Volume 5, Issue 2, February 2015 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

An Efficient Implementation of Digital Signature

Chiranjib Dutta, Swati Sarkar, Animesh Kar
Department of Computer Application
Guru Nanak Institute of Technology,
Kolkata, India

Abstract: ***Cryptographic hash function plays an important role in the world of cryptography. Cryptographers have been studying digital signature technologies for decades since the discovery of one-way functions. Some applications like multi agent systems transfer messages with low size and capacity. We introduce a new digital signature scheme which works on applications which have less file size for sending. The new hash function generates smaller and variable size of hashed/abstract which depends on each byte of message. The main function which is used for hashing is bitwise XOR functions. This algorithm can be used in applications which have less file size for sending and want simple and fast algorithms for generating digital signature.***

Keywords – Digital Signature, Hash Function, Hash File, Non-repudiation, Multi Agent Work, Encryption

I. INTRODUCTION

The advancement of communication technology and the inevitable uses of digital systems have led to the development of various sophisticated tools that use a framework for the authentication of computer-based information with the concept from both the legal and computer security fields. Digital signatures enable people to sign digital documents like a handwritten signature that fulfill the compelling attributes of handwritten signatures such as authentic, unaltered and non repudiated.

A digital signature algorithm is a public key cryptographic algorithm designed to protect the authenticity of a digital message/document. A message is signed by the private key to produce a signature and the signature is verified against the message by the corresponding public key. Thus any party can verify the signatures but only one party with the private key can sign the messages. A valid digital signature gives the recipient(s) reason to believe that the message was created by a known sender who possesses the private key and that it was not altered in transit. A digital signature is an electronic analogue of a written signature in that the digital signature can be used in proving to the recipient or a third party that the message was, in fact, signed by the originator. Digital signatures may also be generated for stored data and programs so that the integrity of the data and programs may be verified at any later time.

Digital signatures have many applications in information security, including authentication, data integrity, and non-repudiation. Digital signatures are used widely in e-commerce applications, in banking applications, in software distribution and in other cases where jurisdiction is involved and it is important to detect forgery or tampering.

Though there are a numerous number of digital signature algorithms in research literature, only three algorithms have been standardized by the National Institute of Standards and Technology (NIST) and have been widely used in almost all commercial applications. These are RSA signature scheme, the DSA and the ECDSA. In this paper we introduce a new digital signature which works on applications that have low file size for sending. The new hash function generates smaller size of bits which is dependent on each byte of the message and their positions. Our attempt is to make the readers familiar with the concepts related to the digital signature and give them an idea of usefulness of a digital signature in the world of electronic information exchange.

II. RELATED WORK

About four decades (1976) back two mathematicians Whitfield Diffie and Martin Hellman first described the notion of a digital signature scheme, Soon afterwards, Ronald Rivest, Adi Shamir, and Len Adleman invented the RSA algorithm, which could be used to produce primitive digital signatures. In the RSA Signature Scheme proposed combine signing and public-key encryption. ElGamal Signature Scheme, described in a 1985 paper. A modification of this scheme has been adopted as a digital signature standard by the National Institute of Standards and Technology (NIST). The ElGamal Scheme is designed specifically for the purpose of signatures, as opposed to RSA, which can be used both as a public-key cryptosystem and a signature scheme. The ElGamal Signature Scheme is non-deterministic, as was the ElGamal Public-key Cryptosystem. This means that there are many valid signatures for any given message. In 1988, Shafi Goldwasser, Silvio Micali, and Ronald Rivest became the first to rigorously define the security requirements of digital signature schemes. They described a hierarchy of attack models for signature schemes, and also present the GMR signature scheme, the first that can be proven to prevent even an existential forgery against a chosen message attack. The first widely marketed software package to offer digital signature was Lotus Notes 1.0, released in 1989, which used the RSA algorithm.

III. WHY DIGITAL SIGNATURE

The purpose of a digital signature is the same as your handwritten signature. Instead of using pen and paper, a digital signature uses digital keys (public-key cryptology). Like the pen and paper method, a digital signature attaches the identity of the signer to the document and records a binding commitment to the document. Compared to a hand-written signature, it is considered significantly more difficult to forge a digital signature, provided the keys used to create it are kept secure. The real value is in avoiding the paper and keeping your data electronic. As the computerized message systems replacing the physical transport of paper and ink documents, the organizations are moving away from paper documents with ink signatures or authenticity stamps. A digital signature algorithm allows an entity to authenticate the integrity of signed data and the identity of the signatory. The recipient of a signed message can use a digital signature as evidence in demonstrating to a third party that the signature was, in fact, generated by the claimed signatory. A digital signature algorithm is intended for use in to provide added assurances of the evidence to provenance, identity and status of an electronic document as well as acknowledging informed consent and approval by a signatory. Below are some common reasons for applying a digital signature to communications:

3.1 Authentication

Although messages may often include information about the entity sending a message, that information may not be accurate. Digital signatures can be used to authenticate the source of messages. When ownership of a digital signature secret key is bound to a specific user, a valid signature shows that the message was sent by that user. The importance of high confidence in sender authenticity is especially obvious in a financial context. For example, suppose a bank's branch office sends instructions to the central office requesting a change in the balance of an account. If the central office is not convinced that such a message is truly sent from an authorized source, acting on such a request could be a grave mistake

3.2 Integrity

In many scenarios, the sender and receiver of a message may have a need for confidence that the message has not been altered during transmission. Although encryption hides the contents of a message, it may be possible to change an encrypted message without understanding it. However, if a message is digitally signed, any change in the message after signature will invalidate the signature. Furthermore, there is no efficient way to modify a message and its signature to produce a new message with a valid signature, because this is still considered to be computationally infeasible by most cryptographic hash functions.

3.3 Non-repudiation

Non-repudiation, or more specifically non-repudiation of origin, is an important aspect of digital signatures. By this property, an entity that has signed some information cannot at a later time deny having signed it. Similarly access to the public key only does not enable fraudulent party to fake a valid signature.

IV. PROPOSED WORK

4.1 Introduction

A digital signature is a checksum which depends on each bit of a transmitted message and also on a private key but which can be checked without knowledge of the private key In this paper we introduce a new digital signature scheme which works on applications which have less file size for sending. The new hash function generates smaller and variable size of hashed/abstract which depends on each byte of message. A simple mechanism for hashing of the message and encryption is one of advantages of suggestive algorithms. This algorithm can be used in applications which have less file size for sending and want simple and fast algorithms for generating digital signature.

4.2 Proposed Technique

The goal of our algorithm is to support a simple and fast mechanism to produce digital signature. Our proposed algorithm introduces a simple mechanism for hashing or abstracting the message and a new technique in coding the abstract message to facilitate generating a unique index for each input file. The most important function in this algorithm is the hashing function.

4.3 Hash Function

The whole message is chopped into N blocks each of 100 bytes. On necessity padding is done in the last block of the message. In the proposed algorithm first we do operation on the first block of 100 bytes of the message.

Byte No 1 to 12 is XORed with corresponding Byte No 51 to 62 of the block and again XORed results are XORed and the result is stored in the 1st byte of a 32 bits variables. Similarly Byte No 13 to 24 is XORed with corresponding Byte No 63 to 74 of the block and the XORed results are XORed again and result is stored in the 2nd byte of the 32 bits variable. Similar actions are taken for Byte No 25 to 36 with the corresponding Byte No 75 to 86 and the result is stored in the 3rd Byte of the 32 bits variable and for Byte No 37 to 48 with Byte No 87 to 98 and the result is stored in the 4th byte of the 32 bits variable. Byte No 99 & 100 are XORed and the result is XORed with the 1st Byte of the 32 bits variable and the result is stored in the 1st byte of the 32 bits variable and Byte No 49 & 50 is XORed and the result is XORed with the last byte of the 32 bits variable and the result is also stored in the 4th byte of the 32 bits variable.

So that each 100 bytes of message is summarized in 32 bits of hashed message. And the overall hashed is stored as H [0, 1, 2…m]; where m=4XN and N is Number of Blocks of the whole 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. One of advantages of our algorithm 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. The following figure summarizes the hash algorithm.

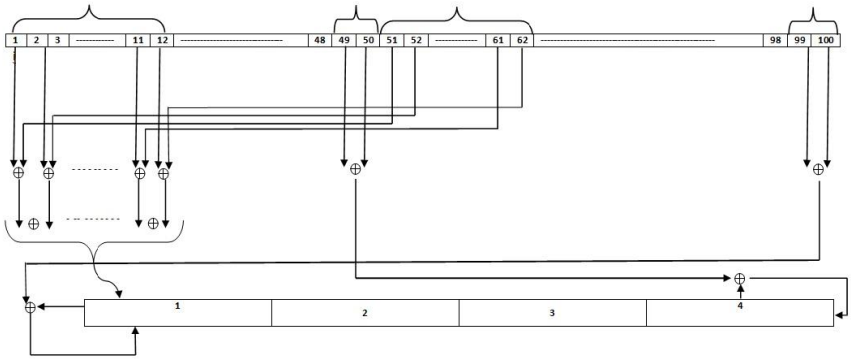

Fig-1

4.4 Signature Creation

We generate 512 bit public and private RSA key pair and use to encrypt and decrypt a random message. Next step in our algorithm is encryption of the hashed/abstract of the message with private key of the sender. A 64 bytes private key is used for encryption. We have two arrays of characters, one is the key named *Key [1...64]* and other is hashed message *H [1...m]*, where m=4XN and N=No of Blocks each of size 100 bytes. In encoding operation, the first byte of hashed message *H [1]* will be XORed with the first byte of the key (*Key [1]*) and will be kept in first byte of the key (*Key [1]*). Then again the *H [1]* will be XORed with second byte of the key *(Key [2])* and result will be replaced with *Key [2]*. The process will repeat for all *Key [...]* with *H [1]*.After the processing of each byte of *Key [1…64]*, we XORed all resultant *Key [...]* and keep the result in *H [1]*. We need to do the procedure because we want to modify the hashed file and repeat the iteration to the end of the abstract/hashed of the message. Then we do the same operation for the second byte (*H [2]*) of the hashed message *H [1…m]*. These iterations are done until the end of hashed message. The following figures depict the proposed technique for signing a random low size message.

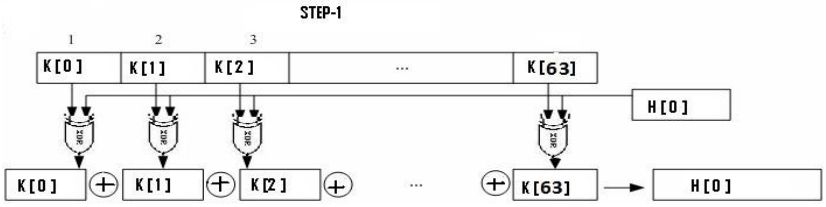

Fig-2

V. RESULT ANALYSIS

Our proposed algorithm is tested for different type of files such as .pdf, .txt etc. For our experiments we use 100 text files of different sizes and the file size ranges from 1KB to 1600 KB

5.1 Size of Hashed Files

One of the advantages of our algorithm in comparison with other existing algorithms is that our algorithm generates a variable sized hash file. Our algorithms hashed a file and its size depends directly on the size of original file so that the size of hashed file is not constant. The following figure shows a comparison of our algorithm with the existing algorithms in size of message file and size of the hashed files. It illustrates that our algorithm works very good with files with low size. The average of hashed size is 4% of the size of original file.

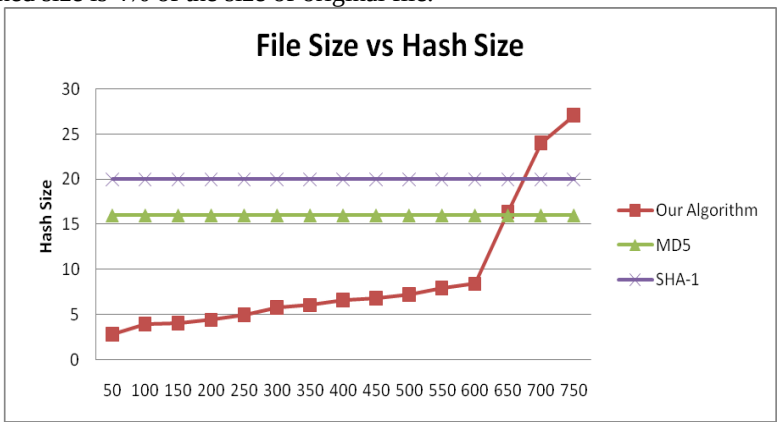

Fig-3

In our algorithm each 100 bytes message hashed to 4 bytes and in MD5 to 16, SHA-1 to 20 and SHA-2 to 64 bytes, so that our algorithm works better in files lower than 400 bytes, 500 bytes and 1600 bytes than MD5, SHA-1 and SHA-2 respectively. By this result we conclude that our algorithm generates hashed file with lower size than other algorithms. It is very much favorable for applications which want to send messages with lower size.

5.2 Time Complexity of Algorithm

In our algorithm the original file is chopped into blocks each of 100 bytes. In each block we have only one iteration to do our hash operations, but in other hashing algorithms in each block of data, more than one iterations are performed.

In this algorithm, the operation complexity is very low and for each block of message about 100 XORed operations are required to hash it into 4 bytes digest. So our proposed algorithm uses simple and fast functions to generate digital signature, it is a fundamental advantage which could help applications to be simple, fast and secure.

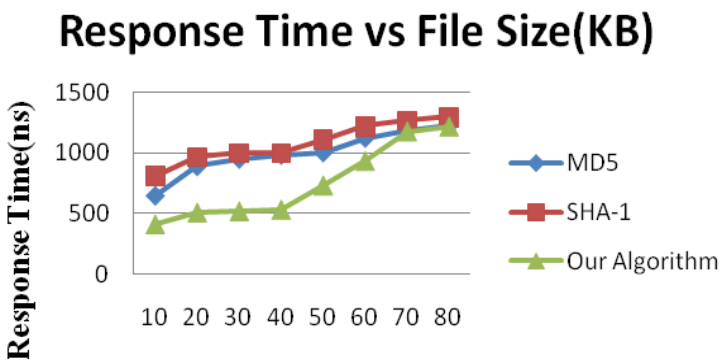

Fig-4

Our method computes the hash of data stored in a file. We performed the tests with a data size of 4 KB, 135 KB, and 1 MB to see how the size of data impacts performance. With increase in size of data, we see that the performance difference between the various algorithms has increased.

5.3 Theoretical tests against Attacks

We distinguish the following three levels of security that hash functions may satisfy to be useful in cryptographic applications.

One-Way: Hash function H is one-way if, for random key k and an n-bit string w, it is hard to find x so that $H_k(x) = w$.

Second-Pre-image Resistant: Hash function H is second-pre-image resistant if it is hard to find y ($\neq$ x) with a random key k and random string x so that $H_k(x) = H_k(y)$.

Collision Resistant: Hash function H is collision resistant if it is hard to find x and y with a random key k so that $H_k(x) = H_k(y)$ where $x \neq y$.

A generic attack is an attack that applies to all hash functions, no matter how good they are to exploit flaws of a particular design. The running time of generic attacks is measured in the number of calls to the hash function. We know the best strategy for inverting the hash function and for finding a second pre-image of the hash is the exhaustive search. The only strategy which is guaranteed to work for any hash function is to probe arbitrary chosen strings until a pre-mage of w of n bits is hit. For a random function H it would take on an average 2^{n-1} number of calls of H.

Our proposed hash technique creates a dynamic hash of 4N bytes or 32N bits where N is the number of blocks each of 100 bytes for a message M. We test our algorithm on a file of size 1KB. The following table gives a comparative study of optimal number of calls of the hash required to find pre-image collision.

Table-1

Scheme	File Size (input)	Hashed Size(bits) (output)	No of Calls
MD5	1 KB	128	2^{127}
SHA-1	1 KB	160	2^{159}
Our Algorithm	1 KB	326	2^{325}

Finding collisions that goes under the name of the "birthday paradox" is the chances that among 23 randomly chosen people there are two who share the same birthday are almost 51%. More careful analysis of the birthday paradox shows that in order to attain probability better than 50% of finding a collision in a hash function with n-bit output, it suffices to evaluate the function on approximately 1.2 X $2^{n/2}$ randomly chosen inputs. The running time of generic attacks on different properties of hash functions provide upper bounds on security of any hash function. We say that a hash function has ideal security if the best attacks known against it are generic. Cryptanalysts consider a primitive broken if its security

is shown to be less than the ideal security, even though it may still be sufficient for some applications. The ideal securities of hash functions on generic attacks are summarized in the following table.

Table-2

Property	Ideal Security
One Way	2^{n-1}
Second Pre image Resistance	2^{n-1}
Collision Resistance	$1.2 \times 2^{n/2}$

The following table shows our experiments on ideal security on collision resistance for a file size of 1 KB.

Table-3

Scheme	File Size (input)	Hashed Size(bits) (output)	No of inputs for Collision
MD5	1 KB	128	1.2×2^{62}
SHA-1	1 KB	160	$1.2X2^{80}$
Our Algorithm	1 KB	326	$1.2X2^{163}$

5.4 Algorithm against Attacks

By virtue of digital signature scheme, a bit of change in the original file will change the hashed file and hence the encrypted message. In this section we show a little change in the original file gets effected on hashed file. The following figure shows 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.

Table-4

Original File	Hashed File
Tiger is in the cage	a4f5j908
Tiger is on the cage	c5f0j9a5
Tiger is in the cag	z414i30m
Tiger is in tho cage	b8f1l50x
Tigar is in the cage	a6f5uv0k

VI. FUTURE SCOPE OF WORK

We need more experiments to evaluate and to analyze the effectiveness of the proposed algorithm under other attacks. We want to develop an environment to deploy the scheme for applications that send low size message like a Multi Agent System (MAS) in which agents send message in a few bytes.

VII. CONCLUSION

Digital signature relies on the protection afforded a private signature key by the signer, and the procedures implemented by a Certification Authority. Forgery of digital signatures, in the absence of compromise of the private signature key or hijacking of the signature mechanism, is virtually impossible. Due to the cryptographic nature of digital signatures, attempted forgeries are immediately obvious to any verifier, except in the case where a private signature key has been compromised, or control of the signing mechanism has been seized. In these cases, distinguishing between a valid and invalid digital signature may be impossible, even for a computer forensics specialist. Digital signatures have the potential to have the greatest impact on commerce since the invention of money. Digital signatures allow us to identify ourselves and make commitments in cyberspace in much the same way as we do in actual space. Nonetheless, digital signatures have important limitations, the most significant being their temporary nature.

REFERENCES

[1] Cryptography and Network Security—Principles and Practice by William Stallings, Pearson

[2] A Classical Introduction to Cryptography—Applications for Communications Security by Serge Vaudenay, Springer

[3] Data Communications and Networking by B A Forouzan, McGraw-Hill

[4] A Novel Secure Hash Algorithm for Public Key Digital Signature Schemes by T Lakshmanan et el , IAJIT, Vol-9,No-3, May 2012

[5] 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.

[6] 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.

[7] New Implementation of Hashing and Encoding in Digital Signature by E Noroozi et el, IPCSIT, Vol-29, 2012

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

[9] Hash functions: Theory, attacks, and applications by Ilya Mironov, Microsoft Research, November 2005

[10] 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.

