2009 International Conference on Machine Learning and Computing
IPCSIT vol.3 (2011) © (2011) IACSIT Press, Singapore

A New Digital Signature Algorithm

Erfaneh Noorouzi1 [1+] ,AMIR REZA ESTAKHRIAN HAGHIGHI[2] ,Farzad Peyravi[3], Ahmad Khadem zadeh[4]

[1,2]Islamic Azad University, Sepidan Branch, Sepidan, Iran
[3]Islamic Azad University, Marvdasht Branch, Marvdasht, Iran
[4]Iran Telecom Research Center, Tehran, Iran

Abstract. Digital signature schemes are commonly used as primitives in cryptography protocols. Some applications like multi agent systems transfer messages with low size and capacity. In this paper we introduced a new digital signature which works very well for such applications which have low file size for sending. The new hash function generates dynamic and smaller size of bits which depends on each bytes of message. A simple mechanism for hashing the message and encryption is one of advantages of suggestive algorithms. The main function which is used for hashing is bitwise Or and Multiply functions. Testing new algorithms showed that its hashed file size is 4% of the original file in messages with size lower than 1600 bytes. This algorithm can be used in applications which have low file size for sending and want simple and fast algorithms for generating digital signature.

Keywords: Digital Signature, Hash Algorithms, Symbolized Message, Security

1. Introduction

Emerging applications like electronic commerce and secure communications over open networks have made clear the fundamental role of public key cryptosystem as unique security solutions [1, 2]. A Digital Signature is an important type of authentication in a public-key (or asymmetric) cryptographic system and it is in wide use [3-5] .

A digital signature is a checksum which depends on the time period during which it was produced [2, 6, 7] . It depends on all bits of a transmitted message and also on a secret key but which can be checked without knowledge of the secret key. A digital signature is computed using a set of rules and a set of parameters such that the identity of the signatory and integrity of the data can be verified [8-10].

Cryptographers have been studying electronic signature technologies for decades since the discovery of one-way functions [11]. Several electronic signature [12] schemes are (mathematically) proved to be secure under some complexity theoretical assumptions. In [13] they proposed a simple server-based electronic signature system in which a small number of common private keys were used. Another work was done on designing an off-line signature verification system based on a displacement extraction method [14] in which a questionable signature is compared with a corresponding authentic one .

In this paper we proposed a digital signature algorithm in which the new hash function generates dynamic and smaller size of bytes and a simple and fast generating digital signature which will append at the end of message.

2. Digital Signature

[+] Corresponding author. Fax:00987116470540,Tel:00989177870053,
Email addres: Erfaneh_Noorouzi@iausepidan.ac.ir

In 1976, [11] W. Diffie and M. Hellman studying the manner of key distribution were succeeded in designing general key systems. They also showed how the validity of message by these systems can be covered. They called it digital signature. The idea of digital signature has derived from handmade one, so digital signature should have its all features. The main features of handmade signature are simple production, simple identification and difficult generation [12, 15, 16] .

General Key coding can be used both for confident ability and digital signature [2, 11]. For make confident ability, encryption key is private so that, everyone can code a message. Generation of digital signature is based on keeping encryption key private and decryption key is public. Digital signature depends on some parameter, like person who signature and data that should be visited [17].

2.1. Hash algorithm

In the digital signature a little message is elicited from every document with a few bytes. These details with a few bytes influence on one by one bytes of message and their place in text [12, 18, 19] . By this way every type of general and partial change will cause big change in the details. After the eliciting, details of message will be coded with particular key and the result attached with message. Suppose that we have b message bits as internal and we want to gain the summary of the message. b is ordinal number and positive and also it can have zero or more bits. Here is the summery of hash algorithm.

$$F(X,Y,Z) = XY \vee not(X)Z \tag{1}$$

$$H(X,Y,Z) = X \ xor \ \mathrm{Y} \ xor \ Z \tag{2}$$

$$G(X,Y,Z) = XZ \vee Ynot(Z) \tag{3}$$

$$I(X,Y,Z) = Y \ xor \ (\mathrm{X} \vee not(Z)) \tag{4}$$

3. The suggestive algorithm

In digital signature algorithms [2, 18, 19] , the abstract of message is extracted by use of some complex hash algorithms. Hashing algorithms should be dependent of all bytes of the message and their positions. Then abstract is encrypted by private key and appended to end of message. At the other side, the receiver again extracts an abstract of message by use of sender hashing algorithm. Receiver compares his abstract with abstract of message, which is decrypted with public key of receiver. If it does not match, the message is changes by some one. Figure 1 shows the summery of a digital signature.

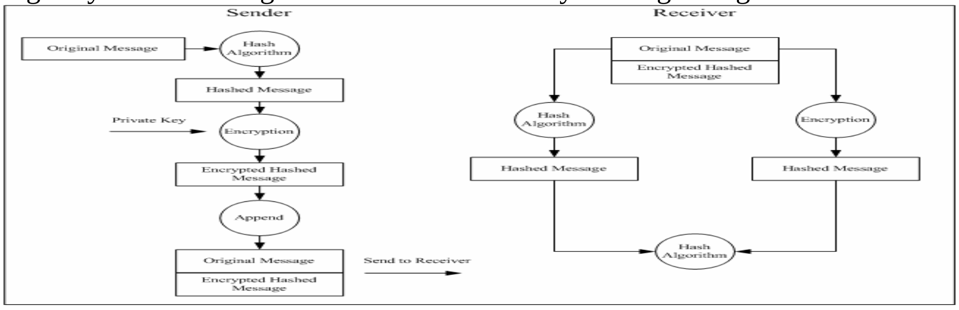
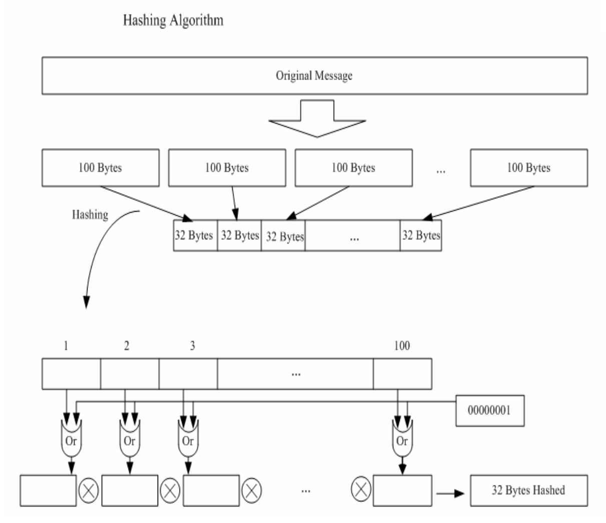


Figure 1. Summery of digital signature

We do the operation on block of 100 bytes. Each byte is read from file and does OR functions with (00000001). Then we put the result into the variable character. After the processing of first 100 bytes, we multiply all of the result ORed bytes and keep them in a 32 bits variable. We use the 32 highest bits of the multiplication of the result. After that the next 100 bytes will be processed with the same operation. It doesn't matter if size of the message is not multiple of 100. So that each 100 bytes of message is summarized in 32 bits of hashed message. By this step we use hash algorithm to generate an abstract of the message which is dependent on all bytes of the message and their positions. Figure 2 summarizes the hash algorithm.

Next step in our algorithm is encryption of the abstract of the message with private key of the sender. A 16 bytes private key is used for encryption. We have two arrays of characters, one is the key named $Key[0..15]$ and other is hashed message $H[0...n]$. In encoding operation, the first byte of hashed message

$H[0]$ will be XOR with the first byte of the key ($Key[0]$) and will be kept in first byte of the key ($Key[0]$). Then again the $H[0]$ will be XORed with $Key[1]$ and result will be replaced with $Key[1]$.



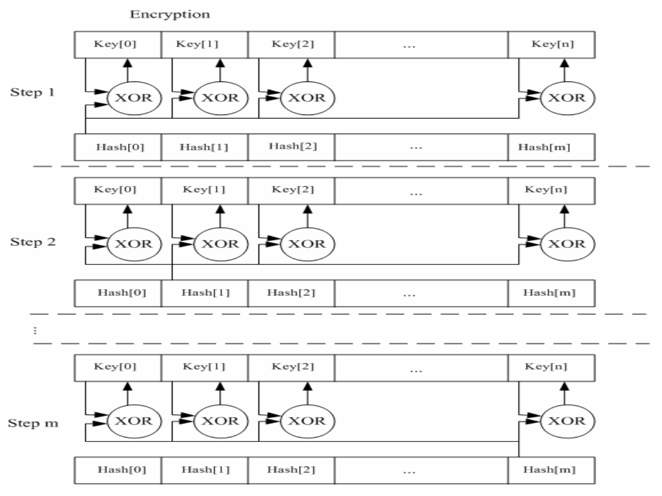


Figure 2. Summery of hashing in suggestive algorithm

Figure 3. Summery of encryption in suggestive algorithm

For convenience of using and testing our algorithm, we convert the array to the base of 16. At the receiver side, the message and hashed one will be separated. At first we generate a hashed form the received message and at the other side we decrypt the sign by the receiver's public key. So here a comparison of hashed message and decrypted sign can show whether any changes has been done on the original message or not. By this comparison we can specify any attack.

4. Experimental Result

Our suggestive algorithm is customized for any type of file such as .doc, .pdf, .txt and etc. For our experiments we used 100 pdf files which 50 of them were English and 50 of them were Farsi. We downloaded most of our test data from papers in Elsevier web site and Farsi papers from a Persian site2 which has the abstract and Farsi papers of PhD. students. Size of our files was variable and it ranges from 2 kb to 1804 kb.

4.1. Size of Hashed Files

One of advantages of our algorithms in comparison with other algorithms is that our algorithm generates a dynamic sized hash file. It means that our algorithms hashed a file and its size is dependent directly on size of original file so that the size of hashed file it is not constant. Fig4 shows the size of file and size of hashed files. It illustrates that our algorithms works very good with files with low size. The average of hashed size is 8.51% of the size of original file.

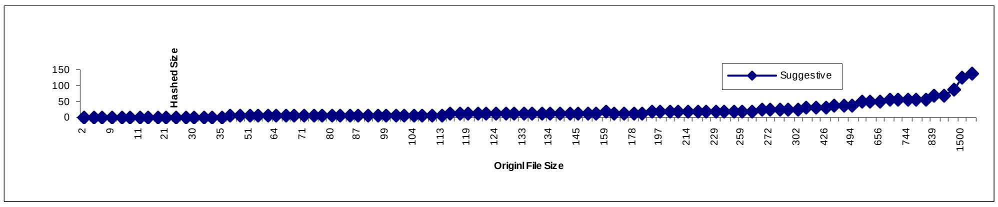


Figure 4. Size of 100 hashed files

Figure 4 illustrates that our algorithm uses a dynamic size of hash files and in files with lower size it works good by a simple and fast generating hash file.

Another comparison that we have done was on files lower than 800, 1000 and 1600 bytes. In our algorithm [2, 5] each 100 bytes hash to 4 bytes and in MD5 to 32, SHA-1 to 40 and SHA-2 to 64 bytes, so

2 www.irandoc.ir

that our algorithm works better in files lower than 1600 bytes than to SHA-2. We have done these operations because in SHA-2, the size of hashed file is 64 bytes, but in our algorithm if the size of file is 1600 then size of hash file will be 64 bytes and lower files have lower size.

In table shows average of hashed file size respect of the original file size. By this result we conclude that our algorithm generates hashed file with lower size than other algorithms. It is very applicable for applications which want to send messages with lower size.

Table 1. Comparison of 4 algorithms in different size files

File Size	MD5	SHA-1	SHA-2	Our
Size <= 800	10.87%	13.59%	21.74%	4.00%
Size <= 1000	9.37%	11.72%	18.75%	4.00%
Size <= 1600	6.76%	8.45%	13.52%	4.00%

Table 2. Comparison of 4 algorithms

Algorithm	Time	Functional	State
MD5	O(n)	Medium	Attacked
SHA-1	O(n)	Fairly high	Attacked
SHA-2	O(n)	High	Running
Our Algorithm	Klog(n)	Low	Running

4.2. Time Complexity of Algorithm

Here we want to compare number of operations which will be done in our suggestive hash algorithm and compare with other hash algorithm. Our suggestive algorithm works on block of 100 bytes of data. It divides the original file into block of 100 bytes and it’s not important that size of file is dividable by 100. So that the last block of data may have one to 100 bytes of data. In each block we have an iteration to do our hash operations, but in other hashing algorithms in each block of data, more than one iteration will be done. In this algorithm, the operation complexity is very low and time complexity will be O(K log n). Log n is using since in each process, 100 bytes of the total file is reduced and K will be used as a parameter for our algorithm. So that, our suggestive algorithm uses simple and fast functions to generate digital signatures, it is a fundamental advantage which could help applications want to be simple, fast and secure.

4.3. Algorithm against Attacks

In digital signature algorithms, if a bit of original file changes so that the hashed file and encrypted message should be changed. In this section we show the changes and its effect on hashed file and encrypted message. We changed our test data and observed that their hashed files and encrypted message will be changed too.

Table 3. Algorithm against Attacks

Original File	Hashed File
in the name of god	f938d14d
in mhe name of god	16213d85
in the nams of god	26bc84b0
in the name pf god	4e92a015
bn the name of god	001e2e4f

In table 3 we showed that if the original file is changed then the hashed fie will be changed too. Table 3 showed some simple example of our tests. We examined the algorithm on our test data. We changed our test data by shifting data, rotating and changing the character of each byte of our data.

5. Conclusion

A digital signature is computed using a set of rules and a set of parameters such that the identity of the signatory and integrity of the data can be verified. A hash function is used in the signature generation process to obtain a condensed version of data. These details with a few byte influence one by one bytes and their place in text so that a partial change will cause big change in the details.

In this paper, we introduced a new digital signature algorithm which generates the hashed file with dynamic size. By dynamic it means that the hash function result depends on size of message. Our algorithm introduces a more simple mechanism for hashing and encryption.

Our algorithm works with all types of files such as .doc, .pdf, .txt and other types of files. We showed that our hashing algorithms works better than other hash algorithms for files with lower size. Our suggestive algorithm is applicable for applications that send messages with low size such as multi agent systems in which agents send massages that is few bytes.

Our future work is testing our algorithm under other attacks and proves it. We want to model check it by an actor based language such as rebeca. We can apply our algorithm

6. References

[1] P. Kitsos, N. Sklavos, and O. Koufopavlou, "An efficient implementation of the digital signature algorithm," *Electronics, Circuits and Systems*, vol. 3, pp. 1151- 1154, 2002.

[2] D. E. Denning, Cryptography and Data Security: *Addison-Wesley Publishing Co*, July 1998.

[3] W. C. Cheng, C.-F. Chou, and L. Golubchik, "Performance of Batch-based Digital Signatures," presented at the 10th IEEE Int'l Symp. *on Modeling, Analysis, & Simulation of Computer & Telecommunications Systems* (MASCOTS' 02), 2002.

[4] R. M. Gray, Entropy and Information Theory: ew York: *Springer-Verlag*, 1990.

[5] B. Schneier, Applied Cryptography: *Wiley,* 1996.

[6] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithm," *IEEE Transactions on Information Theory,* vol. 31, pp. 1469-1485, 1985.

[7] S. Even, O. Goldreich, and S. Micali, "On-line/off-line digital signatures," *presented at Proceedings of CRYPTO'89,*, 1990.

[8] R. C. Merkle, "A digital signature based on a conventional encryption function," *presented at Proceedings of CRYPTO'87, 1988.*

[9] "Biometrics: the Future of Identification," *IEEE Computer*, vol. 33, pp. 46–81, 2000.

[10] R. Gennaro and P. Rohatgi, "How to sign digital streams," presented at Proceedings of CRYPTO'97, *Santa Barbara,CA,* 1997.

[11] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on information theory,* vol. 22, pp. 644-654, 1976.

[12] A. J. Menezes, P. C. v. Oorschot, and S. A. Vanstone, Handbook of applied cryptography: *CRC Press*, 1996.

[13] A. Buldas and M. Saarepera, "Electronic Signature System with Small Number of Private Keys," *presented at 2nd Annual PKI Research Workshop,* 2003.

[14] Y. Mizukami, M. Yoshimura, H. Miike, and I. Yoshimura, "An Off-line Signature Verification System Using an Extracted Displacement Function," 2004.

[15] M. Mambo, K. Usuda, and E. Okamoto, "Proxy signatures: delegation of the power to sign message," *IEICE Transactions Fundamentals,* vol. 9, pp. 1338-1354, 2002.

[16] A.Asokan, G.Tsudik, and M.Waidner, "Server supported digital signatures," *presented at proceedings of ESORICS'96*, Rome, Italy, 1996.

[17] S. J. Hwan and C.-C. Chen, "New multi-proxy multi signature schemes," *Applied Mathematics and Computation,* vol. 147, pp. 57-67, 2004.

[18] C. Chang and Y. F. Chang, "Signing a digital signature withouting using one- way hash functions and message

redundancy schemes," *IEEE Trans*, vol. 8, pp. 485-487, 2004.

[19] R. L. Rivest, A. Shamir, and L. M. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Communications of the ACM*, vol. 21, pp. 120- 126, 2003.