An Efficient Implementation of Digital Signature Algorithm with SRNN Public Key Cryptography

*Mr. Hemant Kumar, **Dr. Ajit Singh

**Asst. Professor, Computer Science & Engineering Dept., Kumaon Engineering College, Uttarakhand, India

ABSTRACT

Digital Signature schemes are mostly used in cryptographic protocols to provide services like entity authentication, authenticated key transport and authenticated key agreement. This architecture is related with Secure Hash Function and 512-bit SRNN cryptographic algorithm. SRNN algorithm is based on RSA algorithm with some modification and included more security. In this algorithm we have an extremely large number that has two prime factors (similar to RSA). In addition of this we have used two natural numbers in pair of keys (public, private). This natural number increases the security of the cryptosystem. If the security of our method proves to be adequate, it permits secure communication to be established without the use of carriers to carry keys. In this paper, a new algorithm has been designed for generating signature that overcomes the

Shortcomings of the RSA system (longer processing time & computational overheads), also the new algorithm can be achieves high security for digital signature.

Key Words: Public Key Cryptography, Digital Signature, RSA, SRNN.

1.INTRODUCTION

In the present scenario the applications like e-commerce and secure communications over open networks have made clear the fundamental role of public key cryptosystem as unique security solutions. But, the problem that goes with these solutions is that the private keys need to be secured in these applications. This problem is further worsened in the cases where a single and unchanged private key has to be kept secret for very long duration (such is the case of certification authority keys, and e-cash keys). [1]

Digital signatures are used to detect unauthorized modifications to data and to authenticate the identity of the signatory. In addition, the recipient of signed data can use a digital signature as evidence in demonstrating to a third party that the signature was, in fact, generated by the claimed signatory. Basically a Digital Signature is a checksum which depends on the time period during which it was produced. It depends on all the bits of a transmitted message, and also on a secret key, but which can be checked without knowledge of the secret key. A major difference between handwritten and digital signatures is that a digital signature cannot be a constant; it must be a function of the document that it signs. .If this were not the case then a signature, could be attached to any document. A digital signature is a function of the entire document, variation of even a single bit should produce a different signature.[1]A digital signature is represented in a computer as a string of bits. A digital signature is computed using a set of rules and a set of parameters that allow the identity of the signatory and the integrity of the data to be verified. Digital signatures can be generated on both stored and transmitted data [2].

A digital signature algorithm authenticates the integrity of the signed data and the identity of the signatory. A digital signature can be used as an evidence by a recipient of signed data in demonstrating to a third party that the signature was, in fact, generated by the claimed signatory. Mainly digital signature is use in e-mail, electronic data interchange, software distribution, and other applications that require data integrity assurance and data origin authentication [3]. The wireless protocols, like HiperLAN/2 [4], and WAP [5], have specified security layers and the digital signature algorithm have been applied for the authentication purposes.

In this paper an implementation of the Digital Signature Standard is proposed where a Secure Hash Function [6], and a 512-bit SRNN algorithm is used.
The remainder of this paper is organized as follows: in section 2 a brief description of the Digital Signature Scheme is given. In the next section the proposed architecture is presented and analyzed in details.

Digital signature algorithm

A digital signature is represented in a computer as a string of binary digits. A digital signature is computed using a set of parameters and authenticates the integrity of the signed data also with the identity of the signatory. An algorithm provides the functionality to generate and verify signature. Signature generation uses of a private key to generate a digital signature.

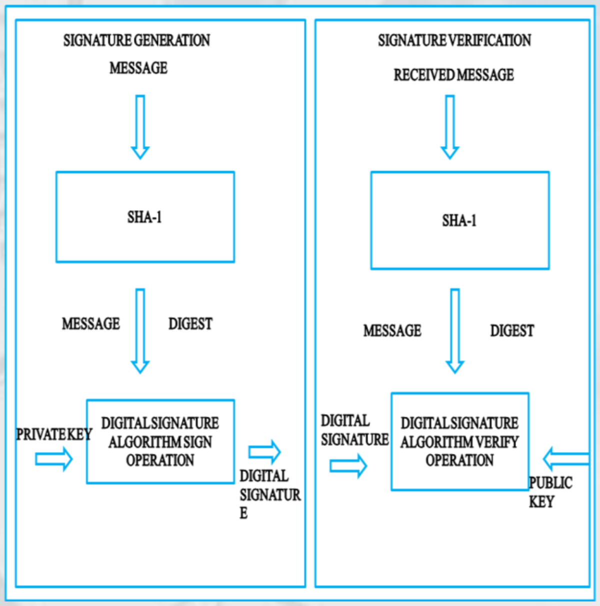

Fig1: Digital Signature Scheme

Signature verification uses of a public key, which corresponds to, but is not the same as, the private key. Each user has a private and public key pair. Public keys are known to the public in general and Private keys are never shared. Anyone can verify the signature of a user by using that user public key. Only the possessor of the user private key can perform signature generation.

Basically a hash function is used for signature generation process to obtain a other version of data, known as message digest (Figure 1). The message digest is then input to the digital signature algorithm to generate the digital signature and sent to the intended verifier along with the message after that the verifier of the message, signature verifies the signature by using the sender's public key.

The same hash function must also be used in the verification process. The hash function is specified in a separate standard, the Secure Hash Standard, FIPS 180-1 [4], [5]. FIPS approved digital signature algorithms must be implemented with the Secure Hash Standard. Similar procedures may be used to generate and verify signatures for stored as well as transmitted data.

The Digital Signature Standard (DSS) uses three algorithms for digital signature generation and verification [1]. The Digital Signature Algorithm (DSA).

2.PROBLEM STATEMENT

SRNN is a highly secure algorithm .The only known way to attack it is to perform a "brute-force" attack on the modulus. This attack can be easily defeated by simply increasing the key size [9]. However, this approach can lead to a number of problems:

Computational Overheads – the computation required to completion public and private key transformations.

Increased key storage requirement – SRNN key storage (private keys and public key) requires significant amounts of memory for storage. Furthermore, Key generation is complex and time consuming (times increase significantly as key sizes increase). In each of the systems (SRNN) considerable computational savings can be made.

In SRNN, to speed up signature verification and encryption we need to make digital signature more secure with small bits (512 bits) for keys generation.

3.PROPOSED ARCHITECTURE

The proposed architecture for the implementation of the digital signature scheme is shown in Figure 2. Basically it consists of two main parts, the Secure Hash Function SHA-1, and a 512-bit SRNN algorithm [6]. For the SRNN implementation the methodology for computing the modular exponentiation is used. This is chosen because it can achieve an appreciable decrease of covered area, and sometimes can increase the time-performance comparing with others methodologies [8]. In fig: 2 initially the sender digest the message with SHA- 1 (160 bit) algorithm and send it to recipient. After the completion of this task sender encrypt the message with public key of SRNN algorithm and then the data is signed with private key of SRNN algorithm.

The verification of digital signature is started after this process with the help of public key at the recipient side. The decryption of digital signature is done in this process which eventually results in the generation of a message.

Finally the two messages are compared at the recipient side. if the messages are matched then the signature is said to be authenticated while the mismatch of the two messages exhibit de-authentication of signature and also called VALID or INVALID of the messages.

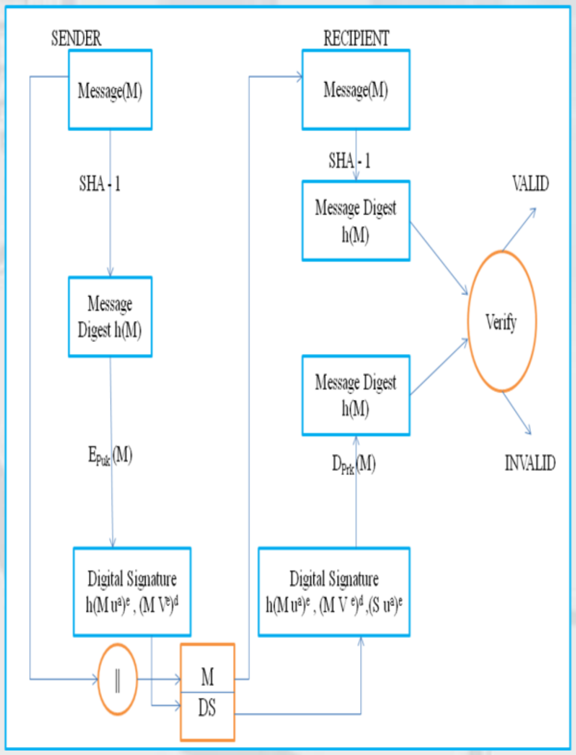

Fig2: Proposed Architecture for process of sign generation and verification.

KEY GENERATION

- Generate two large random primes p and q of approximately equal size such that their product $n = p \times q$ is of the required bit length, such as 1024 bits.
- Compute $n = p \times q$.
- Compute $\phi = (p-1)(q-1)$.
- Choose an integer e, $1 < e < \phi$, such that gcd $(e, \phi) = 1$. Compute the secret exponent d, $1 < d < \phi$, such that $(e \times d) \bmod \phi = 1$.
- Choose short range natural number u randomly such that $u < \phi$.
- Choose another short range natural number a randomly such that $\phi > a > u$ and compute u^a.
- Find d such that $e * d \bmod ((p-1)*(q-1)) = 1$.

The public key is (n, e, u^a) and the private key is (d, a, u).

The generation of SRNN keys is very important for the application. The application maintains a constant named “KEY_SIZE” which is the size of prime numbers p and q (bits). These random numbers p and q never repeated.

Now we select the two short range natural number **u** and **a** and we have two primes p and q, calculating the values of n **($n=p \times q$)** and phi **($\phi = (p-1) \times (q-1)$)**. Next, to determine the value of public key **e** with condition **gcd $(e, \phi) = 1$** is satisfied.

With the help of **e** to compute private key **d** with method **$(e \times d) \bmod \phi = 1$.**

In this key, we obtain the value of **d**, which completes our Key Generation for the SRNN keys. Finally the keys **d,e,n,u^a** are stored in private key as **(d, a, u)** and public key as **(n, e, u^a).**

It is noted that entire application can use a higher or lower bit length of the SRNN modulus n by simply changing the constant KEY_SIZE.

SIGNATURE GENERATION

INPUT: Private Key (d, u, a) and Public key (n, e, u^a) for user (A) to sign, message to be signed S.

OUTPUT : S, signature of M

1) $A = h(M\ u^a)^e \bmod n$

2) $S = (M V^e)^d \bmod n$

Where $V = (u^{\phi-a}) \bmod n$

In signature generation u and a is the natural numbers, V is denoted as a value to compute signature.

SHA – 1, common hash algorithm is used

- Create a message digest of the information to be sent.
- Uses private key to compute signature.
- Sends the signature S to the recipient.

SIGNATURE VERIFICATION

INPUT: Public key (n, e, u^a), message (M) and signature(S) for user (B).
OUTPUT : VALID or INVALID.
1) $B = (S\ u^a)^e \bmod n$
2) If B = (M), Return VALID Else, return INVALID

- User uses public key to compute message from integer.
- Extracts the message digest from the integer.
- Independently compute the message from message digest.

4.RESULTS

Verifying and Signing (let to be two users A and B).

Signing:
User (A) creates a signature out of the message using his private key $S = (M\ V^e)^d \bmod n$ and send the message and the signature to user (B).

Verifying:
User (B) receives M and S. User (B) applies public key to the signature to create a copy of the message $M' = (S\ u^a)^e \bmod n$.

Now user (B) compares the value of M' with the value of M. If the two values are same, User (B) accepts the message otherwise not.

In actual practice, very large prime numbers are used but for this example small numbers will be used to keep the math easy, to prove this algorithm.

5.CONCLUSION

In this paper SRNN algorithm is used for digital signature scheme with 512-bit SRNN algorithm behalf of this algorithm we can secure the communication channel for sender and recipient. We can further use this algorithm for (1024-bit) SRNN algorithm. In this proposed algorithm and proposed signature generation, verification is more secure than RSA algorithm but little slower in speed.

REFERENCES

[1] National Institute of Standards and Technology (NIST), Digital Signature Standard, FIPS PUB 186-2,

[2] Martin Johnsson, "HiperLAN/2 – The Broadband Radio Transmission Technology Operating in the 5 GHz Frequency Band", HiperLAN/2 Global Forum, 1999, Version 1.0 white-paper.

[3] WAP Forum: "Wireless Application Protocol Architecture Specification" and "WAP White Paper".

[4] National Institute of Standards and Technology (NIST), Secure Hash Standard, FIPS PUB 180-1.

[5] Robert D. Silverman, "An Analysis of Shamir's Factoring Device", RSA Laboratories, May 3, 1999.

[6] Wikipedia, the free encyclopedia: Natural Numbers.

[7] Cetin Kaya Koc, "RSA Hardware Implementation", RSA Laboratories, RSA Data Security.

[8]An Efficient Implementation Of The Digital Signature Algorithm P. Kitsos, N. Sklavos and O. Koufopavlou.

[9] Yaun Xue "Public Key Cryptography and RSA Algorithm "Technical notes and papers.

[10] Carnegia Mellon Software Engineering Institute "Public Key Cryptography.

[11] R Gennaro. (2000), "RSA-Based Undeniable Signatures",
Journal of Cryptology, Vol 13, No. 4, pp 397-416.

[12] R Cramer, V Shoup. (2000), "Signature schemes based on the strong RSA assumption", ACM Transactions on Information and System Security, Vol 3, No 3, pp 161-185.

[13] R Gennaro. (2000), "Robust and Efficient Sharing of RSA Functions", Journal of Cryptology, Vol 13, No 2, pp 273-300.

[14] An Efficient Implementation of RSA Digital Signature Algorithm, Ying-yu Cao, Chong Fu.