Volume 3, Issue 10, October 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A New Design and Implementation of Mixer Hashing Mechanism for Mobile Transactions

M.Gobi
Chikkanna Government Arts College, India

S.Selvi
PSG College of Arts and Science, India

Abstract-An elaborate set of data protection methods and techniques has been created to deal with the information security issues which are more while the information is conveyed as physical documents. Various mathematical algorithms and protocols are used to maintain security and integrity of transactions. Sun Java Wireless Toolkit is generally adopted to implement m-commerce applications. J2ME's Connected Limited Device Configuration (CLDC), Mobile Information Device Profile (MIDP) designed to run on cell phones and other small mobile devices are mainly involved for creating mobile applications .This paper describing a new hash algorithm called Mixer digest and has been written in response to a Hash competition (SHA-3) and MD5. The new proposed mixer digest algorithm is designed as secure hash algorithm and it can be used to generate a condensed representation of a message called Message Digest and it can be used to develop and implement a class in Sun Java Wireless Toolkit to ensure integrity of transactions. There is also possibility to generate a jar file that can be of the proposed classes at the time of deployment.

Keywords — J2EE, J2ME, MD5, SHA-3,Sun Java Wireless Toolkit.

I. INTRODUCTION

The increasing use of m-commerce applications have necessitated the need for maintaining integrity of transactions. J2EE (Java 2 Enterprise Edition) provides a class called MessageDigest which is used to provide integrity. Details on how this class can be used can be had from [1]. The Sun Java Wireless Toolkit which is based on J2ME's Connected Limited Device Configuration (CLDC) and Mobile Information Device Profile (MIDP), do not have a class that can maintain integrity of transactions. In this paper, The Mixer digest algorithm is designed and proposed to implement further in a J2ME environment. This paper proposed a class details on hashing and message digest. II and III describe how this class can be designed. The jar file creation for a class is presented in section IV. Section V and VI demonstrates the mixer digest algorithm functions and how this class could be used to ensure integrity in m-commerce transactions.

The proposed Mixer Digest algorithm is an iterative, one-way hash function that can process a message to produce a condensed representation called a Message Digest. This algorithm determines the message's integrity: any change to the message will, with a very high probability, result in a different Message Digest. This property is useful in the generation and verification of digital signatures and message authentication codes, and in the generation of random numbers (bits).
The algorithm can be described in two stages:

1) Preprocessing - Dividing a message into m-bit blocks, padding process takes place if number of bits of a message less than m-bits for the last block and setting up initialization values to be used in the hash computation.
2) Hash Computation - is performed on each m-bit input block in the Cipher Block Chaining (CBC) mode. The final hash value is generated by the hash computation and is used to determine the message digest.

II. HASHING and MESSAGE DIGEST

A hash function takes a message of any length as input and produces a fixed length string as output, sometimes termed a message digest. [2]
The characteristics of a good hash function are:

- Should avoid collisions.
- Should try to spread keys evenly in the array.
- Should be easy to compute.

The two most-commonly used hash functions are MD5 and SHA-1 [3].

III. BASICS of MD5

MD5 (**Message-Digest algorithm 5**), is an Internet standard (RFC1321) [4] and is one of the widely used cryptographic hash function with a 128-bit message digest. This has been employed in a wide variety of security applications.

The main MD5 algorithm operates on a 128-bit, divided into four 32-bit words. These are initialized to certain fixed constants. The main algorithm then operates on each 512-bit message block in turn, each block modifying the state. The processing of a message block consists of four similar stages, termed rounds; each round is composed of 16 similar operations based on a non-linear function F, modular addition, and left rotation. In this work, It is attempted to implement MD5 algorithm in Sun Java Wireless Toolkit. Since there is no in-built class for message-digest computation, an MD5 class is designed.

A. DESIGN OF MD5 CLASS IN MOBILES

The MD5Pack package which we designed in Sun Java Wireless Toolkit contain the core message digest class which includes Mixer Digest functions and it is called as MD5cls class. The MD5cls constructor accepts a string of any length and converts it into fixed length 128 bit message digest.

The main methods used in this class are initiate(),update(), final() and asHex(). The message digest result is the string type which contains the hexa-decimal value.

The following are the sequence of statements used to generate the message digest for the given string.

```
m.Init();              //Init method is used to perform initialization process.
m.Update(String);      //update method is used to concatenate and update the string which is passed.
m.asHex ();            // asHex method is used to return the message digest as hexadecimal value;
                          'm' is the MD5cls object.
```

This MD5cls class object is created and accessed from the MD5Demo class which extends the MIDlet class. This MD5Demo class is used to design the interface on mobile and is used to pass the string to main message digest class.

IV. JAR FILE CREATION

The Java™ Archive (JAR) file format enables one to bundle multiple files into a single archive file. Typically a JAR file contains various class files and auxiliary resources associated with the application. Details on what a Java Archive file is can be had from [7]. Sun Java Wireless Toolkit, provides necessary menus (project→ package → create package) for creation of .jar file and this .jar file can be transferred and executed in any mobile that supports Java.

V. MIXER DIGEST FUNCTIONS

Mixer Digest Algorithm takes an input of any size and returns an output of fixed size (224-bits). This algorithm has three functions, Shuffling(x), T-function(x),Lfsr(x), where x is a 32-bit word. The algorithm operates on 224 bits at a time in the CBC mode.

A.Shuffling Bits

The shuffling operation divides the message into 32-bit words and shuffles them. It interleaves the bits in two halves of each 32-bit word. The shuffling procedure used is an outer perfect shuffle, which means that the outer (end) bits remain in the outer positions.

If the 32 bit word (where each element denotes a single bit) is x1x2x3x4 x5x6x7x8 x9x10x11x12 x13x14x15x16 y1y2y3y4 y5y6y7y8 y9y10y11y12 y13y14y15y16, then the outer perfect shuffle changes it to x1y1x2y2 x3y3x4y4 x5y5x6y6 x7y7x8y8 x9y9x10y10 x11y11x12y12 x13y13x14y14 x15y15x16y16. Similarly, shuffling is performed on each 32-bit word of the entire padded message.

B.T-function

A T-function is a bijective mapping that updates every bit of the state, which can be described as $yi = xi+f(x0,...,xi-1)$. In simple words, it is an update function in which every If a single less significant bit is included in the update of every bit in the state, such a T-function is called Triangular. All the Boolean operations and most of the numeric operations in modern processors are T-functions, and all their compositions are also T-functions. The T-function helps to achieve Avalanche effect. The Avalanche effect tries to mathematically abstract the much desirable property of high non-linearity between the input and output bits, and specifies that the hashes of messages from a close neighborhood are dispersed over the entire space. As T-functions are bijective, there are no collisions, and hence no entropy loss, regardless of the Boolean functions and the selection of inputs. After shuffling, the T-function takes the shuffled output and performs the Avalanche effect.

C.Linear Feedback Shift Registers

A degree-n polynomial over GF(2n) is primitive if and only if it has 2n -1 periods. The period of a primitive polynomial of degree n is the maximum period (2n -1) realized by its corresponding Linear Feedback Shift Register (LFSR) implementation. Primitive polynomials over GF(2n) are useful in the design of Linear Feedback Shift Registers (LFSRs) to generate sequences for a maximum period. More taps are used in primitive polynomials in order to increase uncertainty on the outcomes of LFSRs. In our hash algorithm, regularly clocked LFSR is used. Each time the 32-bit input is executed for different number of rounds; thus, even if the input bits are identical, the output will be random. This is known as irregularly clocked LFSR over GF(2n).

PREPROCESSING

Preprocessing takes place before hash computation begins. It consists of three steps:

1) Dividing a message into m-bit blocks.
2) Padding process takes place if number of bits of a message is less than m-bits for the last block.
3) Setting up the initial hash values to zero.

VI. MIXER DIGEST ALGORITHM

A new secure hash algorithm Mixer Digest designed, which is different from the existing hash algorithms. The new algorithm is a one-way hash function, which is a combination of message preprocessing (bijective function) and Cipher Block Chaining (CBC) mode. The algorithm has been designed in two stages, in a sequential manner: padding and hash computation. Message is divided into 'm' blocks of a fixed length (the padding is done for the last block if the length of the last block is less than messaged digest length) and setting up the initial hash values are set to zero (H(i) = 0). Message preprocessing involves performing bijective operations such as Shuffling, T-function and Linear Feedback Shift Register (LFSR). These operations are used in the hash computation, which is performed on each input block of a fixed length in Cipher Block Chaining (CBC) mode, with six iterative rounds. The final value generated by the proposed hashing is used to determine the message digest / hash value.

Mixer Digest (224-bits)

Mixer Digest (224-bits) Preprocessing

To preprocess the Khichidi-1 (224-bits) algorithm:

1. Divide the padded message into N 224-bit message blocks, M(1), M(2) , ..., M(N),.
2. Pad the message, M, by following the process.
3. Set the initial hash values H(i), as specified.

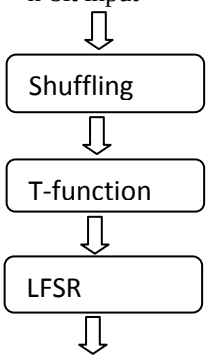
Mixer Digest(224-bits) Hash Computation

After preprocessing is completed, each message block, M(1), M(2) , ..., M(N), is processed in a sequential order, by using the following steps:

Digest

for j = 0 to 6
{
1. Perform Shuffling:
   $Xj^{(i)}$=Shuffling($Mj^{(i)+}$ $Hj^{(i)}$)
2. Pass it to the T-function
   $Yj^{(i)}$ = T-function($Xj^{(i)}$)
3. Pass this word to Linear Feedback Shift Register (LFSR)
   $Zj^{(i)}$ = LFSR($Yj^{(i)}$)
   $Hj+1^{(i)}$ = $Zj^{(i)}$

}
$H0^{(i)}$ = $H7^{(i)}$
for round =1
{
for i = 1 to N
{
Digest($M^{(i)}$)
}
}
for round 2 to 6
{
M^x = $Z0^{(N)}$ || $Z1^{(N)}$ || $Z2^{(N)}$ || $Z3^{(N)}$ || $Z4^{(N)}$ || $Z5^{(N)}$ || $Z6^{(N)}$
Digest(M^x)
M^y = $Z0^{(x)}$ || $Z1^{(x)}$ || $Z2^{(x)}$ || $Z3^{(x)}$ || $Z4^{(x)}$ || $Z5^{(x)}$ || $Z6^{(x)}$
Digest (M^y)
}
Hash = $Z0^{(y)}$ || $Z1^{(y)}$ || $Z2^{(y)}$ || $Z3^{(y)}$ || $Z4^{(y)}$ || $Z5^{(y)}$ || $Z6^{(y)}$
The resulting 224-bit message digests of the message, M, is
$Z0^{(y)}$ || $Z1^{(y)}$ || $Z2^{(y)}$ || $Z3^{(y)}$ || $Z4^{(y)}$ || $Z5^{(y)}$ || $Z6^{(y)}$



VII. CONCLUSION

A method for designing mixer message digest algorithm for mobiles has been presented with this paper. The hash algorithm has been tested in the Sun Java Wireless Toolkit. The implementation of this design is compatible for secured

transaction of all m-commerce applications.

REFERENCES

[1] http://java.sun.com/j2se/1.4.2/docs/api/java/security/MessageDigest.html.
[2] Ilya Mironov , *Hash functions: Theory, attacks, and applications*, Microsoft Research, Silicon Valley Campus, November 14, 2005
[3] Federal Information Processing Standards Publication 180-2, Secure Hash Standard, 2002 August 1, http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf
[4] http://www.scit.wlv.ac.uk/rfc/rfc13xx/RFC1321.html
[5] http://java.sun.com/javame/reference/apis.jsp
[6] http://java.sun.com/products/sjwtoolkit/download-2_5_1.html. http://java.sun.com/docs/books/tutorial/deployment/jar/

