Digital Signature For Text File

Ayad Ibrahim Abdulsada
Dept. of Computer Science, College of Education, University of Basrah, Basrah, Iraq.
E-mail: mraiadibraheem@yahoo.com

Abstract:

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. This is known as *non-repudiation*, since the signatory cannot easily repudiate the signature at a later time. In this paper, an algorithm for signing text file has been presented. The famous RSA public key algorithm and SHA-l(Secure Hash Algorithm) hash function are used to generate the digital signature of a text file. Our algorithm consist of two parts: signature generation and signature verification. Many experiments tested to examine the security of the presented work.

Keywords: Computer security, Digital signatures, Public key cryptography, Hash functions.

1. Introduction

Digital documents that are exchanged over the Internet can be accessed or modified by a malicious user with relative ease. This creates an important security concern while exchanging multimedia data over the Internet. Multimedia data contains information in the form of audio, video, still images, etc. This increases the need for authentication and verification of document integrity for users of such data. One of the well-known methods used for authentication of digital documents is the public key encryption-based authentication [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(authentication) and the integrity of the data to be verified. Digital signatures may be generated on both stored and transmitted data. Signature generation uses a private key to generate a digital signature; signature verification uses a public key that corresponds to, but is not the same as, the private key. Each signatory possesses a private and public key pair. Public keys may be known by the public; private keys are kept secret. Anyone can verify the signature by employing the signatory's public key. Only the user that possesses the private key can perform signature generation.

In detail, a Signature procedure looks like this: Senders use their message and secret key to calculate the digital signature for the message. Compared to hand-written signatures, digital signatures therefore have the advantage that they also depend on the document to be signed. Signatures from one and the same participant are different unless the signed documents are completely identical. Even inserting a blank in the text would lead to a different signature. The recipient of the message would therefore detect any injury to the message integrity as this would mean that the signature no longer matches the document and is shown to be incorrect when verified. The document is sent to the recipient together with the signature. The recipient can then use the sender's public key, the document and the signature to establish whether or not the signature is correct. Because a signature is about as long as the straight data stream to be signed, the procedure we just described has in practice, however, a decisive disadvantage. The signature is approximately as long as the document itself. To prevent an unnecessary increase in data traffic, and also for reasons of performance, we apply a cryptographic hash function to the document— before signing. The output of the hash function will then be signed [2].

A digital signature algorithm is intended for use in electronic mail, electronic funds transfer, electronic data interchange, software distribution, data storage, and other applications that require data integrity assurance and data origin authentication.

2. Hash Functions

A hash function maps a message of any length to a string of characters with a constant size, the hash value. Cryptographically secure hash functions fulfill the following requirements [3, 4]:

• ***Resistance Against 1st Pre-Image Attacks:*** It should be practically impossible, for a given number, to find a message that has precisely this number as hash value. If we have the hash value H', it's impossible to find message m, so that: H(m) = H'.

• ***Resistance Against 2nd Pre-Image Attacks:*** It should be practically impossible, for a given message, to find another message, which has precisely the same hash value. If we have message ml [and so the hash value H1 = H(ml)], it's impossible to search message m2, so that: H(m2) = H1.

• ***Collision Resistance***: It should be practically impossible to find any two messages with the same hash value (it doesn't matter what hash value). Searched 2 messages ml and m2, so that: H(ml) = H(m2).

Hash functions are normally used to provide the digital fingerprints of files to ensure that the content of the file has not been altered in transit.

2.1 Secure Hash Algorithm (SHA-1) [5]

The Secure Hash Algorithm (SHA) was developed by National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS 180) in 1993; a revised version was issued as FIPS 180-1 in 1995 and is generally referred to as (SHA-l). This algorithm has been well taken and appreciated by experts.

SHA-1 Logic

The algorithm takes as input a message with length of less than 2^{64} bit. And produces as output is 160-bit message digest. The input is processed in 512-bit blocks. The algorithm steps are:

Step 1: Append padding bits. The message is padded so that its length is congruent to 448 modulo 512.

Step 2: Append length. A block of 64-bit is appended to the message. This block contain length of the original message.

Step 3: Initialized buffer. The buffer can be represented as five 32-bit registers (A, B, C, D, E). These registers initialized with the following values:

A=67 45 23 01 D=10 32 54 76
B=ef cd ab 89 E=c3 d2 el f0
C=98 ba dc fe

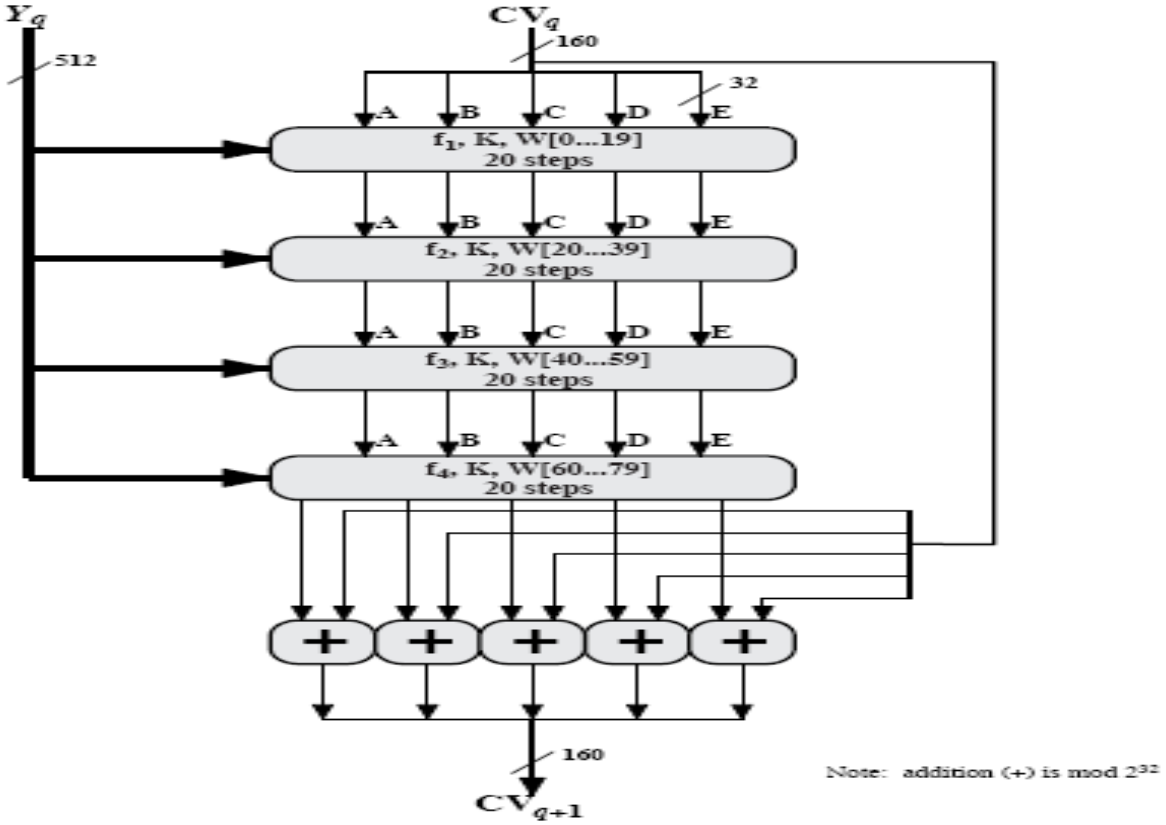


Figure (l): SHA-1 Compression function.

Where Y_q: the current 512-bit being processed.

CV_q: 160-bit, the output of the previous block. CV0=(ABCDE).

CV_{q+1}: 160-bit, the output of the current block.

Wt: a 32-bit word derived from the current 512-bit input block. The first 16 values of Wt are taken directly from the current block. The remaining values are defined as follows:

$$W_t = S^1 (W_{t-16} \text{ XOR } W_{t-14} \text{ XOR } W_{t-8} \text{ XOR } W_{t-3}) \qquad 16 \le t \le 79 \qquad \ldots(1)$$

S^k: rotation of the 32-bit argument by k-bits to the left. +: addition module 2^{32}.

K: constant. Table (1) explain K values.

Table (1): K values.

$0 \le t \le 19$	K_t=5A827999	$[2^{30} * \sqrt{2}]$
$20 \le t \le 39$	K_t=6ED9EBA1	$[2^{30} * \sqrt{3}]$
$40 \le t \le 59$	K_t=8F1BBCDC	$[2^{30} * \sqrt{5}]$
$60 \le t \le 79$	K_t=CA62C1D6	$[2^{30} * \sqrt{10}]$

Each steps from the 80 steps (4-round ***** 20-step) has the following form:

$A,B,C,D,E \leftarrow (E+f(t,B,C,D)+S^5(A)+W_t+K_t),A, S^{30}(B),C,D)$ . . . (2)

The four rounds have a similar structure, but different logical functions, which referred as f_1, f_2, f3, and f_4**.** Table(2) explain the logic of the four functions.

Tab1e (2): Logic of SHA-1 functions.

Step	**Function Name**	**Function value**
$(0 \le t \le 10)$	f_1=f(t,B,C,D)	(B^C)OR(B'^D)
$(20 \le t \le 39)$	f_2=f(t,B,C,D)	B XOR C XOR D
$(40 \le t \le 59)$	f_3=f(t,B,C,D)	(B^C) OR (B^D) OR (C^D)
$(60 \le t \le 79)$	f4=f(t,B,C,D)	B XOR C XOR D

Step 5: Output. After all L 512-bit blocks have been processed, the output from the last stage is the 160-bit message digest (CV_L).

2.2 Signing with Hash Functions

Rather than signing the actual document, the sender now first calculates the hash value of the message and signs this. The recipient also calculates the hash value of the message (the algorithm used must be known), then verifies whether the signature sent with the message is a correct signature of the hash value. If this is the case, the

signature is verified to be correct. This means that the message is authentic, because we have assumed that knowledge of the public key does not enable you to derive the secret key. However, you would need this secret key to sign messages in another name. Some digital signature schemes are based on asymmetric encryption procedures, the most prominent example being the RSA system, which can be used for signing by performing the private key operation on the hash value of the document to be signed.

3. The RSA Procedure

As early as 1978, R**.** Rivest, A**.** Shamir, L**.** Adleman introduced the most important asymmetric cryptography procedure to date. It encrypt and decrypt numbers M less than n. The following algorithm explain the RSA work [6]:

Key Generation
- Select two primes p,q, such that p≠ q. - Calculate N= p*q. - Calculate phi=(p1)*(q1). Phi: Euler phi function. - Select integer e, such that: gcd (phi, e)=l; 1<e<phi. - Calculate d, such that d ≡e1 mod phi. - Public key {e, N} - Private key = {d, N}

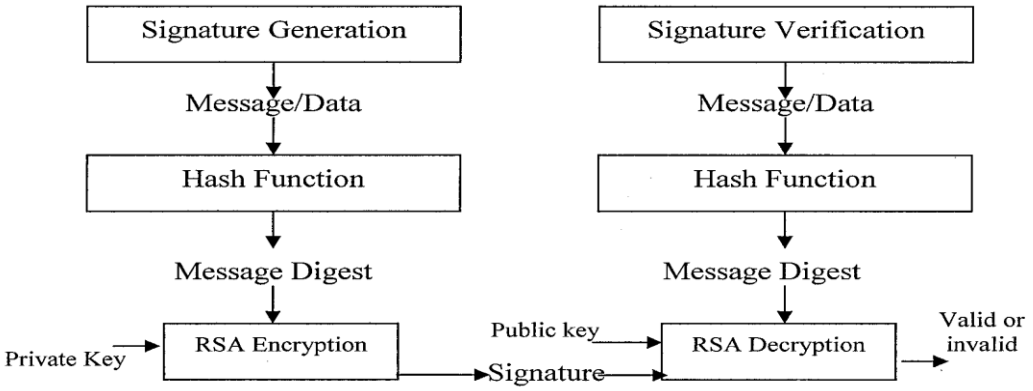
Encryption
Plaintext: M<n Cipher text: $C=M^e \pmod N$

Decryption
Cipher text: C Plaintext: $M=C^d \pmod N$

The size of an RSA key pair is commonly considered to be the length of the modulus *n* in bits. The corresponding RSA private key consists of the same modulus *N* and a private key exponent *d* that depends on *N* and the public key exponent *e(d is the multiplicative inverse of e).* Thus, the RSA private key is the pair of values *(N, d)* and is used to generate digital signatures. In order to provide security for the digital signature process, the two integers p and q, and the private key exponent *d* must be kept secret. The modulus *N* and the public key exponent *e* may be made known to anyone.

4. RSA Digital Signature

Figure (2) explain the basic steps of our presented RSA digital Signature.



Figure(2): RSA Digital Signature.

Our work consist of the following two algorithms:

4.1 Signature Generation

In this algorithm, the sender sign his message by using his private key and send it to the recipient as the following steps:

1- Message Choosing: Determine the message to sign it. In this paper, a text file has been as a message.

2-Hash Function: We use SHA-1 algorithm to compress the message and compute the hash code H, which consist of 160-bit.

3- Key Generation: Choose two prime numbers p, q. Compute the RSA modulus N, such that N=p*q. then compute Euler phi function, where: phi=(p1)*(q1). Select the public key e, where gcd (e, phi)1, and 1<e<phi. Finally compute the secret key d which is the multiplicative inverse (e^{-1}) of e. The d value computed by using extended Euclid algorithm as the following [4]:

EXTENDED EUCLID(*phi*, *e*)
1. (A1, A2, A3)=(1, 0, *phi*);
(B1, B2, B3)=(0, 1, *e*)
2. if B3 = 0
return A3 = gcd(*phi*, *e*); no inverse
3. if B3 = 1
return B3 = gcd(*phi*, *e*); B2 = e^{-1} mod *phi*
4. Q = A3 div B3
5. (T1, T2, T3)=(A1 – Q B1, A2 – Q B2, A3 – Q B3)
6. (A1, A2, A3)=(B1, B2, B3)
7. (B1, B2, B3)=(T1, T2, T3)
8. goto 2

Where d = B3.

4- Pre-encryption: In this step, we initialize the hash code by converting it to bytes vector, to produce 20 byte vector (Vec). Then we convert the above vector into blocks of two numbers. By combining each two bytes together into one block as:

$$block=(byte_1 *256)+byte_2 \quad \ldots (3)$$

we obtain the following 10 blocks vector (Bvector). Notice that each number in Bvector must be less than N (module).

5- Sign Computation: In this step, we use the private key (d) of the sender (signatory) to encrypt each number M in Bvector (hash code) as the following:

$$C=M^d \bmod N \quad \ldots (4)$$

to obtain the 10 numbers sign vector (sign).

6-Sign Appending: the obtained sign is appended in the bottom of the signed text and send to the recipient of that text.

4.2Signature Verification

In this algorithm, the recipient do the steps to verify the sign in the received message by using the public key of the sender:

1- Signature Authentication: The received signature numbers are decrypted first by the public key (e) of the sender(signatory). As the following equation:

$$M=C^e \bmod N \quad \ldots (5)$$

Then each obtained block is converted back to its original two bytes. As the following: $Byte_1$= block/256, $byte_2$= mod(block, 256). The final result of this step is 20 bytes.

2- Hash code Comparison (integrity): Compute the hash code of the received message (text) again and compare its hash code with the received hash code (result of step l). If the two codes are equivalent, then we obtain the following results:

a. The message has not been modified during the transmission, since its hash code equal the hash code of the signed message.

b. Message authentication, since its delivered by the public key of the sender.

c. No source-repudiation, Since the public key of the sender used to produce the correct hash code for the received message.

5. Experimental Results

In the present work, different experiments has been tested to explain how can we sign a text file by RSA digital signature. MATLAB language programming has been used as a language for our tests. We use the following text as a message to be signed:

> what is cryptool ?
> cryptool is a freeware program which enables you to apply and analyses cryptographic mechanisms. cryptool contains an exhaustive online help, which can be understood without deep knowledge in cryptography, therefore no user manual on how to use cryptool is provided cryptool is completely available in English and German cryptool has implemented almost all state-of-the-art crypto functions and allows you to learn about and use, modern and classic cryptography. the methods available include both classic methods (e.g. the caesar encryption algorithm) and modem cryptosystems (for example, the RSA and AES algorithms, as well as algorithms based on elliptic curves).

Experiment1

In this experiment, signature generation has been tested. First we compute the hash code H for our text by using SHA-l hash algorithm. We obtain the following 160- bit: $H = (C6D74E00F627F4B9F231627AB36AE633463913D3)_{Hex}$. Then we use the following parameters for the key generation of RSA algorithm:

- p=2617,q=3541,
- Compute N=p*q=2617*354l=9266797.

- Phi =2616*3540=9260640.
- Select the public key e=37.
- Compute the private key d by using extended Euclid algorithm. We obtain d=l7520l3.

In the Pre-encryption step we obtain:
Vec=[198,215,78,0,246,39,244, 185,242,49,98, 122, 179, 106,230, 51,70, 57, 19,211].
And Bvector = [50903, 19968, 63015, 62649, 62001, 25210, 45930,5893 1, 17977, 5075].
In the Sign computation step, we obtain the following:
Sign = [3908160, 9149082, 3252908, 8721426,1981821, 1803332, 2734167, 5217033, 2461338, 430770].
the generated signature is appended to the text file to obtain the signed message.

Experiment2

In this experiment, we verify the message by the public key of the sender (e). We find that the hash code of the received message and the hash code of the signature both are equal: [198,215,78,0,246,39,244, 185,242,49,98, 122,179, 106,230,51,70, 57, 19,211]. So, the signature is correct and the message is authenticated and not altered during the transmission. If we use another public key such as e=36.,then we obtain the following hash code: [15563, 102,19720, 63, 32405, 201, 20939, 150, 8094, 150, 7649, 228, 17126, 210, 25343, 67, 3762, 215, 11217, 215]. 'Which is very different from the original hash code. So, the received message is not authorized and the signature is incorrect.

Experiment3

In this experiment, we test the sensitivity of the hash function SHA- 1 by altering only 1-byte of the message during the transmission. We compare the hash code of our text and the hash code of the following text (only remove'?' from the first line):

> what is cryptool
> cryptool is a freeware program which enables you to apply and analyses cryptographic mechanisms. cryptool contains an exhaustive online help, which can be understood without deep knowledge in cryptography, therefore no user manual on how to use cryptool is provided cryptool is completely available in English and German cryptool has implemented almost all state-of-the-art crypto functions and allows you to learn about and use, modern and classic cryptography. the methods available include both classic methods (e.g. the caesar encryption algorithm) and modem cryptosystems (for example, the RSA and AES algorithms, as well as algorithms based on elliptic curves).

We obtain the following hash code:
[C3397C1C82DD008E2913 1 16F27671766407F1250]$_{Hex}$

6.Conclusions

The security strength associated with the RSA digital signature process is no greater than the minimum of the security strength associated with the bit length of the modulus and the security strength of the hash function that is employed. It is recommended that the security strength of the modulus and the security strength of the hash function be the same unless an agreement has been made between

participating entities to use a stronger hash function. A hash function that provides a lower security strength than the security strength associated with the bit length of the modulus ordinarily should not be used, since this would reduce the security strength of the digital signature process to a level no greater than that provided by the hash function.

References

[1]B. Schneier, "Applied Cryptography". John Wiley & Sons, 1996.

[2] FIPS PUB 186-3, “Digital Signature Standard (DSS)”, U.S. Department of Commerce/N.I.S.T. , 2009.

[3] FIPS PUB 180-2, “Secure Hash Standard (SHS)”, U.S. Department of Commerce/N.I.S.T., August 1, 2002.

[4] FIPS PUB 186,” Entity Authentication Using Public Key Cryptography”, U.S. Department of Commerce/N.I.S.T. ,February 18, 1997.

[5] William Stalling, Cryptography and Network Security Principles and Practice, 3’ Edition, Prentice Hall, 2003.

[6] Rivest R. L., Shamir A. and Adleman L.,”A Method for Obtaining Digital Signatures and Public Key Cryptosystems”, Commun. ACM, Vol 21 , pp. 120-126, April 1978.

التوقيع الرقمي لملف نصي

أياد إبراهيم عبد السادة

قسم علوم الحاسبات, كلية التربية, جامعة البصرة, البصرة, العراق.

المستخلص

تستخدم التواقيع الرقمية لتحرّي التعديلات غير المرخصة على البيانات ولإثبات هوية الشخص الموّقع للبيانات. بالإضافة إلى ذلك فان مستلم البيانات الموقعة يمكنه أن يستخدم التوقيع الرقمي كدليل بان البيانات جاءت فعلا من الشخص المدعى. وهذا ما يعرف بعدم الإنكار, لان الموقع لا يستطيع بسهولة إنكار إرساله البيانات الموقعة. تم في هذا البحث تقديم خوارزمية لتوقيع ملف نصي. تعتمد عملية توليد التوقيع الرقمي على خوارزميتي RSA الشهيرة للتشفير بالمفتاح المعلن وخوارزمية الاختزال SHA-1 . تتكون خوارزمية التوقيع الرقمي من جزأين: توليد التوقيع الرقمي وفحص التوقيع الرقمي. تم إجراء عدة تجارب لاختبار أمنية الطريقة المستخدمة.

الكلمات المفتاحية: أمنية الحاسوب, التوقيع الرقمي, التشفير بالمفتاح المعلن, دوال الاختزال.