Signing Message Using Fast MD5 Hashing Digital Signature Algorithm

N Kishore (M.Tech)
Vignan's Institute OF Information Technology,
Visakhapatnam

Dr K Venkata Rao PhD
Vignan's Institute OF Information Technology,
Visakhapatnam

Abstract: Applications such as banking, stock trading, and the sale and purchase of merchandise are increasingly emphasizing electronic transactions to minimize operational costs and provide enhanced services. This has led to phenomenal increase in the amounts of electronic documents that are generated, processed, and stored in computers and transmitted over networks. This electronic information handled in these applications is valuable and sensitive and must be protected against tampering by malicious third parties (who are neither the senders nor the recipients of the information). Sometimes, there is a need to prevent the information or items related to it (such as date/time it was created, sent, and received) from being tampered with by the sender (originator) and/or the recipient. For e-documents, a mechanism of signature is necessary. Digital signatures, which are nothing but a string of ones and zeroes generated by using a digital signature algorithm, serve the purpose of validation and authentication of electronic documents. In this paper we use an algorithm called *Fast MD5* is used to provide integrity to e-documents.

Key Words: digital signatures, integrity, validation and authentication

I Introduction

The era of electronic mail" [2] may soon be upon us; we must ensure that two important properties of the current \paper mail" system are preserved: (a) messages are private, and (b) messages can be signed. We demonstrate in this paper how to build these capabilities into an electronic mail system. Validation refers to the process of certifying the contents of the document, while authentication refers to the process of certifying the sender of the document. In this paper, the terms document and message are used interchangeably.

A Traditional signature has the following salient characteristics: relative ease of establishing -
that the signature is authentic, the difficulty of forging a signature, the no transferability of the signature, the difficulty of altering the signature, and the non-repudiation of signature to ensure that the signer cannot later deny signing. A digital signature should have all the aforementioned
e-mail and credit card transactions over the Internet. Since a digital signature is just a sequence of zeroes and ones, it is desirable for it to have the following properties:

The signature must be a bit pattern that depends on the message being signed (thus, for the same originator, the digital signature is different for different documents); the signature must use some information that is unique to the sender to prevent both forgery and denial; it must be relatively easy to produce; it must be relatively easy to recognize and verify the authenticity of digital signature; it must be computationally infeasible to forge a digital signature either by constructing a new message for an existing digital signature or constructing a fraudulent digital signature for a given message; and it must be practical to recopies
of the digital signatures in storage for arbitrating possible disputes later. To verify that the received document is indeed from the claimed sender and that the contents have not been altered, several procedures, called authentication techniques, have been developed. However, message authentication techniques cannot be directly used as digital signatures due to inadequacies of authentication techniques. For example, although message authentication protects the two parties exchanging messages from a third party, it does not protect the
two parties against each other. In addition, elementary authentication schemes produce signatures that are as long as the message themselves.

In Basic terminology Digital signatures [3] are computed based on the documents (message/ information) that need to be signed and on some private information held only by the sender. In practice, instead of using the whole message, a hash function [4] is applied to the message to

obtain the message digest. A hash function, in this context, takes an arbitrary- sized message as input and produces a fixed-size message digest as output. Among the commonly used hash functions in practice are MD-5 (message digest 5) and SHA (secure hash algorithm). These algorithms are fairly sophisticated and ensure that it is highly improbable for two different messages to be mapped to the same hash value. There are two broad techniques used in digital signature computation

Symmetric key cryptosystem and public-key cryptosystem (cryptosystem broadly refers to an

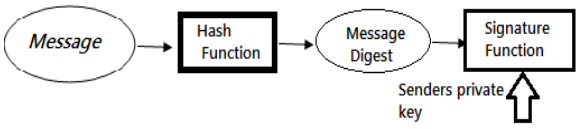

Fig 1: Creating digital signatures

encryption technique). In the symmetric key system, a secret key known only to the sender and the legitimate receiver is used. However, there must be a unique key between any two pairs of users. Thus, as the number of user pairs increases, it becomes extremely difficult to generate, distribute, and keep track of the secret keys. A public key cryptosystem, on the other hand, uses a pair of keys: a private key, known only to its owner, and a public key, known to everyone who wishes to communicate with the owner. For confidentiality of the message to be sent to the owner, it would be encrypted with the owner's public key, which now could only be decrypted by the owner, the person with the corresponding private key. For purposes of authentication, a message would be encrypted with the private key of the originator or sender, who we will refer to as A. This message could be decrypted by anyone using the public key of A. If this yields the proper message, then it is evident that the message was indeed encrypted by the private key of A, and thus only A could have sent it.

A simple generic scheme for creating and verifying a digital signature is shown in Figs. 1 and 2, respectively. A hash function is applied to the message that yields a fixed-size message digest. The signature function uses the message digest and the sender's private key to generate the digital signature. A very simple form of the digital signature is obtained by encrypting the message digest using the sender's private key. The message and the signature can now be sent to the recipient.

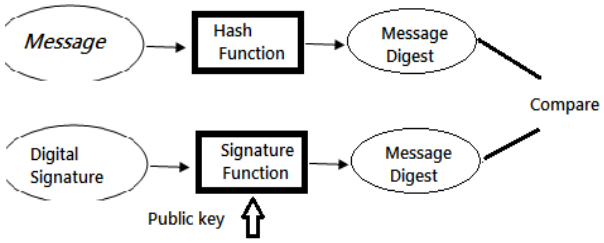

Fig 2: Verifying a Digital Signature

However, the signature ensures authenticity of the sender (something similar to a circular sent by a proper authority to be read by many people, with the signature attesting to the authenticity of the message). At the receiver, the inverse signature function is applied to the digital signature to recover the original message digest. The received message is subjected to the same hash function to which the original message was subjected. The resulting message digest is compared with the one recovered from the signature. If they match, then it ensures that the message has indeed been sent by the (claimed) sender and that it has not been altered.

A digital envelope [5] is the equivalent of a sealed envelope containing an unsigned letter. The outline of creating a digital envelope is shown in Fig. 3. The message is encrypted by the sender using a randomly generated symmetric key. The symmetric key itself is encrypted using the intended recipient's public key. The combination of the encrypted message and the encrypted symmetric key is the digital envelope. The process of opening the digital envelope and recovering the contents is shown in Fig. 4. First, the encrypted symmetric key is recovered by a decryption using the recipient's private key. Subsequently, the encrypted message is decrypted using the symmetric key.

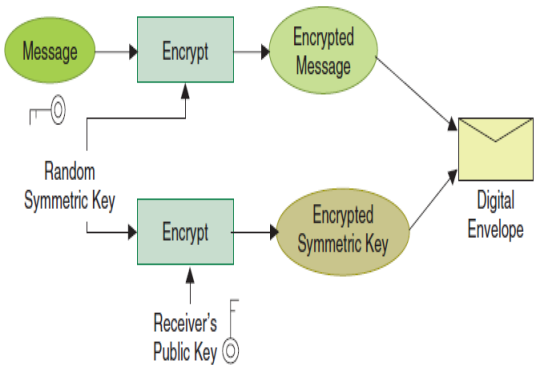

Fig 3: Creating a Digital envelope

And the next step involves creating the Digital envelop and making the envelop carrying the Digital Signatures that is generated from the first three diagrams. The message is encrypted by the sender using a randomly generated symmetric key. The symmetric key itself is encrypted using the intended recipient's public key.

If electronic mail systems are to replace the existing paper mail system for business transactions, \signing" an electronic message must be possible. The recipient of a signed message has proof that the message originated from the sender. This quality is stronger than mere authentication (where the recipient can verify that the message came from the sender); the recipient can convince a \judge" that the signer sent the message. To do so, he must convince the judge that he did not forge the signed message himself! In an authentication problem the recipient does not worry about this possibility, since he only wants to satisfy himself that the message came from the sender.

An electronic signature must be message-dependent, as well as signer-dependent. Otherwise the recipient could modify the message before showing the message-signature pair to a judge. Or he could attach the signature to any message whatsoever, since it is impossible to detect electronic \cutting and pasting."

II Proposed Digital Signature algorithm

Existing System:

* Java's built-in MD5 support is a bottleneck for your program's performance and you want something faster that takes less time.
* You are using a version of Java which doesn't have MD5 support, such as J2ME MIDP/CLDC.
* You want the extra convenience methods for hashing a file, hashing a string, converting the hash to a hex string, etc

Proposed System

* You can then convert the hash into the familiar hex format (e.g., "d41d8cd98f00b204e9800998ecf8427e"), if you wish. If you don't know how to do that, the Fast MD5 Implementation can do it for you.
* Much faster than any other Java implementation that I have tested and (surprisingly) even faster than the native, non-Java MD5 implementation on some systems.
* First of all, it is important to note that the term "fast" is used here in relative terms. The implementation of the MD5 message digest algorithm available on this page is written in Java and is fast compared with other implementations written in Java, both because it is heavily optimized by itself and because there is an optional native method that makes it even faster when the platform supports it. How it compares to a sensible implementation written in a language, such as C, that is compiled directly to machine code, is heavily dependent upon how good of a job the JIT compiler in your JVM does in compiling the code or whether you are able to use the optional native method.
* all the implementations took roughly the same amount of real time to execute, except when the JVM was run in interpreted mode (in which case it took much longer). This indicates that file I/O was likely the bottleneck. Therefore, the MD5 implementation that ships with the JDK will be adequate in a large number of cases. On the other hand, my fast implementation will be useful when the underlying I/O is very fast (e.g., if the data to be hashed is being created on-the-fly in emory), the CPU is inherently slow, CPU cycles are scarce (e.g., many other programs are running at the same time), the Java VM being used does not provide an adequately fast JIT, or in other cases where it is desirable to minimize CPU usage. The very surprising (for me) thing to note is that the Fast MD5 implementation outperforms the native "md5sum" binary even when the native methods aren't used. Oddly enough, the same did not hold true on Windows where the native binary was faster in all cases (perhaps the underlying I/O implementation in the Linux JVM is more efficient than the same in the Windows JVM.

The MD5 algorithm is quite possibly the most widely used digest algorithm out there. So of course, being the geek you are, you want to know how it works. Read on the following paper

First you must think of your message, not as a sequence of bytes, but as a sequence of bits (but only for a short while). The MD5 algorithm will accept messages that are any arbitrary number of bits long. However, so that the algorithm can process the data, it begins by padding the message to a length it can handle. This length just happens to be any number such that length mod 512 is equal to 448, or 64 bits short of being a multiple of 512. The message is padded by first appending a 1 bit to the message, and then enough 0 bits to make the message the proper length. The 1 bit is always added, so even if the message is already the proper length, it will be padded (a message can be padded with anywhere from 1 to 512 bits).

The next step is to calculate the length of the message (before padding). This number is then appended as the last 64 bits of the message, making the message length a multiple of 512. If the

message happens to be greater than 2^64 bits long, then the least significant 64 bits of the message are used (length mod 2^64)[8].

The MD5 algorithm uses 4 state variables, each of which is a 32 bit integer (an unsigned long on most systems). These variables are sliced and diced and are (eventually) the message digest. The variables are initialized as follows:

A = 0x67452301
B = 0xEFCDAB89
C = 0x98BADCFE
D = 0x10325476

Now on to the actual meat of the algorithm: the main part of the algorithm uses four functions to thoroughly goober the above state variables. Those functions are as follows:

F(X,Y,Z) = (X & Y) | (~(X) & Z)
G(X,Y,Z) = (X & Z) | (Y & ~(Z))
H(X,Y,Z) = X ^ Y ^ Z
I(X,Y,Z) = Y ^ (X | ~(Z))

Where &, |, ^, and ~ are the bit-wise AND, OR, XOR, and NOT operators (respectively) that all C programmers should be familiar with.

These functions, using the state variables and the message as input, are used to transform the state variables from their initial state into what will become the message digest. For each 512 bits of the message increasingly, digital signatures are being used in secure e-mail and credit card transactions over the Internet. The two most common secure e-mail systems using digital signatures are Pretty Good Privacy and Secure/Multipurpose Internet Mail Extension. Both of these systems support the RSA as well as the DSS-based signatures. The most widely used system for the credit card transactions over the Internet is Secure Electronic Transaction (SET)[9]. It consists of a set of security protocols and formats to enable prior existing credit card payment infrastructure to work on the Internet. The digital signature scheme used in SET is similar to the RSA scheme. And this type of digital signature with Fast MD5 algorithm can be used for safe and fast transactions. This Fast MD5 algorithm makes Hash Code stronger in less span of time it is the overcome of MD5 algorithm in feature of Fastness.

III Conclusion

Many traditional and newer businesses and applications have recently been carrying out enormous amounts of electronic transactions, which have led to a critical need for protecting the information from being maliciously altered, for ensuring the authenticity, and for supporting non-repudiation. Just as signatures facilitate validation and verification of the authenticity of paper documents, digital signatures with Fast MD5 Hash Algorithm serve the purpose of validation and authentication of electronic documents by integrity checking. This technology is rather new and emerging and is expected to experience growth and widespread use in the coming years.

References:

[1] Erfaneh Noorouzi ,Amir Reza Estakhrian Haghighi , A New Digital Signature Algorithm *IPCSIT vol.3 (2011) © (2011) IACSIT Press, Singapore*
[2] D. E. Denning, Cryptography and Data Security: *Addison-Wesley Publishing Co*, July 1998
[3] 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.
[4] 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.
[5] A. Buldas and M. Saarepera, "Electronic Signature System with Small Number of Private Keys," presented at 2nd Annual PKI Research Workshop, 2003.
[6] 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.
[7] A.Asokan, G.Tsudik, and M.Waidner, "Server supported digital signatures," presented at proceedings of ESORICS'96, Rome, Italy, 1996.
[8] S. J. Hwan and C.-C. Chen, "New multi-proxy multi signature schemes," Applied Mathematics and Computation, vol. 147, pp. 57-67, 2004.
[9] Y. Mizukami, M. Yoshimura, H. Miike, and I. Yoshimura,"An Off-line Signature Verification System Using an Extracted Displacement Function," 2004.