Performance Evaluation of RSA Variants and Elliptic Curve Cryptography on Handheld Devices

Jagdish Bhatta and Lok Prakash Pandey

Central Department of Computer Science and IT, Tribhuwan University, Nepal
Nagarjuna College of Information Technology, Tribhuvan University, Nepal

Abstract
Handheld devices like mobile phones, PDAs have become very popular. They need modern security mechanisms such as the SSL protocols for their connectivity to the unsafe Internet. On the background of security mechanisms, cryptographic approaches are used. The most widely used public key cryptography approach in the Internet is RSA. But RSA needs heavy computing resources such as CPU computing power and memory which handheld devices cannot offer in large scale. Cryptographic approaches should be used in such a manner that they do not affect the user experience of the system. Keeping this in mind, researchers all over the world continuously try to discover newer approaches to the traditional RSA cryptosystem. A number of RSA variants were discovered with this notion. Further a relatively new concept called elliptic curve cryptography has caught the eyes of researchers in recent years with the view that it provides equal security with lesser bit-length of keys than RSA. In this context, this study focuses on the performance benefits of using RSA variants and elliptic curve cryptography over the traditional RSA cryptosystem so as to suggest an effective working cryptographic model for resource-constraint handheld devices.

Keywords
Elliptic Curve Cryptography, RSA, Public Key Cryptography, Network Security, Handheld Devices.

1. Introduction

In this age of universal electronic connectivity, of viruses and hackers, of electronic eavesdropping and electronic fraud, there is indeed no time at which security does not matter [26]. Handheld devices like mobile phones, personal digital assistants (PDAs), etc are needed in day-to-day activities, which also need security mechanisms for their connectivity to the unsafe Internet [11]. Implementing security mechanisms within a system includes enforcing confidentiality, integrity and availability. Cryptography is one of the approaches for that. Even though private key cryptography approaches are more efficient in terms of length and number of keys used than public key cryptography approaches, they are not dominant player in securing communication systems given their difficulty of providing secure key management [18]. So, the better option seems to use public key cryptography like RSA. However, the practical techniques of public key cryptography for network security like RSA demand considerable computing resources for their effectiveness [11].

There is a contradiction between speed and security. The main security parameter of a cryptosystem is the length of the key. This key is used to encrypt and/or decrypt messages. The longer the bit-length of the key is; the more secure is the communication. However, increasing the length of the key can significantly slow down encryption and decryption stages. Thus, the core problem with RSA is the use of large keys in its encryption and decryption.

The time needed to perform encryption/decryption should be sufficiently small on the handheld devices to avoid having an off-putting impact on the user experience of the system, while retaining the security of the cryptosystem. D. Boneh and H. Shacham in [7] have written that 1024-bit RSA decryption on a small handheld device such as the PalmPilot III can take about 30 seconds. Thus the solution to the successful implementation of RSA in constrained-computing environments is mainly by optimizing its decryption time when encryption exponent is small.

A competing public key cryptosystem called elliptic curve cryptosystem has begun to challenge RSA. The basis of this challenge is by the use of smaller keys compared to RSA providing equal level of security for various bit-lengths of keys. This fact suggests that ECC can be a possible alternative to RSA in case of devices with constrained-computing capabilities.

2. RSA Variants

Original RSA

According to K. Hansen, T. Larsen and K. Olsen in [11], the original RSA cryptosystem as proposed by Rivest, Shamir and Adleman consists of three steps: key generation, encryption and decryption.

Key generation: As a part of key generation, choose two distinct large random prime numbers p and q.

Manuscript received November 5, 2011
Manuscript revised November 20, 2011

Then after, compute N = pq, N is used as the modulus for both the public and private keys. Next, compute the totient: $\varphi(N) = (p - 1)(q - 1)$. Choose an integer e such that $1 < e < \varphi(N)$, and e and $\varphi(N)$ share no factors other than 1 (i.e. e and $\varphi(N)$ are relatively prime). Then after, e is released as the public key exponent. Next, choose d to satisfy the congruence relation $ed \equiv 1 \bmod \varphi(N)$; i.e. $de = 1 + k\varphi(N)$ for some integer k. The so computed d is kept as the private key exponent.

Encryption: Encryption is $C = M^e \bmod N$, where M is the message.

Decryption: To recover M from C by using the private key exponent d, the following computation is done: $M = C^d \bmod n$.

In the computation of modular exponentiation in encryption/decryption, repeated square-and-multiply algorithm is used. This algorithm has a running time of O(tv2) where t is the bit-length of the exponent and v is the bit-length of the modulus. To improve encryption efficiency, the encryption exponent is generally chosen to be a small number, while the decryption exponent cannot be chosen that way and comes out to be a large number, so that decryption is usually inefficient in RSA i.e. O(n3).

A number of RSA variants have been proposed by various researchers, all seeking to improve the costly decryption time of RSA so that it can be successfully deployed in limited computing devices.

CRT RSA

CRT RSA is one of the RSA variant for speeding up decryption. The concept behind CRT RSA is to split the costly decryption operation into two smaller and faster modular exponentiations instead of just one using the Chinese Remainder Theorem.

According to the Chinese Remainder Theorem, for a system of r congruences, $x \equiv a_1 \pmod{n_1}$, …, $x \equiv a_r \pmod{n_r}$, where $n_1,\ldots,n_r$ are relatively prime integers and $a_1,\ldots,a_r$ are ordinary integers, has a unique solution modulo $N = n_1 * n_2 * \ldots * n_r$ [11]. This solution can be written as:

$x = (a_1 * N_1 * y_1 + \ldots + a_r * N_r * y_r) \bmod N$, where $N_i = N/n_i$ and $y_i = N_i^{-1} \bmod n_i$ for $1 \le i \le r$ [11].

According to the authors in [11], the three steps in CRT RSA are computed as follows:

Key generation: Key generation is same up to computation of e and d as in the original RSA. Next, $d_p = d \bmod p-1$ and $d_q = d \bmod q-1$ is computed. The public key becomes <N, e> and the private key becomes $<p, q, d_p, d_q>$.

Encryption: Encryption is the same as for the original RSA i.e. $C = M^e \bmod N$.

Decryption: Decryption is divided into the following computations:

First, $M_p = C^{dp} \bmod p$ and $M_q = C^{dq} \bmod q$ is computed. Then, using the Chinese Remainder Theorem, M is found as:

$M = (M_p * q * (q^{-1} \bmod p) + M_q * p * (p^{-1} \bmod q)) \bmod N$.

The major operation in decryption is modular exponentiation. Thus, decryption using CRT RSA requires two times $O((n/2)^3)$ since the bitlength of both the exponent and the modulus are n/2. Compared to the $O(n^3)$ decryption of the original RSA, CRT RSA improves decryption time with a factor $n^3 / (2 * (n/2)^3) = 4^2$ [11]. In [8], the cryptanalysis of CRT RSA has shown that size of d should be large for the security of the cryptosystem.

Multi-Prime RSA

The central idea of Multi-Prime RSA is that by adding more primes to the generation of N, decryption can be split into an arbitrary number of smaller exponentiations so that decryption becomes more efficient than CRT RSA. According to M.J. Hinek in [15], the three steps in Multi-Prime RSA are as follows:

Key generation: Given the security parameter n and $r \ge 3$, r different primes $p_1,\ldots, p_r$ each of (n/r)-bits are generated. Next $N = p_1 * p_2 * \ldots * p_r$ and $\Phi(N) = (p_1-1) * (p_2-1) * \ldots * (p_r-1)$ is calculated. Then e and d are computed as in the original RSA. Finally, $d_i = d \bmod p_i-1$ for $1 \le i \le r$ is computed. The public key becomes <N, e> and the private key becomes $<p_1,\ldots, p_r, d_1,\ldots, d_r>$.

Encryption: Encryption is the same as for the original RSA i.e. $C = M^e \bmod N$.

Decryption: Decryption is divided into the following computations:

First $M_i = C^{di} \bmod p_i$, for $1 \le i \le r$ is calculated. Then using the Chinese Remainder Theorem, M is found as:

$M = (M_1 * N_1 * y_1 + \ldots + M_r * N_r * y_r) \bmod N$ where $N_i = N/p_i$ and $y_i = N_i^{-1} \bmod p_i$ for $1 \le i \le r$.

Decryption in Multi-Prime RSA requires r times $O((n/r)^3)$. Compared to the $O(n^3)$ decryption of the original RSA, Multi-Prime RSA improves decryption time with a factor $n^3 / (r * (n / r)^3) = r^2$ [11]. Since the individual primes need to have a certain size to guard against factorization attacks, so the size of r is limited and thus the actual improvement in decryption is also checked.

Rebalanced RSA

In the original RSA cryptosystem, encryption is more efficient than decryption because e is small and d is large. Thus another way to optimize decryption is to swap the exponents, i.e. make e large and d small. But small values of d are vulnerable to attacks given in [16]. So, in 1990 Wiener proposed a variant Rebalanced RSA that retains the size of d but makes

the decryption exponents $d_p = d \bmod p-1$ and $d_q = d \bmod q-1$ small (at the cost of a larger e). According to the author in [27], the three steps of Rebalanced RSA are as follows:

Key generation: Given the security parameter n and w, two different primes p and q each (n/2)-bits are generated such that gcd (p-1, q-1) = 2. Next $N = p * q$ and $\Phi(N) = (p-1) * (q-1)$ is computed. Then two w-bit integers d_p and d_q satisfying gcd (d_p, p-1) = gcd (d_q, q-1) = 1 and $d_p \equiv d_q \pmod 2$ are calculated. Then an integer d is to be found out such that $d \equiv d_p \bmod p-1$ and $d \equiv d_q \bmod q-1$ according to [7]. Lastly, e is computed using $e * d \equiv 1 \bmod \Phi(N)$. The public key becomes <N, e> and the private key becomes <p, q, d_p, d_q>.

Encryption: Encryption is the same as for the original RSA, $C = M^e \bmod N$, but with a much larger e (on the order of N).

Decryption: Decryption is divided into the following computations:

First, $M_p = C^{dp} \bmod p$ and $M_q = C^{dq} \bmod q$ is computed. Then, using the Chinese Remainder Theorem, M is found as:

$M = (M_p*q* (q^{-1} \bmod p) + M_q*p*(p^{-1} \bmod q)) \bmod N$.

Decryption in Rebalanced RSA requires two times $O(w(n/2)^2)$. Compared to the $O(n^3)$ decryption of the original RSA, Rebalanced RSA improves decryption time with a factor $n^3/(2*w*(n/2)^2) = 2n/w$ [11]. With respect to the security of Rebalanced RSA, it is suggested to set $w \geq 160$ thereby limiting the actual improvement of decryption in practice [11]. Also the speed-up in decryption comes at the cost of a much slower encryption (since e is on the order of N). This means that encryption in Rebalanced RSA is as slow as decryption in the original RSA.

R-Prime RSA

R-Prime RSA is the combination of Rebalanced RSA and Multi-Prime RSA [27]. In Rebalanced RSA, decryption is the same as in CRT RSA. Since Multi-Prime RSA is a generalization of CRT RSA, generalizing Rebalanced RSA to use Multi-Prime RSA in its decryption is the idea behind R-Prime RSA. According to Paixao in [3], the three steps in R-Prime RSA are as follows:

Key generation: Given n, $r \geq 3$ and w; r different primes $p_1,\ldots, p_r$ each (n/r)-bits long are generated such that gcd(p_1-1,…, p_r-1) = 2. Next, $N = p_1 * \ldots * p_r$ and $\Phi(N) = (p_1-1) * \ldots * (p_r-1)$ is calculated. Then r w-bit integers $d_{p1},\ldots, d_{pr}$ satisfying gcd(d_{p1},p_1-1) = … = gcd(d_{pr},p_r-1) = 1 and $d_{p1} \equiv \ldots \equiv d_{pr} \pmod 2$ are computed. Finally d is found out such that $d \equiv d_{p1} \bmod p_1-1,\ldots$, $d \equiv d_{pr} \bmod p_r-1$ according to [7] and e is computed using the congruence relation $e \equiv d^{-1} \bmod \Phi(N)$. The public key becomes <N, e> and the private key becomes <$p_1,\ldots, p_r, d_{p1},\ldots, d_{pr}$>.

Encryption: Encryption is the same as for the original RSA i.e. $C = M^e \bmod N$, but with a much larger e.

Decryption: Decryption is the same as for Multi-Prime RSA, i.e., decryption is split into r modular exponentiations $M_i = C^{d_{pi}} \bmod p_i$ for $1 \leq i \leq r$ and then the Chinese Remainder Theorem is applied. The difference lies in the length of d_{pi} (denoted by d_i in Multi-Prime RSA). In R-Prime RSA, these values are w-bit each whereas in Multi-Prime RSA, they are n/r each.

Decryption in R-Prime RSA requires r times $O(w*(n/r)^2)$. Compared to the $O(n^3)$ decryption of the original RSA, R-Prime RSA improves decryption time with a factor $n^3/(r*w*(n/r)^2) = nr/w$ [11]. The same security considerations as Rebalanced RSA and Multi-Prime RSA apply to R-Prime RSA, i.e. it is mandatory to set $w \geq 160$ and bound the value of r with respect to n [11]. Here also as in Rebalanced RSA, the speed-up in decryption implies a much slower encryption.

3. Elliptic Curve Cryptography

Elliptic curve cryptography is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields [2]. The use of elliptic curves in cryptography was suggested independently by Neal Koblitz and Victor Miller in 1985 [2].

According to H. Tilborg in [9], an elliptic curve over GF(p) or Z_p for prime p is defined by the cubic equation:

$y^2 \equiv (x^3 + ax + b) \bmod p$, where coefficients a and b and the variables x and y are all elements of Z_p such that

$(4a^3 + 27b^2) \bmod p \neq 0$, which implies that the curve has no singular points.

Key Generation: An elliptic curve over GF(p) of the form $y^2 \equiv (x^3 + ax + b) \bmod p$, where coefficients a and b and the variables x and y are all elements of GF(p) is chosen such that $(4a^3 + 27b^2) \bmod p \neq 0$ [4]. The parameters p, a and b are provided. Next a base point G from the group $E_p(a, b)$ (or a generator point if the order of the group is prime) on the elliptic curve is taken whose order must be a large value n [9, 26]. The order n of point G on the elliptic curve is defined as the smallest positive integer n such that n*G = O [26]. Now the private keys to be used by the communicating parties are integers less than this n and are randomly chosen and their respective public

keys are G multiplied by their individual private keys [9].
According to the authors in [20], the key generation steps can be summarized as:

- Suppose Alice and Bob agree on a generator point $G = (x_G, y_G)$ and an elliptic group $E_p(a, b)$.
- Alice chooses an integer n_A and calculates $P_A = n_A*G = (x_A, y_A)$. Now Alice's public key is $P_A = (x_A, y_A)$ and his private key is n_A.
- Bob also chooses an integer n_B and calculates $P_B = n_B*G = (x_B, y_B)$ in the same way as Alice. Now Bob's public key is $P_B = (x_B, y_B)$ and his private key is n_B.

Encryption and Decryption in ECC: To perform encryption, the plaintext message m to be sent is first encoded as a point $P_m = (x_m, y_m)$ and it is this point that will be encrypted as ciphertext and subsequently decrypted [13].
According to the authors in [20] and [26], encryption of a point is done as:

- Suppose that Alice wishes to send a message P_m to Bob. For this, Alice chooses a random positive integer k and computes $c_1 = k * G$ and $c_2 = P_m + k * P_B$.
- Alice sends the ciphertext $C_m = \{c_1, c_2\}$ to Bob.

Here Alice has used Bob's public key P_B for encryption. Now again according to the authors in [20] and [26], upon receiving the ciphertext pair (c_1, c_2) from Alice, Bob recovers the message as follows:

- Bob multiplies c_1 by his private key n_B and subtracts the result from c_2, i.e. $P_m + k * P_B - n_B * (k * G) = P_m + k * (n_B * G) - n_B * (k * G) = P_m$.

4. Implementation

We have implemented the RSA with its variants and the ECC in J2ME™ platform. J2ME™ (Java™ 2 Micro Edition) is the de facto application platform used in handheld devices [2]. J2ME™ contains a subset of the APIs of Java™ Standard Edition. The Connected Limited Device Configuration (CLDC) and the Mobile Information Device Profile (MIDP) define the available APIs [21]. CLDC 1.1 and MIDP 2.1 are used in this work since they are supported in NetBeans 6.5. They do not contain any general cryptographic API for multi-precision computations like the BigInteger and SecureRandom classes [11]. Thus to support those computations the APIs provided by bouncycastle.org have been used [12].

Implementation Details of RSA and its Variants: RSA with key sizes of 1536, 2048, 3072 and 7680 bits are equivalent in security to the ECC finite field size of 192, 224, 256 and 384 bits [13, 20, 26]. So they are implemented with these key sizes and the timings of key generation, encryption and decryption are captured using the standard Java™ function.

Implementation Details of ECC: NIST curve domain parameters in GF (p) for the curve y*y mod p = (x*x*x + a*x + b) mod p were taken for bit sizes of 192, 224, 256 and 384 from [13].

5. Analysis

For making the analysis realistic, great care has been taken. In RSA, CRT RSA and Multi-Prime RSA, the public key is taken to be 65537. The number of individual primes to be generated in Multi-Prime RSA and R-Prime RSA is taken to be 3 for modulus sizes of 1536, 2048 and 3072, and 4 for modulus size of 7680 as suggested by Hinek in [15]. In Rebalanced RSA and R-Prime RSA the size of the individual small decryption exponents is taken to be 224. With ECC, the size of the private key is taken to be equal to the finite field size and the scalar used in encryption is randomly generated to be of 160-bits. The point to be encrypted is taken as the base point of the finite field in ECC, while for RSA and its variants a numeric message of 256-bits is randomly generated and then encrypted/decrypted.

The following graphs summarize the outcomes of the experiments carried out during this study.

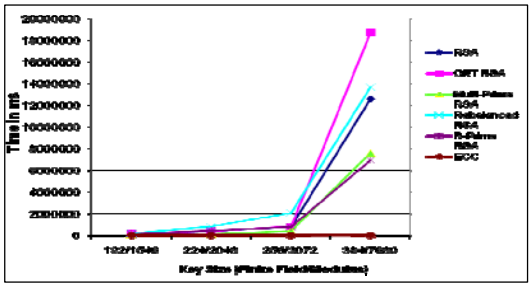
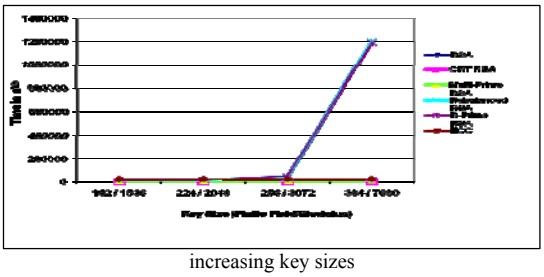


Fig. 1: Graph showing the variation of key generation time with



increasing key sizes

Fig. 2: Graph showing the variation of encryption time with increasing key sizes

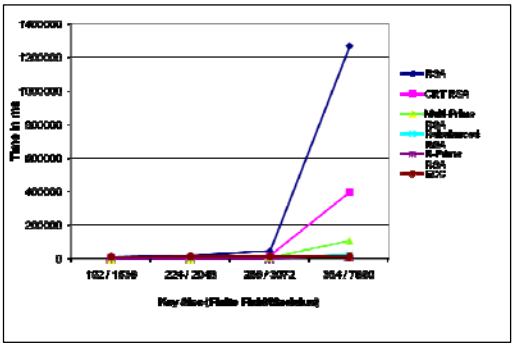


Fig. 3: Graph showing the variation of decryption time with increasing key sizes

6. Conclusion

The central part of RSA cryptosystem and its variants: CRT RSA, Multi-Prime RSA, Rebalanced RSA, R-Prime RSA; and ECC were implemented on J2ME™ platform. The tests were run on typical numerical data which showed that ECC outperformed RSA and each of its variants with a factor when equivalent key sizes were taken of 384/7680 bits large in all three steps: key generation, encryption and decryption. Consequently, it can be asserted that when higher security level is required, ECC will be the suitable choice for handheld devices in future.

7. Recommendation and Future Work

With this study ECC is discovered as an effective working cryptographic model for handheld devices when the required security level is high. The recommendations after this study are:

- The free of cost Bouncy-Castle cryptography library may be changed with other paid libraries for a better performance analysis.
- Optimization algorithms exist for ECC that may be used to further reduce the key generation, encryption and decryption times of ECC that are left in this study.
- Other standard elliptic curves like NIST recommended curves over GF(2^m) and SECG curves over GF(p) and GF(2^m) may be tested.

References

[1] A. Menezes, P. van Oorschot and S. Vanstone, “Handbook of Applied Cryptography”, 1st Edition, CRC Press, 1997.

[2] B. Kayayurt, “End-to-End Security for Mobile Devices”, Department of Computer Engineering, Izmir Institute of Technology, Izmir, Turkey, July 2004.

[3] C. A. M. Paixao, “An Efficient Variant of the RSA Cryptosystem”, Institute of Mathematics and Statistics, University of Sao Paulo, Brasil.

[4] Certicom Research, “Standards for Efficient Cryptography SEC1: Elliptic Curve Cryptography”, Version 1.0, September 20, 2000.

[5] Certicom Research, “Standards for Efficient Cryptography SEC2: Recommended Elliptic Curve Domain Parameters”, Version 1.0, September 20, 2000.

[6] D. Boneh, “Review of Standards for Efficient Cryptography SEC1: Elliptic Curve Cryptography”.

[7] D. Boneh and H. Shacham, “Fast Variants of RSA”, RSA Laboratories, CryptoBytes, Volume 5, No. 1, Winter/Spring 2002.

[8] E. Jochemsz, “Cryptanalysis of RSA Variants Using Small Roots of Polynomials”, Eindhoven Technical University, October 4, 2007.

[9] H.C.A.V. Tilborg, “Fundamentals of Cryptology”.

[10] H. Pietilainen, “Elliptic Curve Cryptography on Smart Cards”, Helsinki University of Technology, Faculty of Information Technology, Department of Computer Science, October 30, 2000.

[11] K. Hansen, T. Larsen and K. Olsen, “On the Efficiency of Fast RSA Variants in Modern Mobile

Phones", International Journal of Computer Science and Information Security, Vol. 6, No. 3, 2009.
[12] Legion of the Bouncy Castle, "The Bouncy Castle Crypto APIs for Java", www.bouncycastle.org, 2011.
[13] M. Brown, Dept. of C&O, University of Waterloo, Canada, D. Hankerson, Dept. of Discrete and Statistical Sciences, Auburn University, USA, J. Lopez, Dept. of Computer Science, University of Valle, Colombia, and A. Menezes, Certicom Research, Canada, "Software Implementation of the NIST Elliptic Curves Over Prime Fields".
[14] Md. Ali-Al-Mamun, Md. M. Islam, S.M.M. Romman and A.H.S.U. Ahmad, "Performance Evaluation of Several Efficient RSA Variants." International Journal of Computer Science and Network Security, Vol. 8, No. 7, July 2008.
[15] M. J. Hinek, "On the Security of Multi-Prime RSA", David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada, June 13, 2006.
[16] M. J. Wiener, "Cryptanalysis of Short RSA Secret Exponents", BNR, P.O. Box 3511 Station C, Ottawa, Ontario, Canada, K1Y 4H7, August 3, 1989.
[17] M. Saeki, "Elliptic Curve Cryptosystems", School of Computer Science, McGill University, Montreal, February 1997.
[18] R.A. Mollin, "An Introduction to Cryptography", Second Edition, Chapman & Hall/CRC, Taylor & Francis Group, 2007.]
[19] RSA Laboratories, "PKCS #1 v2.1: RSA Cryptography Standard", RSA laboratories, June 14, 2002.
[20] R. Soram and M. Khomdram, "Juxtaposition of RSA and Elliptic Curve Cryptosystem", International Journal of Computer Science and Network Security, Vol. 9 No.9, September 2009.
[21] Sun Microsystems Inc., http://java.sun.com/javame.
[22] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, "Introduction to Algorithms", Second Edition, Prentice-Hall India, 2007.
[23] T. Struk, "Elliptic Curve Cryptography as Suitable Solution for Mobile Devices", National University of Ireland, Galway, August 28, 2009.
[24] T. Takagi, "Fast RSA-type Cryptosystem Modulo pkq", NTT Software Laboratories, 3-9-11, Midoro-cho Musashino-shi, Tokyo 180-0012, Japan.
[25] W. Chou, "Elliptic Curve Cryptography and Its Applications to Mobile Devices", University of Maryland, Department of Mathematics, College Park.
[26]
[27] W. Stallings, "Cryptography and Network Security", Fourth Edition, 2009.
[28] Z. Zhong and Z. Xia, "On the Variants and Speed Methods of RSA", Department of Computer Science, School of Computer, Wuhan University, Wuhan, Hubei 430072 P.R. China.

Jagdish Bhatta received the B.Sc. and M.Sc. degrees in Computer Science from Tribhuvan University, Nepal in 2004 and 2007, respectively. Since 2007 he is a full time faculty member at the Tribhuvan University. He has been involved in number of researches conducted in the department and also has supervised graduate students dissertation. His research areas are Cryptography and Network Security, Artificial Intelligence, Automata Theory, Computational Geometry.

Lok Prakash Pandey received the B.Sc. and M.Sc. degrees in Computer Science from Tribhuvan University, Nepal in 2004 and 2009, respectively. He has two years of teaching experience in "Structured Programming", "Compiler Design and Construction", "Cryptography" and "Network Security". He also has worked as a "Chip-Level Hardware Instructor". His main area of interest in research includes "Cryptography and Network Security" and "Operating Systems". He is currently working as a Program Officer and faculty member for BIM and B.Sc. CSIT studies in Nagarjuna College of IT affiliated to Tribhuwan University.