Secured Document Exchange Using Digital Signature

Wai Zin Aung, Khin Than Mya
University of Computer Studies, Yangon, Myanmar
waizinaung85@gmail.com

Abstract

Strong security notions often introduce strong constraints on the construction of cryptographic schemes: semantic security implies probabilistic encryption, while the resistance to existential forgeries requires redundancy in signature schemes. Some padding has thus been designed in order to provide these minimal requirements to each of them, in order to achieve secure primitives. A common practice to encrypt with RSA is to first apply a padding scheme to the message and then to exponentiate the result with the public exponent, which is called OAEP (Optimal Asymmetric Encryption Padding).This paper proposed this notion of universal padding, OAEP, can also be used the same RSA key-pairs for encryption and decryption, in any trapdoor partial-domain one-way permutation.

Keywords: *Cryptography, RSA, OAEP, Message padding, Digital Signature*

1. Introduction

Cryptography has a long and fascinating history. The most complete non-technical account of the subject is Kahn's The Codebreakers. The book traces cryptography from its initial and limited use by the Egyptians some 4000 years ago, to the twentieth century where it played a crucial role in the outcome of both world wars. Completed in 1963, Kahn's book covers those aspects of the history which were most significant (up to that time) to the development of the subject. Cryptography was used as a tool to protect national secrets and strategies.

In 1976, Diffie and Hellman [6] introduced the concept of public key cryptography. In 1978 Rivest, Shamir, and Adleman discovered the first practical public-key encryption and signature scheme, now referred to as RSA. The RSA scheme is based on another hard mathematical problem, the intractability of factoring large integers. This application of a hard mathematical problem to cryptography revitalized efforts to find more efficient methods to factor.

In this paper, we propose a framework for exchanging documents by applying Optimal Asymmetric Encryption Padding (OAEP). The sender send message by encrypting with OAEP by using public key and digest the encrypted file with secured hash algorithm (SHA-1). The receiver receives encrypted file and signature. He/ She decrypt this file with OAEP by using private key. Furthermore, we analyze the security of these paddings, to be used for both encryption and signature with the same public/private key, where the same primitive (trapdoor one-way permutation) at the same time.

The rest of the paper is organized as follows. In Section 2, we reviews related work and background theory. We show the general architecture of the proposed system in Section 3. Section 4 gives a description of processing OAEP. Section 5 describes experimental results of this system and section 6 draws some conclusion and future work.

2. Related Work and Background

Cryptosystems are considered either weak or strong with the main difference being the length of the keys used by the system. U.S. export controls are showing signs of loosening, but they continue to discourage the export of strong cryptography because of fears that government enemies will use the systems to thwart eavesdropping on illegal or anti-government activities. DES was originally designed so that the supercomputers owned by the National Security Agency (NSA) could be used for cracking purposes, working under the premise that no other supercomputers of their sort are in the public hands or control.

Strong cryptography always produces ciphertext that appears random to standard statistical tests. Because keys are generated for uniqueness using robust random number generators, the likelihood of their discovery approaches zero. Rather than trying to guess a key's value, it's far easier for would be attackers to steal the key from where it's stored, so extra precautions must be taken to guard against such thefts. Any cryptosystem that hasn't been subjected to brutal attacks should be considered suspect. The recent announcement by the National

System (AES) to replace the aging DES system underscores the lengths to which cryptographers will go to build confidence in their cryptosystems.

Victor Shoup presented [7] a new scheme OAEP+, along with a complete proof of security in the random oracle model. OAEP+ is essentially just efficient as OAEP, and even has a tighter security reduction. He observes that appears to be a non-trivial gap in the OAEP security proof and proves that this gap cannot be filled; there can be no standard "black box" security reduction for OAEP. T.EIGamal [6] showed that the security of both systems relies on the difficultly of computing discrete logarithms over finite fields. These estimates imply that the public file size is lager in this scheme than in the RSA scheme, but the difference is at most a factor of two due to the structure of both schemes. Also the size of the ciphertext text is double that of the RSA system. J.-Sebastien Coron et al. [3] showed that PSS allows to safety use the same RSA key-pairs for both encryption and signature for nay trapdoor practical-domain one-way permutation.

This system proposes the strong security of cryptographic schemes. OAEP is used to the same key-pairs with RSA for encryption and decryption in secure in the random oracle model.

3. RSA Cryptosystem

The most common public-key algorithm is the RSA cryptosystem, named for (Rivest, Shamir, and Adleman). RSA uses two exponents, e and d, where e is public and d is private. P is the plaintext and C is the ciphertext. Encryption and decryption use modular exponentiation. Modular exponentiation is feasible in polynomial time using the fast exponentiation algorithm. However, modular logarithm is as hard as factoring the modulus, for which there is no polynomial algorithm yet. In this figure 1, Alice encrypts the message M using Bob's public key P_A and transmits the resulting ciphertext C to Bob. An eavesdropper who captures the transmitted ciphertext gains no information about M. Alice receives C and decrypts it using her secret key to obtain the original message M.

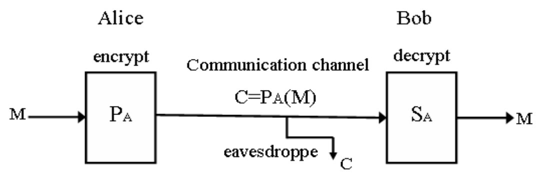

Figure 1. Operations in RSA

3.1. Key Generation Procedure

Bob uses the steps shown in figure 2, to create his public and private key. After key generation, Bob announces the tuple (e,n) as his public key, Bob keeps the integer d as his private key. Bob can discard p, q and ϕ(n). To be secure, the recommended size for each prime or q is 512bits. This makes the size of n, the modulus 1024 bits.

```
RSA_Key_Generation
{
1:      select two large prime p and q such that
        $p \neq q$
2:      $n \leftarrow p \times q$
3:      $\phi(n) \leftarrow (p-1) \times (q-1)$
4:      select e such that $1 < e < \phi(n)$ and e is coprime
        to $\phi(n)$
5:      $d \leftarrow e^{-1} \bmod \phi(n)$
6:      Public_key$\leftarrow$ (e,n)
7:      Private_key$\leftarrow$ d
8:      return Public_key and Private_key
}
```

Figure 2. RSA key generation algorithm

3.2. Encryption Procedure

Anyone can send a message to Bob using his public key. Encryption in RSA can be done using an algorithm with polynomial time complexity, as shown in figure 3 and 4.The size of the plaintext must be less than n, which means that if the size of the plaintext is larger than n, it should be divided into blocks.

```
RSA_Encryption (P,e,n)
{
1:      C$\leftarrow$ Fast_Exponentiation (P,e,n)
2:      return C
}
```

Figure 3. RSA encryption

3.3. Decryption Procedure

Bob can use figure 3, to decrypt the ciphertext message he received. Decryption in RSA can be done using an algorithm with polynomial time complexity. The size of the ciphertext is less than n.

```
RSA_Decryption (C,d,n)
{
1:      P$\leftarrow$ Fast_Exponentiation (C,d,n)
2:      return P
}
```

Figure 4. RSA decryption

4. Implementation of the System Design

In this paper, its system consists of two main components, sender side and receiver side. In sender side, the user selects any original document file to send another user. After selecting document file, RSA key is generated by using secure hash algorithm (SHA-1) .When key file is obtained, this original file is encrypted using OAEP with that key (RSA public key) and compute hash value for that encrypted file. And then, the user sends this encrypted file and hash value to another user as shown in figure 5.

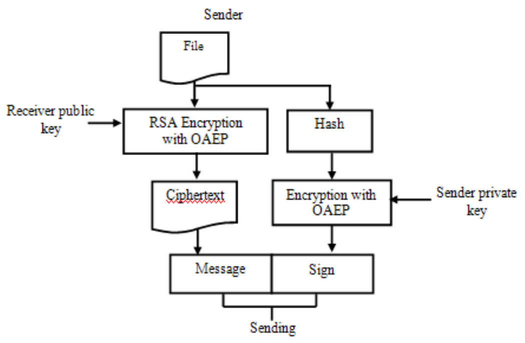

Figure 5. Encryption of System Design

In receiver side as shown in figure 6, receiver receives this encrypted file and hash value. Firstly, the receiver computes hash value for encrypted file and compared hash value with received hash value from sender. If computed hash value is not equal with received hash value, the receiver display "error message" it means someone changed encrypted file, so receiver reject this received encrypted file. If equal, key and decryption process can proceed continuously. To obtain original file, this encrypted file is decrypted by using RSA private key. Finally, receiver can get original file after decryption process complete successfully.

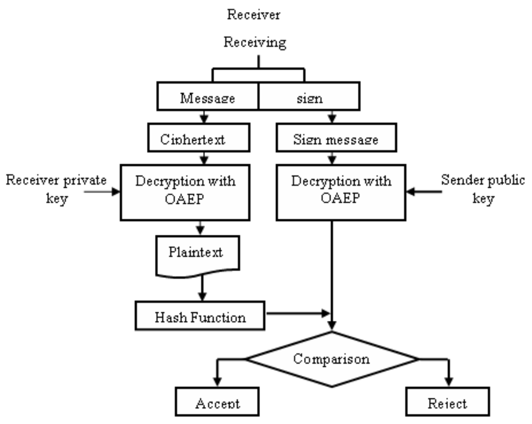

Figure 6. Decryption of System Design

4.1. Optimal Asymmetric Encryption Padding (OAEP)

When instantiated with the RSA function goes by the name RSA-OAEP and is the industry-wide standard for RSA encryption. It has a better message expansion rate. With RSA-OAEP, one can encrypt messages whose bit length is up to just a few hundred bits less than the number of bits in the RSA modulus, a ciphertext whose size is the same as that of the RSA modulus. The operation of OAEP encryption and decryption is shown as figure 7.

Encryption:

- Message padded to m bits
- Random bits r generated to mask padded message
    - Run through hash function G
    - XOR'd with padded message to give P1
- Masked message used to mask the random bits
    - Masked message run through hash function H
    - XOR'd with random bits to give P2
- Masked message and random bits (P1 and P2) encrypted and sent

Decryption:

- Ciphertext decrypted to get masked message and random bits (**P_1** and **P_2**)
- Masked message **P_1** run through hash function **H** and XOR'd with **P_2** to recover ***r***

r run through hash function **G** and XOR'd with **P_1** to recover original padded plaintext

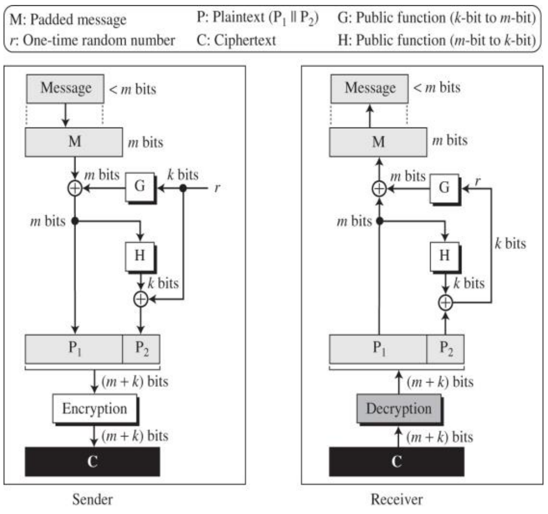

Figure 7. Sending and receiving of message using OAEP

4.2. OAEP Message Padding

OAEP encryption starts by encoding a seed, a hash, padding octets and the secret into an octet string. Masking operations effectively randomize these octets before they are treated as the unsigned binary representation of an integer-the integer used in the RSA modular exponentiation operation. The number of padding octets is chosen so that the encoding consumes one less octet than required for an unsigned binary representation of the modulus. This ensures the integer is less than the modulus as required in RSA. Alternatively, the encoded messages can be considered as an octet string the same length as the modulus, but with the most significant octet set to `00'h.

Figure 8 shows the OAEP padding scheme. OAEP achieved *chosen ciphertext attack* security based on an almost mathematical proof relying on the one-wayness of the RSA function. The mathematical model is called *random oracle model* which models G and H functions which return random independent values.

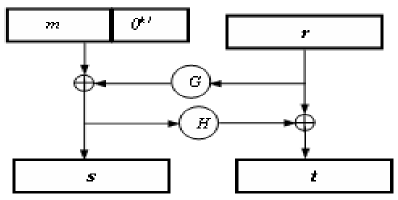

Figure 8. **OAEP Message Padding**

4.3. Padding Algorithm of OAEP

The following algorithm shows the encrypting algorithm. Sender uses the original message (*m*), a random number (*r*) and two functions (G and H).

```
RSA_OAEP_Encrypt (m, e, n, r)
{
    M  ← pad (m)
    P1 ← M ⊕ G(r)
    P2 ← H(P1) ⊕ r
    C  ← Fast_Exponentiation (P1 | P2, e, n)
    return C
}
```

The following algorithm shows the decrypting algorithm. Receiver uses the ciphertext (C), the private key (*d*), modulus (*n*), and two functions G and H.

```
RSA_OAEP_Decrypt (C, d, n)
{
    P ← Fast_Exponentiation (C, d, n)
    P1, P2 ← Split_{m, k} (P)
    r  ← H(P1) ⊕ P2
    M ← P1 ⊕ G(r)
    m ←Extract (M)
    return m
}
```

4.4 Algorithm of OAEP Cryptosystem

We briefly describe the OAEP cryptosystem (κ,ε,D) obtained from a permutation *f*, whose inverse is denoted by g. We need two hash functions G and H:

$$G:\{0,1\}^{k0} \rightarrow \{0,1\}^{k-k0}\, and$$
$$H:\{0,1\}^{k-k0} \rightarrow \{0,1\}^{k0}$$

Then,

κ(1^k): specifies an instance of the function *f*, and of its inverse g. The public key "**pk**" is therefore ***f*** and the secret key "**sk**" is **g**.

- ε_{pk} (m,r): given a message m ∈ $\{0,1\}^n$, and a random value $r \overset{R}{\leftarrow} \{0,1\}^{k0}$, the encryption algorithm ε_{pk} computes
  $$s = (m \,\|\, 0^{k1}) \otimes G(r) \textbf{ and } t = r \otimes H(s)$$
  and outputs the ciphertext c=*f*(s, t).
- D_{sk}(c): thanks to the secret key, the decryption algorithm D_{sk} extracts

  (S, t)=g(c), and next r= t⊕H(s) and
  M= s⊕G(r)

If $[M]_{k1}=0^{k1}$, the algorithm returns $[M]^n$, otherwise it returns "**Reject**"., where $[M]_{k1}$ denotes the k_1 least significant bits of M, while $[M]^n$ denotes the n most significant bits of M.

5. Evaluation Result

In this paper, OAEP proposed a novel trapdoor one way permutation. The trapdoor one-way permutation defines a simple public key encryption system is generated by the public key and the trapdoor is the private key. OAEP has efficiency in both time and message expansion, and its compatibility with more traditional implementation of RSA encryption. This system is implemented using C# programming language and it was tested in Notebook PC with Intel® Core™2 Duo CPU 2.00GHz and 2GB of RAM.

OAEP is so popular is the widespread belief that the scheme is secure in the random oracle model, provided the underlying trapdoor permutation scheme is one way. OAEP can be used to encrypt long messages. The corresponding OAEP-based encryption scheme is plaintext-aware, meaning roughly that an adversary cannot produce a valid ciphertext without actually "knowing" the underlying plaintext. Plaintext awareness of an encryption scheme is closely related to the resistance of the scheme against chosen ciphertext attacks. In this Table 1 shown in runtime of encryption and decryption in various was testing document size.

Table 1. **Runtime of Encryption and Decryption**

Size (KB)	Encryption Time (ms)	Decryption Time (ms)
10	56	66
20	172	313
40	265	430
60	483	476
80	530	638

In this Figure 9 and 10 is shown in encryption and decryption time for various testing document size.

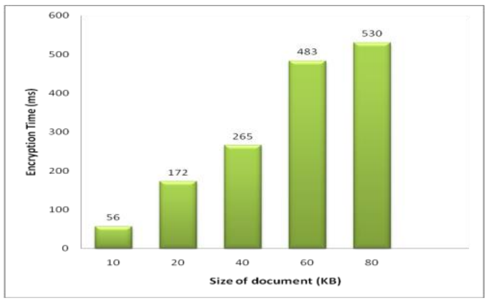

Figure 9. Encryption Time of various document sizes

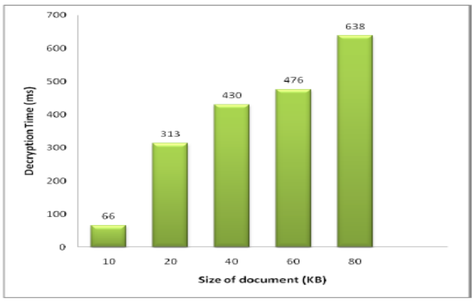

Figure 10. Decryption Time of various document sizes

6. Conclusion

Cryptography is concerned with keeping communications private. This system is tested with various document size such as .txt, .doc, etc. The sender cannot know whether the sending data is received by the receiver or not. So additional recovery and reliability features can be added for more completing the system's performance. Currently, this system padded messages by applying OAEP, and then extend to use others universal padding scheme. This system can extend to encrypt image files. This system can extend to run different machines such as sender site and receiver site. It is the science of writing in secret code and provides the basics for authentication of messages as well as their security and integrity. Asymmetric cryptosystems use one key (public-key) to encrypt a message and a different key (private-key) to decrypt it. OAEP, message padding which could apply for both encryption and decryption. Our conclusion is that one can still trust the security of RSA-OAEP, but the reduction is more costly than the original one. However, for other OAEP applications, more care is needed, since the security does not actually rely on the one-wayness of permutation, only on its partial-domain one-wayness.

References

[1] A. B. Forouzan, Cryptography and Network Security, International Edition , 2008.

[2] B. Chevallier-Mames, D. Hieu Phan and D. Pointcheval, "Optimal Asymmetric Encryption and Signature Paddings", Proceedings of the Conference on Applied Cryptography and Network Security (ACNS '05) (6 – 10 june 2005, New York, USA)

[3] D. Naccache, M. oye, J. -S. Coron and P. Paillier. "Universal Padding Schemes for RSA", In M. Yung, editor, Advance in Cryptograhy, 2002.

[4] J. Stren, " Cryptography and the Methodology of Provable Security", ENS-CNRS, 45rue d'Ulm, 75230 pris CEdex 05, France.

[5] M. Bellare and P. Rogaway, "Optimal Asymmetric Encryption", Proceedings of Eurocrypt'94, LNCS vol. 950, Springer-Verlag, 1994, pp.92-111.

[6] T. EIGamal, "A Public Key Cryptosystem And a Signature Scheme based on Discrete logarithms", Proceedings of Eurocrypt '94, LNCS vol. 950, Springer-Verlag, 1998.

[7] V. Shoup, "OAEP Reconsidered", Proceedings of Crypto 2001, LNCS vol 2139, pp 239-259, 2001. September 18, 2001.

[8] W. Diffie and M. Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory, IT-22 (1978), 472-492.