes. European Network of Excellence in Cryptology II. ICT-2007-216676. Available at http://www.ecrypt.eu.org/documents/D.SPA.17.pdf.

[17] N. R. Potlapally, S. Ravi, A. Raghunathan & N. K. Jha, (2003) “Analyzing the energy consumption of security protocols”, ISLPED’03, ACM 2003, Seoul, Korea, pp 25–27.

[18] K. Kant, R. Iyer, & P. Mohapatra, (2000) “Architectural impact of secure socket layer on internet servers”, proceedings ICCD’ 00 Proceedings on the 2000 IEEE International Conference on Computer Design: VLSI in Computers & Processors, pp 7-14.

[19] R. Ford & M. Howard, (2009) “Man-in-the-middle attack to the HTTPS protocol”, IEEE Security & Privacy, Volume: 7, Issue: 1, pp 78-81.

[20] K. Cheng, M. Gao, & M. Guo, (2010) “Analysis and research on HTTPS hijacking attacks”, International Conference on Networks Security, Wireless Communications and Trusted Computing, pp 223-226.

[21] I. Blake, G. Seroussi, & N. Smart, (1999) “Elliptic curves in cryptography”, London mathematical society, lecture note series, 265. ISBN 0 521 65374 6 (pbk).

[22] W. Tan, X. Wang, X. Lou & M. Pan, (2011) “Analysis of RSA based on quantitating key security strength”, The Advanced in Control Engineering and Information Science, pp 1340-1344.

[23] D. Osvik, J. Bos, D. Stefan & D. Canright, (2010) "Fast software AES encryption", FSE 2010, LNCS 6147, pp 75–93.

[24] D. Elminaam, H. Kader & M. Hadhoud, (2010) "Evaluating the performance of symmetric encryption algorithms", International Journal of Network Security, Vol.10, No.3, pp 216–222.

[25] P. Leong & C. Tham, (1991) "UNIX password encryption considered insecure", In Proceedings USENIX Winter, pp 269-280.

[26] H. Handschuh, (2005) "SHA family (secure hash algorithm)", Encyclopedia of Cryptography and Security, pp 565-567.

[27] Y. Zhang, (2010) "On data utility in private data publishing", Technical report, Department of Computer Science and System Analysis, Miami University, Oxford, Ohio, Available at http://etd.ohiolink.edu/view.cgi?acc_num=miami1272986770.

[28] Y. Joshi, D. Das, & S. Saha, (2009) "Mitigating man in the middle attack over secure sockets layer", IEEE International Conference on Internet Multimedia Services Architecture and Applications (IMSAA), pp 1-5.

[29] K. Singh & S. Samaddar, (2010) "Different attacks on selective encryption in RSA based singular cubic curve with AVK and their possible solutions", International Conference on Advances in Computer Science (ACS 2010) in Trivandrum, Kerala, India, pp. 88-93.

[30] J. Liu, M. Au & W. Susilo, (2007) "Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standard model", ASIACCS '07 Proceedings of the 2nd ACM symposium on Information, computer and communications security, pp 273-283.

[31] K. Koyama, (1995) "Fast RSA -type schemes based on singular cubic curves y^2+axy≡x^3 mod n", Proceeding in LNCS EUROCYPT '95, Volume - 921.Springer Verlag, pp 329-340.

[32] Microsoft Technet, (2010) "Overview of the ECDH algorithm (CNG example)," Available at http://technet.microsoft.com/zh-cn/library/cc488016.aspx, accessed at 24/03/2010.

Authors

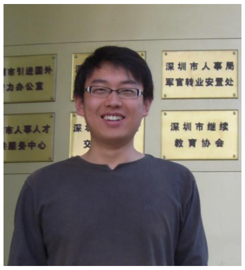
Kalpana Singh

kalpana@deakin.edu.au

Kalpana is pursuing Ph.D at Deakin University in the school of Information Technology. Her academic record is laden with First class throughout. She has been teaching successfully at Department of Computer Science and Information Technology, GLA University INDIA. She has a number of research publications to her credit in reputed journals and conferences in the area of cryptography and Information Security.

Jian Zhong

jian.zhong@rmit.edu.au



Jian is pursuing Ph.D at RMIT University, in the school of Computer Science & Information Technology. He has achieved Masters Degree at Loughborough University (UK). He has number of research publications in reputed international journals and conferences. His research area is information security and privacy protection.

Professor Lynn Batten

lynn.batten@deakin.edu.au

Professor Lynn Batten holds the Deakin Research Chair in Mathematics and is Director of Information Security Research at Deakin University. She is a Fellow of the Australian Computer Society, a Graduate of the Australian institute of Company Directors and a Senior Member of the IEEE. Her research interests cover a broad set of areas in information security from cryptography to malicious software and digital forensics.

A/Prof. Peter Bertok

pbertok@rmit.edu.au

A/Prof. Peter Bertok is working as an Associate Professor at RMIT university, in the school of Computer Science & Infomation Technology. He is also Member of the Distributed Systems and Networking Discipline Group, School of Computer Science and Information Technology at RMIT.