Design and Development of Verifiable and Recoverable Encryption of J_2-RSA Signatures

C. Anitha[1], M. Padmavathamma[2]

ABSTRACT

This paper deals with the design of J_2-RSA-VRES, a novel instantiation of the VRES scheme suited for J_2-RSA signatures[5] with the following features: 1) J_2-RSA assumption, 2) J_2-RSA-VRES design that is based on the theorem of cross decryption, defined with the following four procedures: a) J_2-RSA-VRES initialization, b) J_2-RSA-VRES Generation c) J_2-RSA-VRES Verification d) J_2-RSA-VRES Recovery 3) its security which relies on RSA assumption by proving that it satisfies the correctness, soundness, and secrecy properties and 4) evaluation of its performance by comparing it with the related VRES schemes.

Key words: cryptography, J_2-RSA, VRES, encryption, decryption, TTP-confidentiality.

1. INTRODUCTION

Cryptography is the study of mathematical techniques related to the aspects of message secrecy. Security threats to e-commerce transactions come not only from external attackers, but also from misbehaving business partners.

In order to mitigate the risks associated with conducting e- transactions and enable trust among potential business partners, adequate security services should be provided to ensure that exchanges of valuable business items are performed *fairly* and that evidence of e-transactions cannot be *repudiated. N*on-repudiation is a special case in a broader problem of fair exchange. This article presents the J_2-RSA-VRES method, which enables the protocol to achieve strong fairness, and the e-goods and key certification method, which prevents a dishonest party from using some junk data in exchange for the receipt. The e-goods and the receipt exchanged enjoy the confidentiality protection, and the protocol places only weak security requirements on the STTP. The design of J_2-RSA-VRES scheme is based on the results of the Theorem of Cross Decryption, and its from using some junk data in exchange for the receipt. The e-goods and the receipt exchanged enjoy the confidentiality protection, and the protocol places only weak security requirements on the STTP. The design of J_2-RSA-VRES scheme is based on the results of the Theorem of Cross Decryption, and its security relies on the J_2-RSA Assumption, both of which are presented in the following table that summarizes the J_2-RSA notation used throughout the article.

[1]Assistant Professor, Department of MCA, C.R.Engineering College, Tirupathi-517 506.

[2]Head, Department of Computer Science, Svuccmis, Tirupathi-517 502.

Public J_2-RSA modulus
n = pq, where p, q are large primes, and
p = 2p' + 1, q = 2q' + 1 for p', q' prime and $J_2(n)=(p^2-1)(q^2-1)$
Public key
pk = (e, n): gcd(e, f(J_2(n)) = 1, and 0 < e < (J_2(n))
Private key
sk = (d, J_2(n)): $d = e^{-1} \bmod f(J_2(n))$
Encryption
Plain text message: M, 0 < M < J_2(n)
Message encryption: $C = E_{pk}(M) = M^e \bmod n$
Decryption
Cipher text message: C
Message decryption: $M = D_{sk}(C) = C^d \bmod n$
Signature generation
$Sign(M) = h(M)^d \bmod J_2(n)$
Signature verification
Check if the following holds: h(M)?= $Sign(M)^e \bmod J_2(n)$

Table: J_2-RSA cryptosystem and signatures (summary)

1. J_2-RSA Assumption. Given a composite number $J_2(n) = pq$, where p, q are secret primes and $p = 2p' + 1$, $q = 2q' + 1$ for some prime p', q', an exponent e > 1, and a random element C ° $Z^*_{J2(n)}$. It is hard to compute M° $Z^*_{J2(n)}$ such that $M_e = C \bmod J_2(n)$. Here, $Z^*_{J2(n)} = \{1, \ldots, J_2(n) - 1\}$ is a multiplicative group of integers modulo $J_2(n)$. The J_2-RSA assumption implies that factoring integers such as $J_2(n)$ is computationally hard [4].

Theorem of Cross-Decryption[2]. Let $J_2(n_1)$ and $J_2(n_2)$ be relatively prime module of two different J_2-RSA cryptosystems, $e_1 = e_2 = e$ the corresponding public-key exponents. For any two messages M and M', such that $M, M' < \min(J_2(n_1), J_2(n_2))$, the following holds: $(M^e \bmod (J_2(n_1) \times J_2(n_2))) \bmod J_2(n_1) = M'^e \bmod J_2(n_1)$ if and only if M = M2, $(M^e \bmod (J_2(n_1) \times J_2(n_2))) \bmod J_2(n_2) = M'^e \bmod J_2(n_2)$ if and only if M = M'.

2. Design of J_2-RSA-VRES

Assume that party P_b generates J_2-RSA-VRES for his J_2-RSA signature $Sign_b(x)$ on data item x, denoted as J_2-RSA-$VRES_b(x)$, party P_a performs J_2-RSA-VRES verification, and the TTP P_t performs J_2-RSA-VRES recovery. J_2-RSA-VRES can be defined using the following four procedures.

• **J_2-RSA-VRES initialization**: Party P_b registers with P_t and obtains J_2-RSA key pair ($pk_{bt} = (e_{bt}, J_2(n_{bt}))$, $sk_{bt} = (d_{bt}, J_2(n_{bt}))$). This key is in addition to P_b's original J_2-RSA public key pair ($pk_b = (e_b, J_2(n_b))$, $sk_b = (d_b, J_2(n_b))$).

J_2-RSA modulus $J_2(n_{bt})$is a product of two distinct large secret primes chosen by P_t and is approximately of the same size as $J_2(n_b)$. In addition, e_{bt} is required to be the same as e_b, i.e. $e_b = e_{bt}$. For this special public key, P_t issues P_b with the following certificate C_{bt}: $C_{bt} = (pk_{bt}, w_{bt}, s_{bt})$. Here, pk_{bt} is P_b's new public key. W_{bt} is defined as:

$W_{bt} = (h(sk_t, pk_{bt})^{-1} \times d_{bt}) \bmod J_2(n_{bt})$, where sk_t is P_t's own private key. The reason for including w_{bt} in the certificate is to eliminate the need for P_t to store and safe-keep private key sk_{bt}. P_t can easily compute private key exponent d_{bt} from w_{bt} and its own private key, i.e. $d_{bt} = (h(sk_t, pk_{bt}) \times w_{bt}) \bmod J_2(n_{bt})$. Finally, s_{bt} is P_t's RSA signature on the items (pk_{bt} ,w_{bt}), i.e.: $s_{bt} = Sign_t(pk_{bt}, w_{bt})$. The certificate C_{bt} can be implemented as a standard X.509 v3 certificate [6]. An extension field in X.509 v3 certificate can be used to incorporate the number w_{bt} in the certificate, as shown in the below figure.

J_2-RSA-VRES Generation: In order to generate J_2-RSA-VRES$_b$(x) for his signature Sign$_b$(x), party P_b obtains two numbers r_b and y_b from P_t. r_b is a random number such that $0 < r_b < J_2(n_b)$, and number y_b is computed as: $y_b = r_b^{eb} \bmod (J_2(n_b) \times J_2(n_{bt}))$. P_b then computes: $x_b = r_b \times (h(x))^{db} \bmod J_2(n_b) = r_b \times \text{Sign}_b(x) \bmod J_2(n_b)$,

$$xx_b = r_b \times (h(y_b))^{dbt} \bmod J_2(n_{bt}).$$

J_2-RSA-VRES$_b$(x) is defined as triple $(y_b, x_b, xx_b)_x$. Here, e_b and d_{bt} are public and private exponents of P_b's public key pk_b and private key sk_{bt}, respectively. y_b is a modified J_2-RSA encryption of random number r_b. x_b effectively encrypts signature Sign$_b$(x) using random number r_b. Finally, xx_b is a control number to confirm the correct usage of numbers r_b, y_b and x_b. Number r_b is chosen and number y_b is computed by P_t. In addition, y_b is certified by P_t through certificate $Cert_{yb} = (P_b, y_b, \text{Sign}_t(y_b))$ signed by P_t. This certificate is to confirm the correctness of y_b to P_a during J_2-RSA-VRES Verification. P_b can obtain r_b, y_b and $Cert_{yb}$ from P_t prior to exchange. These items should be fresh for each exchange, and can be obtained from P_t in bulk (i.e. P_b can obtain multiple sets of $(r_b, y_b, Cert_{yb})$ at a time).

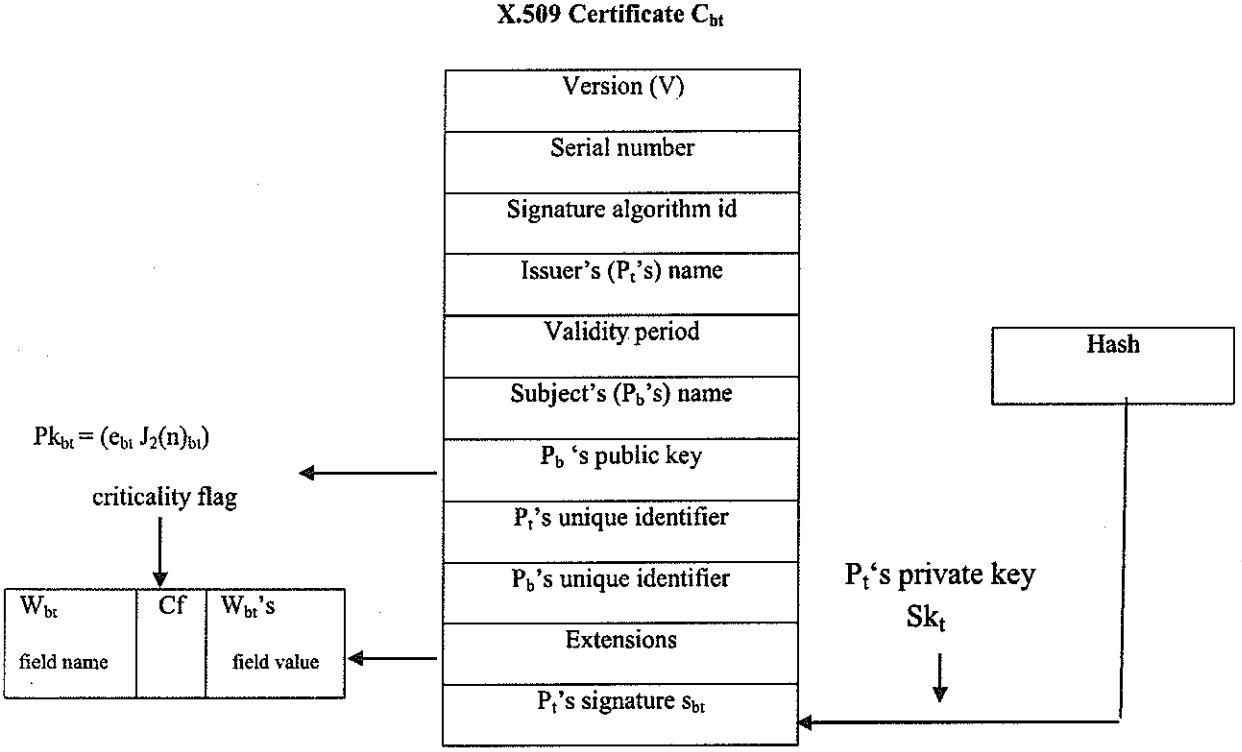

Figure : implementation of certificate C_{bt}

•**J_2-RSA-VRES Verification**: To verify P_b's J_2-RSA-VRES$_b$(x), P_a performs the following verifications:

Verification (a): Check the correctness of P_t's signature s_{bt} in certificate C_{bt}. The purpose of Verification (a) is to ensure that C_{bt} is a valid certificate issued by P_t.

Verification (b): Check the correctness of P_t's signature in certificate $Cert_{yb}$. The purpose of Verification (b) is to ensure that $Cert_{yb}$ is valid and guarantees the correctness of number y_b chosen by P_t. Verification

Verification (c): To confirm that $x_b^{eb} \bmod J_2(n)_b = y_b \times h(x) \bmod J_2(n_b)$. This confirms that number x_b indeed contains P_b's correct signature.

Verification (d): To confirm that $xxb^{ebt} \bmod J_2(n_{bt}) = y_b \times h(y_b) \bmod J_2(n_{bt})$. Verification (d) together with Verification (c) ensures that the same number r_b is used in the computations of y_b, x_b and xx_b, and that the modulus operation in y_b is based on $J_2(n_b) \times J_2(n_{bt})$

• **J_2-RSA-VRES Recovery**: To recover signature $Sign_b(x)$ from J_2-RSA-VRES$_b(x)$, P_t first derives the exponent d_{bt} of the shared private key sk_{bt} from P_b's certificate C_{bt} using its own private key sk_t as:

$d_{bt} = (h(sk_t, pk_{bt}) \times w_{bt}) \bmod J_2(n_{bt})$. P_t then uses d_{bt} to decrypt $y_b \bmod J_2(n_{bt}) = Epk_{bt}(r_b)$ to recover r_b (as in equation below), which can then be used by P_a to compute P_b's signature from x_b as: $Sign_b(x) = (r_b^{-1} \times x_b) \bmod J_2(n_b)$. The purpose of the J_2-RSA-VRES initialization procedure is for P_b and P_t to establish private key sk_{bt} that belongs to P_b, but is also known to P_t. This key is then used by P_b in J_2-RSA-VRES Generation. Because P_t also knows this key (i.e. can compute it from certificate C_{bt}), P_t can use it in J_2-RSA-VRES Recovery to decrypt number r_b from y_b, which can then be used for the computation of P_b's $Sign_b(x)$ from x_b. The idea of making J_2-RSA-VRES$_b(x)$ "decryptable" with P_b's two private keys sk_b and sk_{bt} provides the recoverability of the encrypted signature. According to the theorem of Cross-Decryption, y_b represents the encryption of r_b using either of the public keys pk_b or pk_{bt}.

$y_b \bmod J_2(n_b) = (r_b^{eb} \bmod (J_2(n_b) \times J_2(n_{bt})) \bmod J_2(n_b) = r_b^{eb} \bmod J_2(n_b) = Epk_b(r_b)$.

$y_b \bmod J_2(n_{bt}) = (r_b^{e}{}_b \bmod (J_2(n_b) \times J_2(n_{bt})) \bmod J_2(n)_{bt} = r_b^{e}{}_{bt} \bmod J_2(n_{bt}) = Epk_{bt}(r_b)$

The implication of these two equations is that number r_b, which can be used for computing P_b's signature from x_b, can be decrypted from y_b using either P_b's private key sk_b or P_b's private key sk_{bt}, also known to P_t. There is one final point about number r_b. This number must satisfy $0 < r_b < J_2(n_b)$. Otherwise, if $r_b > n_b$, it would be possible for P_b to cheat P_a in the following way. P_b computes two additional numbers as:

$r'_b = r_b \bmod J_2(n_b)$, $r''_b = r_b \bmod J_2(n_{bt})$.

Note that $r_b \neq r'_b$ as $r_b > J_2(n_b) > r'_b$. P_b then uses these numbers in J_2-RSA-VRES generation as follows:

$$y_b = r_b^{e}{}_b \bmod (J_2(n_b) \times J_2(n_{bt})),$$

$x_b = r'_b \times (h(x))^{d}b \bmod J_2(n_b) = r'_b \times Sign_b(x) \bmod J_2(n_b)$,

$$xx_b = r''_b \times (h(y_b))^{d}bt \bmod J_2(n_{bt}).$$

Both J_2-RSA-VRES Verification (c) and J_2-RSA-VRES Verification (d) would pass using the above values for y_b, x_b, and xx_b. During J_2-RSA-VRES Recovery, P_t would recover r_b from y_b, while it is r'_b ($\neq r_b$) that is needed for computing $Sign_b(x)$ from x_b. As P_a cannot trust P_b to choose r_b such that $0 < r_b < J_2(n_b)$ and compute y_b correctly, it has to be done by P_t and certified in $Cert_{yb}$.

3. Security of J_2-RSA-VRES

Theorem of J_2-RSA-VRES Security. Under the J_2-RSA Assumption, J_2-RSA-VRES is a secure VRES scheme for J_2-RSA signatures.

Proof. To prove the security of J_2-RSA-VRES, we should prove that it satisfies the correctness, soundness, and secrecy properties.

• **Correctness**: If a valid verifiable and recoverable signature J_2-RSA-VRES$_b(x) = (y_b, x_b, xx_b)_x$, produced by J_2-RSA-VRES Generation, is used as an input to J_2-RSA-VRES verification, we have:

$x_b^{e}{}_b \bmod J_2(n_b) = (r_b \times (h(x))^{d}{}_b)^{e}{}_b \bmod J_2(n_b) = y_b \times h(x) \bmod J_2(n_b)$.

Further, we have:

$xx_b^{ebt} \bmod J_2(n_{bt}) = (r_b \times (h(y_b))^d{}_{bt})^{e}{}_{bt} \bmod J_2(n_{bt}) = y_b \times h(y_b) \bmod J_2(n_{bt})$.

This means that both Verification (c) and Verification (d) hold. If P_b's certificates C_{bt} and $Cert_{yb}$ are valid, then Verification (a) and Verification (b) will pass as well. This means that J_2-RSA-VRES$_b$(x) is accepted by J_2-RSA-VRES Verification.

• **Soundness (Unforgeability)**: Forging J_2-RSA-VRES$_b$(x) means generating numbers y_b, x_b and xx_b such that J_2-RSA-VRES Verification will pass, while the signature that is encrypted inside J_2-RSA-VRES$_b$(x) is not P_b's valid signature on x. P_b may attempt the forgery by choosing different numbers r_b, r'_b and r''_b and using them for computing y_b, x_b and xx_b respectively (as defined by equations). As we have discussed previously, it is possible to choose these three numbers if $r_b > n_b$. However, J_2-RSA-VRES Verification (b) will prevent this attempt, as it is designed to confirm the correctness of r_b used to compute y_b. In the case $0 < r_b < J_2(n_b)$, we show that these three different numbers cannot be selected to pass the verification. J_2-RSA-VRES Verification (c) will detect if $r_b \neq r'_b$:

$X_b^{eb} \bmod J_2(n_b) = (r'_b \times h(x)^{db})^{eb} \bmod J_2(n_b)$

$= (r'_b{}^{eb} \times h(x)) \bmod J_2(n_b)$

$\neq (r_b^{eb} \times h(x)) \bmod J_2(n_b) = y_b \times h(x) \bmod J_2(n_b)$.

J_2-RSA-VRES Verification (d) will do the same for numbers r_b and r''_b. More generally, a successful forgery would mean that P_b can generate numbers y_b, x_b and xx_b such that:

$y_b = r_b^{ep} \bmod (J_2(n_b) \times J_2(n_{bt}))$, for some r_b (as vouched by P_t in Cert y_b),

$x_b^{eb} \bmod J_2(n_b) = (y_b \times h(x)) \bmod J_2(n_b)$, (for Verification (c) must pass),

$xx_b^{e}{}_{bt} \bmod J_2(n_{bt}) = (y_b \times h(y_b)) \bmod J_2(n_{bt})$ (for Verification (d) must pass),

while $x_b \neq r_b \times Sign_b(x) \bmod J_2(n_b)$.

However, this is not possible, as decrypting $Epk_b(x_b) = x_b^{eb} \bmod J_2(n_b)$ with key sk_b leads to:

$X_b = Dsk_b(x_b^{eb} \bmod J_2(n_b))$

$= (x_b^{eb} \bmod J_2(n_b))^{db} \bmod J_2(n_b)$

$= ((y_b \times h(x)) \bmod J_2(n_b))^{db} \bmod J_2(n_b)$

$= y_b^{db} \times h(x)^{db} \bmod J_2(n_b)$

$= D_{skb}(y_b \bmod J_2(n_b)) \times Sign_b(x) \bmod J_2(n_b)$

$= Dsk_b(Epk_b(r_b)) \times Sign_b(x) \bmod J_2(n_b)$

$= r_b \times Sign_b(x) \bmod J_2(n_b)$.

Thus, no forgery attempt will succeed.

• **Secrecy**: To prove that J_2-RSA-VRES possess the secrecy property, it should be proved that it leaks no knowledge to party P_a about P_b.s original signature $Sign_b(x)$ encrypted in J_2-RSA-VRES, without knowing random number r_b or either of P_b's private keys sk_b and sk_{bt}. In order to crack J_2-RSA-VRES, P_a may try to obtain $Sign_b(x)$ directly from $x_b = r_b \times Sign_b(x) \bmod J_2(n_b))$. This can be done by trying to guess r_b (or $Sign_b(x)$), which is computationally hard, or by factoring x_b, which is also computationally difficult under the J_2-RSA Assumption. Another way P_a may try to get $Sign_b(x)$ is to obtain r_b from y_b first, and then use it to compute $Sign_b(x)$ from x_b. However, r_b can be obtained from y_b only by J_2-RSA decryption with either private key sk_b or sk_{bt}. . As P_a does not have knowledge of P_b's private keys, this is computationally difficult under the J_2-RSA Assumption. Therefore, we can conclude that it is computationally difficult for P_a to illegitimately obtain $Sign_b(x)$ from J_2-RSA-VRES$_b$(x). Based on the above considerations we can conclude that J_2-RSA-VRES indeed

is a secure VRES scheme. In addition to the above properties, J_2-RSA-VRES also protects the confidentiality of the P_b's signature from the TTP during recovery, i.e. it satisfies:

• **TTP-confidentiality**: It is computationally difficult for the TTP to obtain P_b's signature $Sign_b(x)$ as a result of executing J_2-RSA-VRES recovery. Clearly, in order to recover P_b's signature from J_2-RSA-$VRES_b(x)$ for P_a, P_t only needs to recover random number r_b from y_b, which can then be used to compute $Sign_b(x)$ from x_b. Number x_b need not be disclosed to P_t, and disclosing y_b to P_t does not affect the secrecy of Sign $_b(x)$. Therefore, the confidentiality of $Sign_b(x)$ is protected. Also, apart from its own private key sk_t, the TTP has no need to safe-keep any security sensitive items, such as shared private key sk_{bt}. The TTP can always compute sk_{bt} from P_b's certificate C_{bt}. This enables to relax the security and storage requirements placed on the TTP, as no security sensitive information is kept at or disclosed to it during recovery. We call such a third party semi-trusted (STTP).

4. Comparison with Related Schemes

In this Section we compare J_2-RSA-VRES with related VRES schemes by considering their various characteristics (e.g. type of signature being encrypted, interactivity, TTP-confidentiality) and their theoretical performances. Performance evaluation is carried out with respect to the number of modular exponentiations used in J_2-RSA-VRES generation, verification and recovery, as exponentiations are computationally most expensive operations. The most expensive part (in terms of communication and computational costs) of a J_2-RSA-VRES scheme is J_2-RSA-VRES verification. According to Bao et al.'s scheme for DSA signatures [1], VRES verification is also an interactive ZK proof requiring 9 exponentiations in each round, which is repeated 100 times as in Bao et al.'s scheme that totals to 900 exponentiations for just VRES verification. In Asokan et al.'s VRES scheme for DSA signatures [3], VRES verification is also an interactive ZK proof requiring 4 exponentiations in each round, which is repeated 100 times as in Bao et al.'s scheme that totals to 400 exponentiations for just VRES verification. The above VRES schemes are thus rather inefficient. In all, compared schemes exponentiations are performed either modulo $J_2(n)$, where $J_2(n)$ is a public J_2-RSA modulus, or modulo p, where p is a public prime parameter in DL-based schemes. Since the time taken to perform a modular exponentiation depends on the size of modulus, one should be careful when comparing exponentiations modulo two different numbers. However, Digital Signature Standard prescribes p be a 512 to 1024-bit number, and J_2-RSA Laboratories currently recommend J_2-RSA modulus n should have 1024 bits. Thus, assuming that n and p have the same bit length of 1024 bits, our comparison can be justified.

5. Conclusions

In this article, we have formally defined J_2-RSA-VRES and proved its strength. In J_2-RSA-VRES, a signature is not encrypted under the TTP's public key but it is encrypted with certain random number r_b, which in turn, encrypted in a special way in number y_b so that it can be encrypted with either P_b's private key sk_b, or with private key sk_{bt} shared between P_b and P_t. To perform recovery, the TTP needs only to know y_b, from which it recovers r_b. As y_b and r_b contain no information of P_b's original signature $Sign_b(x)$, its confidentiality can be preserved from the TTP. This enables us to make it a Semi-Trusted Third Party (STTP), which need not be unconditionally trusted as a TTP. The performance evaluation of J_2- RSA-VRES with respect to the number of modular exponentiations required for its generation, verification

and recovery. Comparison of J_2-RSA-VRES to related VRES schemes has shown that it is among the most efficient schemes. It provides non-interactive VRES verification (only provided by one more VRES scheme) and is the only scheme so far to satisfy the STTP-confidentiality property.

REFERENCES :

[1] F.Bao, R.Deng, and W.Mao. *Efficient and practical fair Exchange Protocols with Off-line TTP.* In Proceedings of Symposium on Security and Privacy, pages 77-85, 1998.

[2] I.Ray and Narasimhamurthi. *A Fair Exchange E-commerce Protocol with Automated Dispute resolution.* In the proceedings of the IFIP Workshop on Database Security, pages 27-38, 2000.

[3] N.Asokan, V.Shroup, and M.Waidner. *Optimistic Fair Exchange of Digital Signatures.* IEEE Journal on Selected Areas in Communications, 18(4):593-610, 2000.

[4] R.Cramer and V.Shoup.*Signature schemes based on the strong RSA assumption.* ACM Trans. Inf. Syst. Secur.,3(3):161-185,2000.

[5] R.L.Rivest, A.Shamir, and L.M.Andleman. *A method for Obtaining Digital signatures and Public-key Cryptosystems.* Communications of the ACM, 21(2):120-126, 1978.

[6] The Internet Engineering Task Force (IETF). The PKIX Working Group. X.509 Certificate Specification. Available at http://www.itef.org/rfc/rfc/2401.txt.

Author's Biography

Mrs. C. Anitha has done her M.Sc., MCA, MTech, and M.Phil, and pursuing her PhD in S.V.University, Tirupati. She has been an Assistant Professor in the Department of MCA, C.R.Engineering College, Tirupati with 8 years of experience teaching MCA, M.Sc and B.Tech students. Her areas of specialization are cryptography and network security, privacy preserving data mining.

Prof. M. Padvathamma, M.Sc, M.S, M.Phil, M.Ed, Ph.D has been working as Head, Dept of Computer Science, in Sri Venkateswara University, Tirupati. She has 20 years of teaching experience for PG and 5 years for UG and has guided many PhDs. Her areas of specialization are cryptography & network security, privacy preserving data mining.