DATA SECURITY ANALYSIS AND SECURITY EXTENSION FOR SMART CARDS USING JAVA CARD

Ms. A A Deshmukh[1], Ms. Manali Dubal[2], Dr. Mahesh TR[3], Mr. C R Chauhan

[1] Professor, Department of Computer Engineering, Sinhgad College of Engineering and Technology, Pune

aadeshmukh@gmail.com

[2] Department of Computer Engineering, Sinhgad College of Engineering and Technology, Pune

manali.dubal@gmail.com

[3] Associate Professor, Department of Computer Engineering, Adhiyamaan College of Engineering, Hosur, Banglore

dr.maheshtr@gmail.com

[4] Department of Computer Engineering, PIET, Limbda

crchauhan2310@gmail.com

ABSTRACT

Smart cards improve the convenience and security of any transaction. They provide tamper-proof storage of user and account identity. Multifunction cards are used to manage network system access, store value and other data. The cards carry personal account, credit and buying-preference information and thus, security becomes a primary issue here. Public Key Cryptography plays an essential role in electronic banking and financial transactions. ECC is one of the best public key techniques for its small key size, high security and is suitable for secure access of smart cards. This article gives principles of public key cryptography, illustrates two cryptographic algorithms RSA and ECC. The elliptic curve cryptography is implemented on smart card using Menezes-Vanstone Elliptic Curve Cryptosystem and Nyberg-Rueppel Signature Scheme [2]. The implementation of these algorithms is done using Java Card technology. The test results are analysed and comparison about the public key sizes and security aspects are also discussed.

KEYWORDS

Smart Card, Public key Cryptosystem, RSA, ECC, ECDSA, ECNRA

1. INTRODUCTION

Today smart cards are used for many different purposes in daily life. The smart card can be a phone card, a card carrying our health insurance information, or an electronic purse. The smart

DOI : 10.5121/ijait.2012.2204

card itself is a device which is able to store data and execute commands. It has its own CPU, memory and COS, almost equivalent to a computer, and it can implement information storage and data processing. With the development of software and hardware technology, smart card is widely used in medical, transportation, communications, finance, and other areas.

As information technology continues to evolve, people increasingly have high demand for information security of smart card. Information security is one of the main directions of smart card; this naturally gives rise to the need for reliable, efficient and convenient cryptographic algorithms which provides Authentication, Confidentiality, Integrity, Non-Repudiation, and Availability. There are several different wireless terminals on the market, with which the consumers can have Internet access.

When these kind of wireless terminals, which need the user to authenticate them before use, are getting more common, the need for some kind of authentication method other than typing in passwords becomes apparent. The smart cards might offer an easier way to identify users. When using a GSM phone, a smart card chip is already hidden inside your handset to provide strong authentication to the network operator.

Smart card is an ideal medium for use with PKI applications. It provides secure storage of confidential data and is capable of executing complex cryptographic algorithms, such as RSA, EIGammal and elliptic curve cryptography (ECC). This paper describes RSA and ECC algorithms, compares these two cryptosystems performance implicated in smart card, and gives improvement proposal and further development.

2. PRINCIPLES OF PUBLIC KEY CRYPTOSYSTEMS

The basic idea that led to public key algorithms was that keys could come in pairs of an encryption and decryption key and that it could be impossible to compute one key given the other. Public key Cryptosystems (PKC) algorithm can be divided into two kinds of public key and private key. In PKC system, public key is open, however private key is kept confidential, and the private key cannot be calculated only from the public key.

Public key algorithms have a big advantage when used for ensuring privacy of communication. Public key algorithms use different keys for signing and decryption, and for encryption and signature verification. The private key may only be known to its owner and must be kept in secret. It may be used for generation of digital signatures or for decrypting private information encrypted with the public key. The public key may be used for verifying digital signatures or for encrypting information. It needs not to be kept secret, because it is infeasible to compute the private key from a given public key. Anyone who informed the user public key is available to encrypt information for secure information exchange with the user. As the public key and private key are different, only the user can decrypt the message, any user who were not authorized and the sender cannot decrypt this message [3]. If Sender A want to send massage m to Receiver B, he calculate cipher text c with the encryption function of ENCeB (m), then transport cipher text to Receiver B. When Receiver B gets massage m' he calculates with the function of DECdB (c) to obtain massage m.

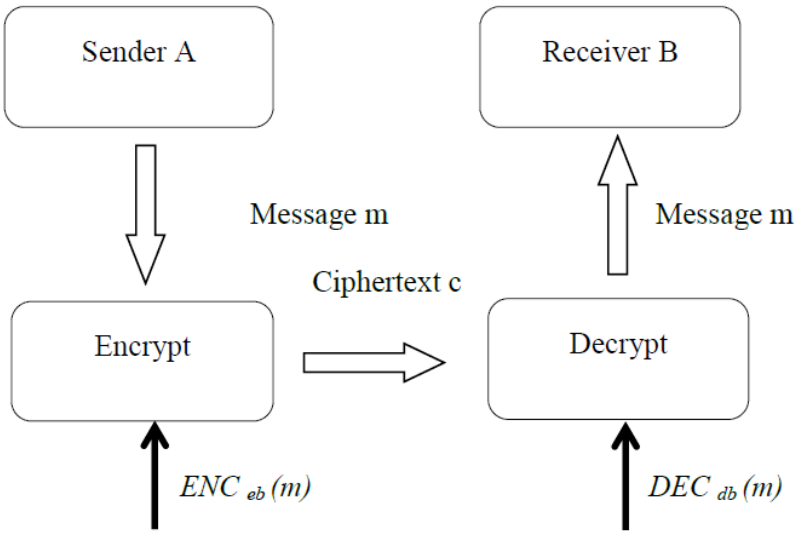

Figure 1. Public Key Cryptosystems

2.1. RSA Cryptosystem

The RSA cryptosystem was invented in 1977 by Rivest, Shamir and Adleman, and was the first realization of Diffie and Hellman's abstract model for public key cryptography [4]. The RSA algorithm is the best known of the integer factorization family of cryptosystems where the strength of the cryptosystem lies in the mathematical difficulty of factoring large integers.
The RSA key pair generation algorithm is generated by following steps:

1. Choose two random primes, p and q, of length l/2.
2. Calculate n=p*q and Φ= (p-1) (q-1).
3. Choose integer e to meet 1<e<Φ and gcd(e, Φ)=1.
4. Calculate integer d to meet 1<d<Φ and e*d=1(mod Φ).
5. Get the public key pair (e,n) and the private key pair (d, n).

Where n is RSA's operation mode, e is signified encryption index, l is signified security parameters and d is signified private key. RSA encryption scheme is get cipher text by the encryption formula c=me mod n, and get the explicitly by the decryption formula m= cd mod n.
The premise behind RSA's security is the assumption that factoring a big number (n into p, and q) is hard. And thus it is difficult to determine Φ (n). Without the knowledge of Φ (n), it would be hard to derive d based on the knowledge of e.

2.2. ECC Cryptosystem

ECC is a public key primitive that is increasingly important as alternative to RSA. ECC was proposed independently by Miller and Koblitz in 1985, is becoming widely known and accepted [8]. Elliptic curves are mathematical constructions, that can be defined over and field. A field is defined by a set of elements and some operations that have some special properties. Order E is the finite field Fp on the elliptic curve, P is a point on the elliptic curve [1]. While using ECC, we deal with various properties of points on curve, and functions. The only aim is to use elliptic curves as an encryption tool which converts the information m into the point P on the curve E. It is defined by formula:

$$y^2 = x^2+ax+b$$

Where a , b ∈ Fp, and satisfies the equation:

$$4a^3+27b^2 \neq 0 \text{ (mod p)}$$

Set the order of P is a prime n, so that assemblage P is cyclic subgroups of elliptic curves which generated by P. Prime P, elliptic curve equation E and order n constitute a public set of parameters. The whole ECC algorithm works on the principle of generating the secret key from the random generated integer and the key pair obtained by once adding the multiplying the elliptic curve points [1]. The ECC key pair generation algorithm is generated by following steps:

1. Choose a random key d in [1,n-1].
2. Calculate Q=d*p.
3. Get the public key pair (Q,d).

Where d is signified private key and Q is signified public key. To achieve the elliptic curve encryption, following steps need to do. Express plaintext m as elliptic curve point M.

1. Choose a random key k in [1, n-1]
2. Calculate C1=k*p
3. Calculate C2=M+k*Q
4. Get the public key pair (C1, C2). ; where C1 and C2 are ciphertexts.

The decryption process is receiver calculate M by formula :

M= C2+d* C1 ; where M is the plaintext.

One of the advantages of ECC is that the elliptic curve discrete logarithm problem is believed to be harder than both the integer factorization problem and discrete logarithm problem modulo p. This extra difficulty implies that ECC is one of the strongest public key cryptographic systems known today [6].

2.2.1 Menezes-Vanstone Elliptic Curve Cryptosystem

The Menezes-Vanstone elliptic curve cryptosystem is defined as follows [10]. Let E be an elliptic curve defined over Z_p (p > 3 prime), or in GF (p_n) with n > 1, such that E contains a cyclic subgroup H in which the discrete logarithm problem is intractable.

Let, $P = Z_p * Z_p$, $C = E * Z_p * Z_p$, and define,

$$K = \{ (E, \alpha, \alpha, \beta) : \beta = \alpha\alpha \},$$

where α ∈ E. The values α and β are public and α is secret.

For K = (E, α, α, β), for a (secret) random number $k \in Z_{|H|}$ and for $x = (x_1 , x_2) \in Z_p * Z_p$, define

$$e_k(x , k) = (y_0 , y_1 , y_2)$$

where,

$$y_0 = k \alpha,$$
$$(c_1, c_2) = k \beta,$$
$$y_1 = c1x1 \bmod p,$$

$$y_2 = c_2 x_2 \bmod p.$$

For a ciphertext $y = (y_0, y_1, y_2)$, define,

$$d_K(y) = (y_1 c_1^{-1} \bmod p,\ y_2 c_2^{-1} \bmod p)$$

where, $ay_0 = (c_1, c_2)$.

The Menezes-Vanstone cryptosystem is a more efficient variation of the well-known ElGamal cryptosystem [11]. In the Menezes-Vanstone variation of the ElGamal cryptosystem an elliptic curve is used for masking, and plaintexts and ciphertexts are allowed to be arbitrary ordered pairs of (nonzero) field elements (i.e. they are not required to be points on E). This yields a message expansion factor of two, the same as original ElGamal cryptosystem.

2.2.2 Nyberg-Rueppel Signature Scheme (ECNRA)

The Nyberg-Rueppel signature scheme can be defined as follows.

Let E be an elliptic curve defined over Z_p ($p > 3$ prime) such that E contains a cyclic subgroup H in which the discrete logarithm problem is intractable.

Let, $P = Z_p * Z_p$, $C = E * Z_p * Z_p$, and define,

$$K = \{ (E, \alpha, \alpha, \beta) : \beta = \alpha\alpha \},$$

where $\alpha \in E$. The values α and β are public and α is secret.
For $K = (E, \alpha, \alpha, \beta)$, for a (secret) random number $k \in Z_{|H|}$ and for $x = (x_1, x_2) \in Z_p * Z_p$, define

$$sig_K(x, k) = (c, d)$$

where,

$$(y_1, y_2) = k\alpha$$
$$c = y_1 + hash(x) \bmod p$$
$$d = k - ac \bmod p$$
$$ver_K(x, c, d) = true \leftrightarrow hash(x) = e,$$

where

$$(y_1, y_2) = d\alpha + c\beta$$
$$e = c - y_1 \bmod p$$

The reason to use the hash algorithm is to make it impossible to and a match between the real input and some majorly changed version that would give the same hash value. The problem is considered exceptionally difficult to solve with the above hash algorithms [9]. The message expansion can be reduced by using point compression. That is, the y-coordinate of the point can be recovered given its x-coordinate and a single bit of extra information.

3. CONCLUSIONS

When comparing public key cryptographic systems, there are three distinct factors to take into account:

1. Security: What is the security based on. How long has the cryptosystem been in wide use and how much its security has been studied.
2. Efficiency: How much computation is required to perform the public key and private key transformations. How many bits must be communicated to transfer an encrypted message or signature.

3. Space requirements: How many bits are required to store the key pairs and associated system parameters.

3.1 Security

RSA and ECC are two public key algorithms. We can comparison these two cryptographic algorithms in key size. As is shown in Table 1, if security level is given, ECC has a small parameter than RSA. The higher is the level of security, the gap of parameters size more obvious. Smaller parameter will make computing faster, shorter keys and smaller key certificates, the computation speed of ECC is many times faster than the RSA. For example, RSA key size is 1024 bit and ECC key size is 160 bit at the 80 security level, their key size ratio is 1 to 6. When the security level is raised to 256, their key size ratio increases from 1 to 30.

Table 1. Key size Comparisons (security)

RSA Key size (in bits)	ECC Key size (in bits)	RSA / ECC Key size ratio
512	106	5 : 1
768	132	6 : 1
1024	160	7 : 1
2048	210	10 : 1
21000	600	35 : 1

It is found that to achieve reasonable security; RSA would need to employ a 1024-bit modulus, whereas a 160-bit modulus should be sufficient for the ECC. ECC required a smaller modulus than RSA and that the security gap between the systems grew as the key size increased. For example, 300-bit ECC is significantly more secure than 2000-bit RSA.

Another way to look at this security issue is to compare the equivalent strength of RSA keys and ECC keys for smart card applications. The Table 1 shows that in smart card applications requiring higher levels of security, ECC is able to offer security without a great deal of additional system resources.

3.2 Efficiency

In both RSA and ECC, considerable computational savings can be made. In RSA, a short public exponent can be employed to speed up signature verification and encryption. In ECC, a large proportion of the signature generation and encrypting transformations can be precomputed [7]. Also, various special bases for the finite field F_2^m can be employed to perform the modular arithmetic involved in ECC operation more quickly. Certicom has performed the tests using 167-MHz UltraSparc running Solaris 2.5.1.

It can be found from the Table 2, ECC is an order of magnitude (roughly 10 times) faster than RSA [12]. The use of a short public exponent in RSA can make RSA encryption and signature verification timings comparable with timings for these processes using the ECC.

Table 2. Benchmarks for Solaris (Efficiency) [13]

Function	163-bit ECC (ms)	1024-bit RSA (ms)
Key pair Generation	3.8	4708.3
Sign	2.1 (ECNRA) 3.0 (ECDSA)	228.4
Verification	9.9 (ECNRA) 10.7 (ECDSA)	12.7
Diffie – Hellman Key Exchange	7.3	1654.0

3.3 Space Requirements

Elliptic curve cryptosystems have the potential to provide security equivalent to that of existing public key schemes, but with shorter key lengths. Having short key lengths is a factor that can be crucial in some applications, for example, the design of smart card systems. The arithmetic processor on a smart card is restricted in size to an area of roughly 25 mm^2. An RSA chip designed to do modular multiplication of 512-bit numbers has about 50,000 transistors, while a chip designed to perform arithmetic in the field F_2^{593} has about 100,000 transistors [13]. By comparison, a chip designed to do arithmetic in F_2^m where m is about 200, would have less than 15,000 transistors, and would occupy about 15% of the 25 mm^2 area assigned for the processor. Another advantage to be gained by using elliptic curves is that each user may select a different curve E, even though all users use the same underlying field K. Table 3 is from Certicom, and compares the size of the system parameters and selected key pairs for the different systems, and presents evidence that the system parameters and key pairs are shorter for the 160-bit ECC than for 1024-bit RSA.

Table 3. Space requirements

	System Parameters (bits)	Public Key (bits)	Private Key (bits)
1024-bit RSA	n / a	1088	2048
160-bit ECC	481	161	160

Both of the systems have similar bandwidth requirements when they are used to encrypt or sign long messages, but say this situation changes for the case where short messages are being transformed. The encryption algorithm used in encrypting 100-bit message is an ElGamal variant with point compression. Therefore it would appear from this comparison that ECC offers considerable bandwidth savings over the RSA when being used to transform short messages [9]. In summary, ECC provides greater efficiency than either integer factorization systems, in terms of computational overheads, key sizes and bandwidth. In implementations, these savings mean higher speeds, lower power consumption, and code size reductions. These benefits make ECC widespread used.

4. IMPLEMENTATION

At present, smart card is mainly used for electronic identification and storing user information. The security services offered by a smart card often include both data encryption and public key operations. Creation of a digital signature is often the most computationally intensive operation demanded of a smart card. The hardware resources of smart card are limited; security system is facing the constraints of memory capacity and computing power. ECC encryption is capable of compensate for the limitations of the smart card hardware. On the one hand, the key generated from ECC is short, which means less storage capacity, faster information transfer rate and computing power can be achieved. On the other hand, the use of ECC in the smart card does not require additional hardware, thereby reducing the cost of hardware and improving the usability. To implement an ECC, an implementer must select a finite field in which to perform arithmetic calculations. Elliptic curve selection and parameter determination may make difference results, how to improve the efficiency of the cryptosystem becomes the focus of researchers.

The Elliptic curve cryptosystem and RSA is implemented using Microsystems' Java Card emulator. There are three different bases, which can be used to implement fields of characteristic two: polynomial base, normal base, and subfield base. We have implemented both the polynomial base and normal base fields on Java Card.

4.1 Java Card

The Java Card platform allows the on-card application to be written in Java. This brings the main advantages of Java to on-card software development. In addition, it provides a good basis for multi-application cards, where on the same card more than one application is supported. Many of the disadvantages of Java also tag along, unfortunately, like inefficiency and clumsiness of doing unsigned 16-bit arithmetic [4]. The on-card executable code consists of byte codes that are interpreted by the Java Card Runtime Environment, which controls the execution of the different applications while making sure that these applications do not interfere [3]. The goal is that Java Card applets can be run in any Java Card. This goal is not fully achieved yet because current implementations still differ slightly from the present specification and from each other.

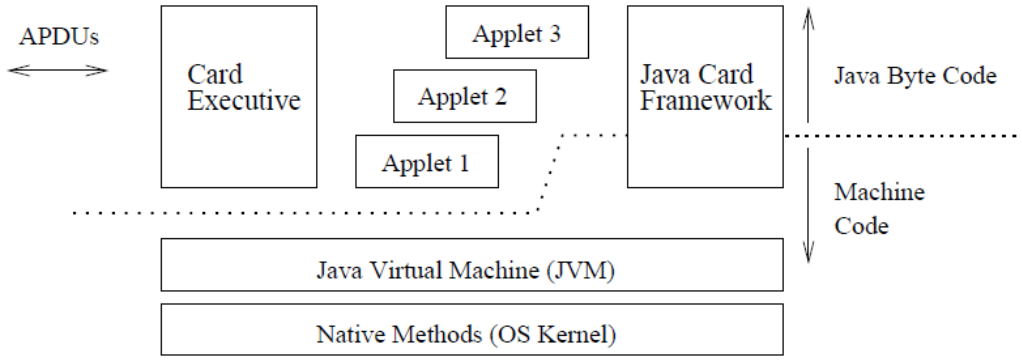

Figure 2. Software Stack of Java card

The software stack of a Java Card is shown in Figure 2. The Java Card Runtime Environment (JCRE) has the following interfaces: The Card Executive manages the card and is the communication link between the card applet and the off-card code. The Java Virtual Machine (JVM) executes the bytecode of the applet and of the library functions it uses. The Java Card Framework provides the library functions [4]. They form the standard Java Card API.

4.2 Structure of Implementation

The implementation has been divided into three separate packages: RSA, PB_ECC, and ONB_ECC. The RSA package contains all the classes needed to implement the RSA cryptosystem, the PB_ECC package contains the classes needed to implement polynomial base elliptic curve cryptosystem, and respectively ONB_ECC package contains classes needed to implement optimal normal base elliptic curve cryptosystem.

As can be seen from the Figure 3 the only difference between the PB_ECC and ONB_ECC packages is one class. PB_ECC uses the PolyField class and ONB_ECC uses the ONBField class.

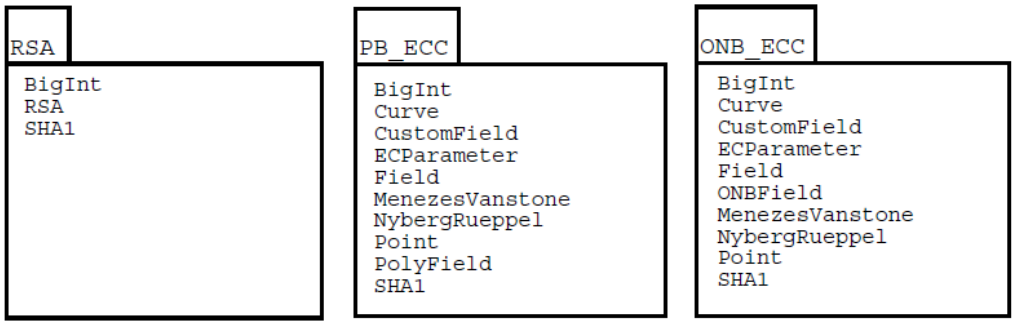

Figure 3. Structure of Implementation

4.2.1 RSA Implementation

The RSA implementation consists of the RSA package. The package contains a class named RSA and the previously described classes BigInt and SHA1. The „RSA" class implements the RSA encryption algorithm and RSA signature scheme. In addition it contains CRT (Chinese Remainder Theorem) variations of the algorithms (crtDecrypt, crtSign).

All the methods in the RSA class use BigInts. In addition, the RSA signature scheme and the CRT variation of that scheme use SHA1 to compute SHA-1 message digests. The public methods of RSA are shown in Figure 4.

As described earlier, the RSA encryption and signature verification can be speeded up significantly by selecting a small public exponent b. Another way to speed up RSA is to use Chinese Remainder Theorem (CRT) as described in RSA Cryptography Standard. In this method the RSA private key consists of a quintuple (p; q; dP; dQ; qInv), where the components have the following meaning: p is the first factor, q is the second factor, dP is the first factor's exponent, dQ is the second factor's exponent and qInv is the CRT coefficient [17]. All components are nonnegative integers.

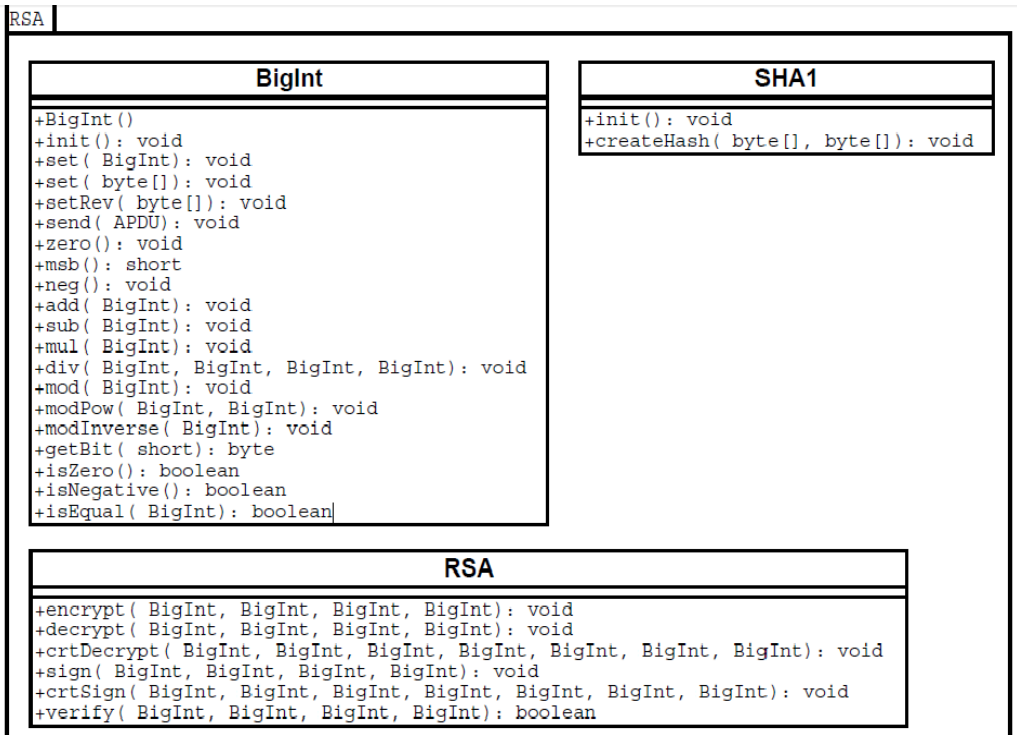

Figure 4. Class structure of RSA implementation

4.2.1.1 RSA Encryption

In a valid RSA private key the two factors p and q are the prime factors of the modulus n, the exponents dP and dQ are positive integers less than p and q respectively, b is the public exponent, satisfying,

$$b * dP \equiv 1 \pmod{(p-1)}$$
$$b * dQ \equiv 1 \pmod{(q-1)}$$

and CRT coefficient qInv is a positive integer less than p, satisfying

$$\text{Theorem is } q * qInv \equiv 1 \pmod{p}$$

4.2.1.2 RSA Decryption

The RSA decryption using Chinese Remainder defined as follows. c is a ciphertext representative, an integer between 0 and n-1. m is a message representative, an integer between 0 and n-1.

$$m_1 = c^{dP} \bmod p$$
$$m_2 = c^{dQ} \bmod q$$
$$h = qInv\ (m_1 - m_2) \bmod p$$
$$m = m_2 + h * q$$

4.2.2 Implementation of Elliptic Curve Cryptosystem

There are two main representation of the field F_{2n}, polynomial representation and normal base representation. In polynomial base representation the binary multipliers are written from the highest power to the lowest. The normal base is handled respectively, the multipliers are listed from the most significant to the least significant. The point (0 , 0) is selected to be the point at infinity because it is never on the curve.

The elliptic curve operations require addition, multiplication, squaring and inversion in the underlying field. The inversion operation is by far the most expensive. The elliptic curve cryptosystem implementation consists of PB_ECC and ONB_ECC packages. These both packages are very similar, the only difference between them is that because the PB_ECC uses polynomial base fields it has a PolyField class, and because ONB_ECC uses optimal normal base fields it has an ONBField class.

The optimal normal base implementation consists of the ONB_ECC package. The ONB_ECC package contains the ONBField class and the previously described classes Curve, CustomField, ECParameter, Field, Menezes Vanstone, Nyberg Rueppel, and Point [11]. The implementation of normal basis arithmetic is quite simple, only bitwise and, bitwise exclusive-or, and shift operations are needed. The fact that these are the fastest operations possible on any microprocessor makes optimal normal base (ONB) attractive. Squaring a normal base number amounts to a rotation. Addition is simply an exclusive-or operation [6].

The inversion uses Inverse algorithm and the basics of multiplication are the same in any mathematical system, just multiply coefficients and sum over all those that have the same power. The optimal normal base implementation uses a precomputed lambda matrix to speed up the multiplication. The lambda vector for Type I ONB stores all the values of j for each value of i that satisfies the equation
$2i+2j = 1 \bmod (m+1)$.
The lambda matrix for Type II ONB is built by working with group of four equations. To build the lambda matrix, we find solutions to

$$2i + 2j = 1$$
$$2i + 2j = -1$$
$$2i - 2j = 1$$
$$2i - 2j = -1$$

The operation for field addition is implemented in the Field class. The rest of the operations multiplication, squaring and inversion) needed in elliptic curve cryptosystem for optimal normal base fields are implemented in the ONBField class. ONBField class implements optimal normal base fields over F_2^n.

The genLambda method together with the initTwo method creates the lambda vectors described above. The field multiplication is implemented in the mul method, which uses the precomputed lambda vectors. Squaring a field is implemented in the square method. The inv method computes the inversion of a field using inverse algorithm [6] [14].

The public methods in ONBField are shown in Figure 5.

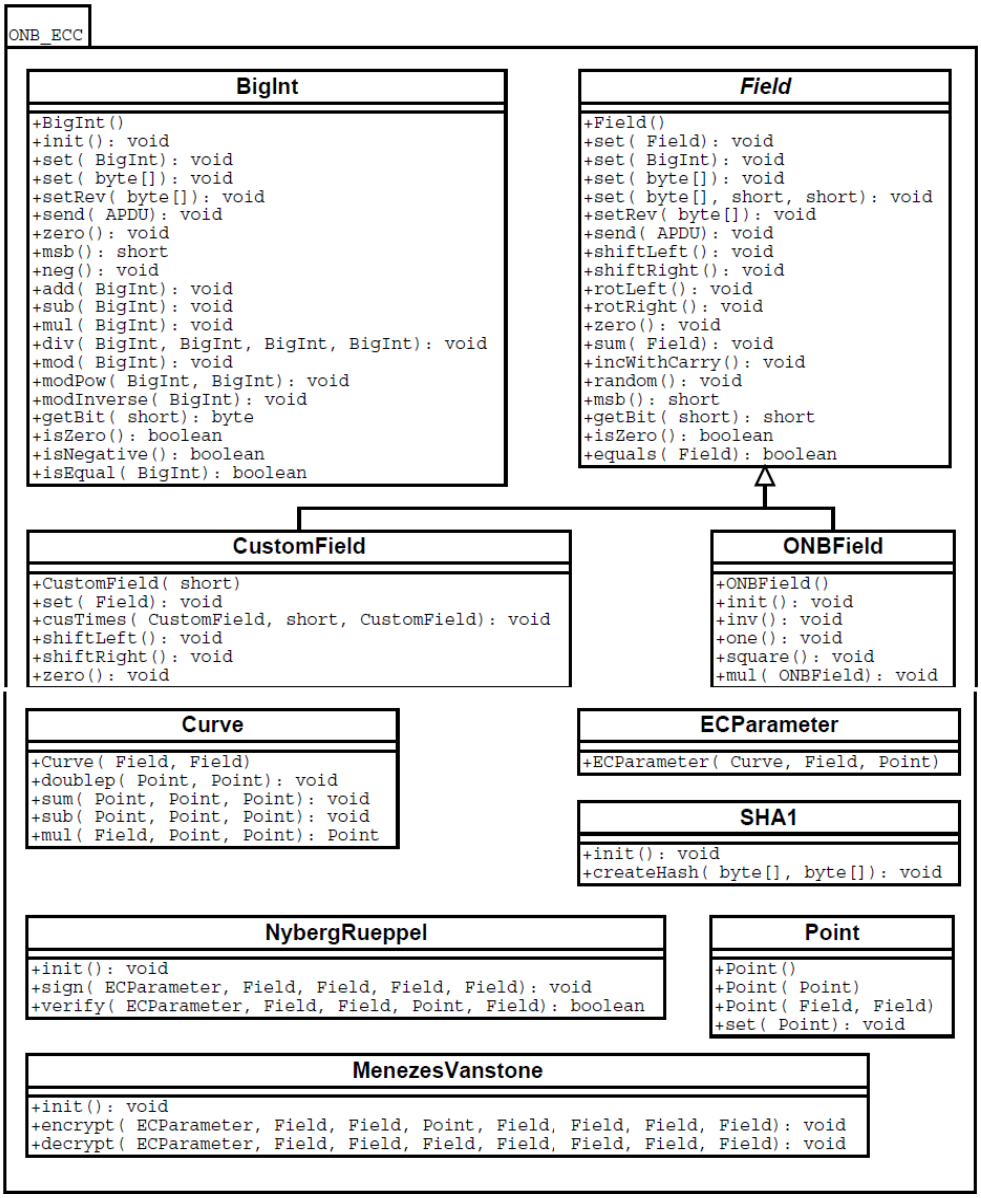

Figure 5. Class structure for optimal normal base ECC implementation

5. IMPLEMENTATION RESULTS

This section explains the test results of the RSA and elliptic curve cryptosystem implementations. The test results contain the lower and upper limits of 95% confidence interval calculated using the T-distribution [8].

5.1 RSA Implementation results

Table 4. RSA Cryptosystem's test results

RSA key	Encryption (ms)	Decryption (ms)	Signing (ms)	Signature Verification (ms)
1024-bit	45.8 ± 2.9	80427.9 ± 5.2	80271.1 ± 2.8	213.6 ± 2.7
2048-bit	123.0 ± 10.2	622099.1 ± 260.8	621918.8 ± 366.3	827.4 ± 24.8

5.2 ECC Implementation results

Table 5. Polynomial Base ECC test results

Elliptic Curve	Encryption (ms)	Decryption (ms)	Signing (ms)	Signature Verification (ms)
163-bit	4026.4 ± 4.5	1951.8 ± 3.8	2009.3 ± 0.3	4031.2 ± 0.8
193-bit	6708.4 ± 2.1	3302.1 ± 0.3	3353.7 ± 0.3	6445.8 ± 0.6
233-bit	11617.9 ± 13.0	5844.6 ± 13.1	5797.5 ± 1.0	11477.8 ± 2.8

Table 6. Optical Normal Base ECC test results (using ECNRA and ECDSA)

Elliptic Curve	Encryption (ms)	Decryption (ms)	Signing (ms)	Signature Verification (ms)
158-bit	1613.0 ± 0.3	830.1± 0.3	806.0 ± 0.3	1569.7 ± 0.5
194-bit	3006.3 ± 1.6	1415.2 ± 0.4	1502.5 ± 0.3	2700.1 ± 4.5
209-bit	3730.3 ± 0.6	1814.2 ± 0.4	1863.0 ± 1.2	3492.2 ± 5.0

6. PERFORMANCE ANALYSIS

6.1 Cryptographic cost of the protocol

Computation cost and communication cost are the most important aspects of password authentication protocols which affect the overall performance. The performance of the protocol relies directly on the asymmetric encryption load/cost and on the Smart Cards resources and capacity. In order to evaluate the performance, we studied the cryptographic cost of protocol on two 32 Javacards. Smartcard A and B have 2304 bytes RAM, 96 Kbytes ROM, 32 Kbytes EEPROM and 10 MHZ Maximum clock (card A), 8 MHZ Maximum clock (card B). The estimated cryptographic cost at the client level is about 250 ms to calculate the key encryption using card A (600 ms using card B) [4]. On the other hand, the cryptographic computation costs about 420 ms at the server level using card A (900 ms using card B) [4][12].

This reveals that our enhanced scheme manages to reduce the processing time of cryptographic loads to open authenticate and secure sessions for entities especially if we compare it with other mechanisms such as certificate based protocols. In certificate-based protocols, each certificate verification takes about 2 seconds if the certificates of entities had to be directly signed by the Certificate Authority (CA) root. This cost hardly increases in parallel with certificate chain increase.

6.2 Comparison of RSA, ECC, ECDSA on Java Card platform

Encryption with 1024-bit RSA is always faster than encryption with 158-, 163- or 174-bit ECC. The greatest difference appeared when encryption was made using 1024-bit RSA with public exponent 3, and then RSA encryption was 88 times faster than encryption with 163-bit polynomial base ECC [17].

Signature verification with 1024-bit RSA was also always faster than with 158-, 163- or 174-bit ECC. The greatest difference appeared when signature verification was made using 1024-bit RSA with public exponent 3, and then RSA signature verification was 19 times faster than signature verification with 163-bit polynomial base ECC [1]. The smallest difference occurred when RSA's public exponent was 2^{16+1} then RSA signature verification was only 2 times faster than signature verification with 158-bit optimal normal base ECC [15].

On the other hand, decryption with 158-, 163- or 174-bit ECC was always faster than with 1024-bit RSA. The greatest difference appeared when decryption was made using 158-bit optimal normal base ECC then ECC decryption was 97 times faster than decryption with 1024-bit RSA using public exponent 3 [11]. The smallest difference occurred with 163-bit polynomial base ECC then ECC decryption was 22 times faster than decryption with 1024-bit RSA using CRT and public exponent 3 [15].

6.3 Graphical Analysis

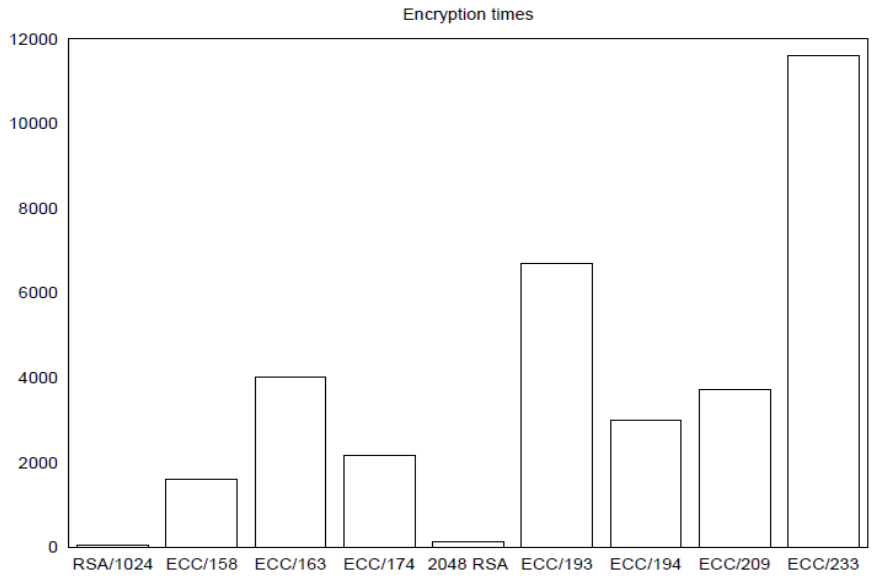

Figure 6. Encryption time (in milliseconds)

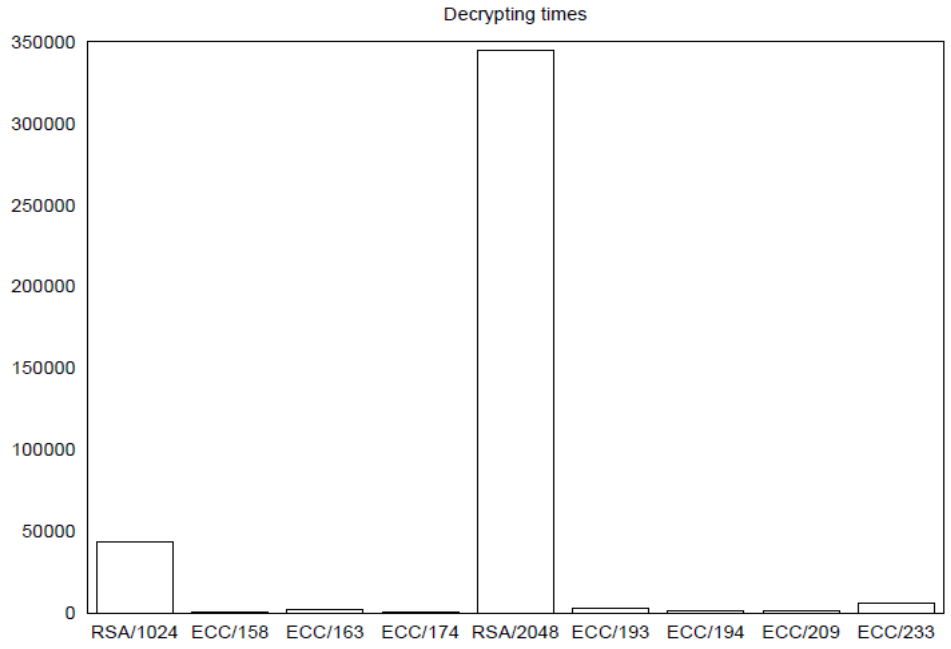

Figure 7. Decryption time (in milliseconds)

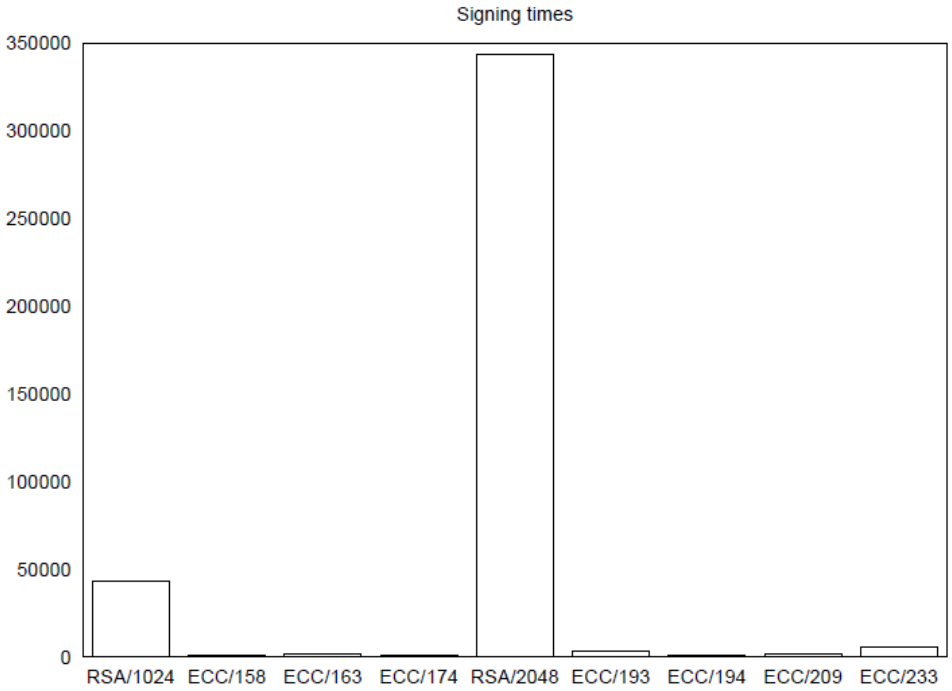

Figure 8. Signing time (in milliseconds)

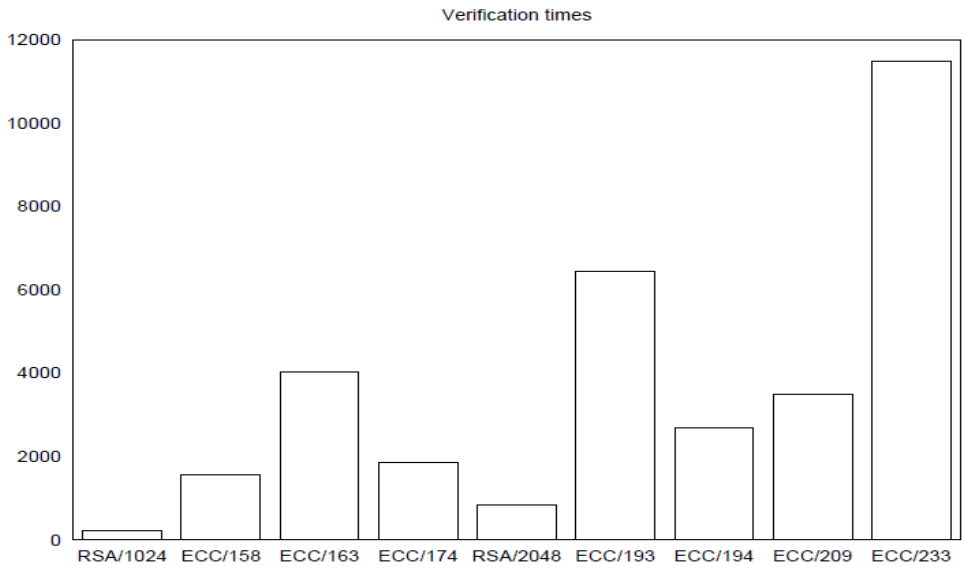

Figure 9. Signature Verification time (in milliseconds)

7. CONCLUSION AND OUTLOOK

We have described an authentication and key agreement protocol for smart card communication based on elliptic curve cryptographic techniques. The proposed protocol is a public key type with the feature of signature generation procedure. The new protocols are based on previous classic authentication protocols, including the protection of integrity and session key exchange. This can be used to provide the integrity of the data being transferred during the authentication process in order to prevent from active attacks.

The protocol is based on the Elliptic Curve Digital Signature Algorithm (ECDSA), and inherits the security and implementation properties of the elliptic curve cryptosystems, which seem to offer the highest cryptographic strength per bit among all existing public-key cryptosystems. With a 160-bit modulus, an elliptic curve system seems to offer the same level of cryptographic security as DSA or RSA with 1024-bit moduli. The smaller key sizes result in smaller system parameters, smaller public-key signatures, bandwidth savings, faster implementations, lower power requirements, and smaller hardware processors.

ACKNOWLEDGEMENTS

The authors are grateful to the principal and management of respective colleges for all the facilities and constant encouragement for carrying out this research work. Also they heartily thank the AIRCC committee and the ITCS conference committee for giving a profound platforms and an opportunity to present the paper.

REFERENCES

[1] M.Badra and P.Urien, “Introducing SmartCards to Remote Authenticate Passwords using Public Key Encryption”, 7803-8219-6/04, IEEE 2004

[2] Sandra Kay Miller "Facing the challenge of wireless security" Technology news July 2001.
[3] IEEE P1363. Standard specifications for public key cryptography. Draft version 7, September 1998.
[4] Mohammad Abdul Azim and Abbas Jamalipour." An Efficient Elliptic Curve Cryptography based Authenticated Key Agreement Protocol for Wireless LAN Security",7803-8924-7/05, 2005 IEEE
[5] P.E. Abi-char, A.Mhamed, B. El Hassan, "A Secure Authenticated Key Agreement Protocol Based on Elliptic Curve Cryptography", International Symposium on Information Assurance and Security, Vol.57, IEEE 2007, pp.89-94.
[6] K.Lauter, "The Advantages of Elliptic Curve Cryptography for Wireless Security", IEEE Wireless communications Magazine, February 2004, (1536- 1284/04)
[7] K. Araki, S. Miura, and T. Satoh. Overview of elliptic curve cryptography. In International Workshop on Practice and Theory in Public Key Cryptography, pages 1-14, 1998.
[8] J. Buchmann and V. Muller. Computing the number of points of elliptic curves over finite felds, July 1991.presented at International Symposium on Symbolic and Algebraic Computation, Bonn.
[9] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21:120-126, 1978.
[10] Bruce Schneier. Applied Cryptography. John Wiley and Sons, Inc., 1994.
[11] Bruce Schneier et al. Elliptic curve public-key cryptography. Crypto-Gram, November 1999.
[12] Certicom Corp. ECC Standards. http://www.certicom.com/research/ECC standards.html.
[13] Certicom Corp. Current Public-Key Cryptographic Systems, April 1997. http://www.certicom.com/research/wecc2.html.
[14] Diffie,W.and Hellman M., "New directions in Cryptography", IEEE Transactions on Information Theory, 1976 22(6), pp:644-654.
[15] Johan Borst,Bart Preneel,Vincent Rijmen,"Cryptography on smart cards",Computer Networks, 200136,pp:423-435.
[16] Dr.R.Shanmugalakshmi and M.Prabu, "Research Issues on Elliptic Curve Cryptography and Its applications", International Journal of Computer Science and Network Security, 2009 9(6),pp:19-22.
[17] Lercier R, et al,"Counting the number of points on elliptic curves over finite fields:Strategies and performances", Eurocrypto 1995,LNCS 2045,pp:79.

Authors

1). Prof. (Ms.) Aaradhana Arvind Deshmukh obtained Masters [Computer Engineering] from Pune University . She has also obtained various degrees like A.M.I.E. Computer Engineering, B.E. (Computer Engineering) M.A. (Economics) from Pune University. She is having 10 years' experience in Teaching Profession and 2 ½ years R & D experience in various institutes under Pune University. She has published 33 papers, 8 in International Journals like ACM, IJCSI, ICFCA,IJCA etc, 16 in International Conferences like IEEE etc., 5 in National Conferences, 4 in symposiums . She has received Gold Medal at International level Paper Presentation on "Neural Network" as well as one more for "UWB Technology based adhoc network", in International Conferences. She is recipient of 'Distinguished Alumni Award' in 2011 from Inst. Of Engineers [India] , Gunawant Nagrik Puraskar for the year 2004 – 2005, 'Anushka Purskar' from Pimpri Chinchwad Municipal Corporation, and also won many Firodiya awards. She has organized many 15 multidisciplinary Short Term Training Program, workshops, conferences on National and International Level.

2). Ms. Manali Dubal is a pursuing M.E in Computer Science from Pune University. She has published 2 papers in International Journals and 1 in International Conference. Her areas of interests include Cryptography and Network Security, Biometrics and Information systems.

3). Dr. Mahesh TR is an Associate Professor in Adhiyamaan College of Engg., and Tech., Hosur, Banglore. He has completed his Ph.D program in Computer Science. His areas of interests are Data mining and Warehousing, Cryptography implementation on Java.

4). Mr. CR Chauhan is pursuing M.E in Computer Science from PIET, Limbda. His area of interests includes cryptography and network security, data warehousing and mining.