International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 – 5161

Available at http://inpressco.com/category/ijcet

Research Article

Carapace for Intranet Security of Cryptography

Premchand Ambhore†*, B B Meshramr‡ and Archana Wankhadeψ

†Computer Science and Engineering, GCOE Amravati University, Amravati,(M.S.) India
‡Computer Engineering Department, VJTI Mumbai,(M.S.) India
ψInformation Technology Department, GCOE Amravati (M.S.) India

Accepted 15 Nov 2014, Available online 01 Dec 2014, **Vol.4, No.6 (Dec 2014)**

Abstract

Thus, in this paper, we had seen knowledge about number theory which is very essential in implementation of cryptographic algorithm, Public key and Private Key algorithm these are main part of this paper. We discussed some part of Elliptic Curve Cryptography and Security. Cryptographic algorithms and Applications. Hence, we conclude that by using above algorithms, security in 'text file' as well as 'PDF file the cryptographic algorithms are implemented using java. The input and output of each algorithm is shown in the screen shot. The algorithm was also used for developing "pdf-encryption" application. Thus, these algorithms are useful in providing security, authentication & avoiding denial of service. Cryptography with java application software which tries to show the demo of cryptographic algorithms AES, DES, RSA, MESSAGE DIGEST, SHA-1, DIGITAL-SIGNATURE GENRATION AND VERIFICATION Beside demo digital signature accepts the input as file generate signature and verifies from other user on Application part we have implemented PDF Encryption and Decryption:-In this we have to give input as PDF file only and output will be encrypted test file. Audio Encryption and decryption :- this application we have made keeping in mind of military use as someone want to encrypt the voice messages so it will output it as the Text file ,the advantage is that Intruder will not come to know that its voice message, but actual user who have this software will automatically know via the transmission line Text Encryption and Decryption:-Its simple Text encryption of .txt files SHADOW:-This application is really worth to have notice in this we are trying to recover the master key in case sender or recipient dies. Third part is NUMBER THEORY modules SQRT:-simple code written to explain how SQRT works in cryptography Poling: - This module of Number theory, which is called Pohlig-Hellman Exponentiation cipher, uses Fermat little Theorem. GCD:- NUMBER THEORY module which generated Greatest common divisor of two numbers. GAUSSIAN: - Gaussian Elimination module m. The user enters the modulus m, the square matrix of coefficients A, and vector of constants B. If a unique solution X exists modulo m, the applet will compute and display it.

Keyword*: Algorithm, Encryption, Decryption, Key*

1. Introduction

Cryptography is the science of secret writing. It's a branch of mathematics; part of cryptology has one other child, cryptanalysis, which is the science of breaking (analyzing) cryptography. The main security concerns of applications are addressed by cryptography. First, applications need assurance that users are who they say they are. Proving identity is called authentication. In the physical world, a driver's license is a kind of authentication. When we use a computer, we usually use a name and password to authenticate our self. Cryptography provides stronger methods of authentication, called signatures and certificates. Computer applications need to protect their data from unauthorized access. We don't want people snooping on our data (we want confidentiality), and we don't want someone changing data without our knowledge (we want to be assured of our data's integrity). Data stored on a disk, for example, may be vulnerable to being viewed or stolen.

Data transmitted across a network is subject to all sorts of nefarious attacks. Again, cryptography provides solutions; Cryptography can protect our data from thieves and impostors. Most web browsers now support SSL, a cryptographic protocol that encrypts information before it is transmitted over the Internet. SSL allows we to buy things, using our credit card number, without worrying too much that the number will be stolen. We can encrypt the files on our hard disk so that even if our enemies gain physical access to our computer, they won't be able to access its data. Email is notoriously easy to steal and easy to forge. Cryptography can make it hard to forge email and hard to read other people's messages. One of the really nice things about the Java Security API is that, like any good

*Corresponding author **Premchand Ambhore is a Research Scholar; Dr.B B Meshramr** is working as Professor and **Archana Wankhade** as Assistant Professor

software library, it hides a lot of complexity. The Security API exposes concepts, like Signature and Cipher, and quietly deals with the underlying details. We can use cryptography effectively in a Java application without knowing too much about what's going on underneath the hood. In the chapter of literature survey we have discussed various theorems related to number theory. **Number theory** mainly includes modular arithmetic, random number generation and prime number generation with algorithm and examples. As in Cryptography large numbers of primes are required, hence we discuss the testing methods to check whether given number is prime or not. Next part mainly consists of **Public key algorithms** which also known as **'Asymmetric Key Encryption'** uses public and private key pair to encrypt and decrypt the data. Asymmetric algorithms are usually efficient encrypting small amount of data only. **Private key encryption**, which is also known as **Symmetric Key Encryption**, uses single key to encrypt and decrypt the data. These algorithms are relatively fast and can be used to encrypt and decrypt large amount of information.

2. Statement of Problem

The main aim of this research is to study the various cryptographic algorithms and implementing cryptographic applications based on those algorithms. The Prerequisite For studying these algorithms is Number Theory which includes the study of Chinese-Remainder Theorem, Rabin-Miller Test, Euclidean Theorem, Fermat's Little Theorem, We have implemented following algorithms Public key cryptography algorithm includes: RSA, DSA, DIFFIE HELLMAN, HASH FUNCTION, MESSAGE DIGEST Private Key cryptography algorithm includes: DES, AES Using these algorithms we can make certain applications like, Texts file Encryption Application, PDF Encryption Application, Audio Encryption Application Modular Arithmetic: The fundamental arithmetic operations performed by most computers are actually modulo arithmetic, where the modulus is 2b (b being the number of bits of the values being operated on) Modular arithmetic is remainder. So for instance, 7 mod 5 = 2 because 5 goes into 7 with a remainder of 2. Congruent modulo is an interesting concept and important to cryptography a = b mod (n) says that a and b are congruent modulo n if a mod (n)=b mod (n). Chinese Remainder Theorem (CRT) example. CRT If the integers n1; n2; : : : ; nk are pair wise relatively Prime, then the system of simultaneous congruence's

x _ a1 (mod n1)
x _ a2 (mod n2) . . .
x _ ak (mod nk)
has a unique solution modulo n = n1 n2 _ _ _ nk.
For example,
x= 3 mod 4
x= 0 mod 5
x= 0 mod 7
x= 8 mod 9

Where a1.a2, a3, a4=3,0,0,8
n1, n2, n3,n4=4,5,7,9
The moduli are relatively prime, the proof of CRT shows us how to get our solutions very quickly by computing M=4*5*7*9=1260, M1=1260/4=315, M2=1260/5=252. M3=1260/7=180, M4=1260/9=140.
We then compute inverses yi of each Mi modulo mi.
M1'= 3 (an inverse of 315 modulo 4)
M2'= 3 (an inverse of 252 modulo 5)
M3'= 3 (an inverse of 180 modulo 7)
M4'= 2 (an inverse of 140 modulo 9)
To get our solution, we now simplify form the sum
X=1M1M1'+a2M2M2'+a3M3M3'+a4M4M4'
= 3.315.3 + 0.252.3 + 0.180.7 + 8.140.2
= 5075
= 35(mod 1260).

Public Key Cryptography: In public-key encryption systems, each entity A has a public key e and a corresponding private key d. In secure systems, the task of computing d given e is computationally infeasible. The public key defines an encryption transformation Ee, while the private key defines the associated decryption trans-formation .The public key need not be kept secret, and, in fact, may be widely available - only it authenticity is required to guarantee that A is indeed the only party who knows the corresponding private key. A primary advantage of such systems is that providing authentic public keys is generally easier than distributing secret keys securely, as required in symmetric-key systems.

Rivest Shamir Adelamn (RSA): The RSA cryptosystem, named after its inventors **R. Rivest, A. Shamir, and L. Adleman**, is the most widely used public-key cryptosystem. It may be used to provide both secrecy and digital signatures and its security.
The RSA Algorithm may be divided, then, into three steps:

(1) Key generation: in which the factors of the modulus (n) the prime numbers (p) and (q) are chosen and multiplied together to form (n), an encryption exponent (e) is chosen, and the decryption exponent (d) is calculated using (e), (p), and (q).
(2) Encryption: in which the message (M) is raised to the power (e), and then reduced modulo (n).
(3) Decryption: in which the cipher text (C) is raised to the power (d), and then reduced modulo (n).

When the RSA Algorithm is used in a public key system, the modulus (n) and one of the exponents (arbitrarily, we can assume (e)) are published. The other exponent (d) is kept secret, as are (p) and (q), the factors of (n). The (p) and (q), the factors of (n) are not needed for encryption or decryption; they are only used in the key generation step (creating the modulus (n) and the second's exponent). In addition, while it is important for key generation purposes that the modulus (n) be the product of two prime numbers, the exponentiation and modular arithmetic operation would work just as

well with prime numbers (which are by definition evenly divisible only by themselves and the number 1).

RSA Algorithm: In the RSA public key cryptosystem, a participant creates his public and private keys with the following procedure. Select at random two large prime numbers p and q. Compute n by the equation, n = p * q. Compute z by the equation, z=(p-1)* (q-1). Select a small odd integer d that is relatively prime to z. Compute e by the equation, e ´ d= 1 mod z Publish the pair (e, n) as the RSA public key. Keep secret the pair (d, n) as the RSA private key.Then encryption and decryption are performed by the following rules: If the message is M, encrypted message C = M^e mod n, and the decrypted message (original message) is found by M = C^d mod n.

RSA Example

An Example of the RSA Algorithm

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
Plaintext digit:
Let E=5 and n=7*17=119
D: 4 Cipher:-4^5 mod 119 =72 =20-> T
I: 9 Cipher:-9^5 mod 119 =25 -> Y
G: 7 Cipher:-7^5 mod 119 =28 =2->B
I: 9 Cipher:-9^5 mod 119 =25-> Y
T: 20 Cipher:-20^5 mod 119 =90 =12->L
Plaintext: - DIGIT \
Cipher text: - TYBYL

Digital Signature algorithm (DSA): A cryptographic primitive which is fundamental in authentication, authorization, and non-repudiation is the digital signature algorithm. The purpose of a digital signature is to provide a means for an entity to bind its identity to a piece of information. The process of signing entails transforming the message and some secret information held by the entity into a tag called a signature.

DSA Algorithm

Step 1: Select a prime number q.
Select a prime number p such that q divides p-1.
Select a number h from 1 to p-1.
Calculate g such that g=h^ (p-1)/q mod p.
Step 2: The user selects a private key x such that x is between 1 to q-1.
Calculate public key y such that y=g^x mod p
Select a number k randomly such that 0<k<q
Step 3: Signature generation
R= (g^k mod p) mod q
S= k-1*H (M) + x*r
Where M=Message & H (M) = hash of the message, Therefore Signature =(r, s)
Step 4: Signature verification
Calculate w = (s')-1 mod q
U1= [H (M')*w] mod q……M'=hrkdv (received message)
U2 = (r')*w mod q
V = [g^u1*y^u2 mod p] mod q
Here r', s', H (M') are the received parameters of the digital signature.
DSA Parameters
The DSA makes use of the following paramet p = a prime modulus, where 2L-1 < p < 2L for 512 = L = 1024 and L a multiple of 64 q = a prime divisor of p - 1, where 2^159 < q < 2^160
g = h(p-1)/q mod p, where h is any integer with 1 < h < p - 1 such that h(p-1)/q mod p > 1 (g has order q mod p)
x = a randomly or pseudo randomly generated integer with 0 < x < q
y = g^x mod p 6. k = a randomly or pseudo domly generated integer with 0 < k < q The i teger's p, q, and g can be public and can be common to a group of users. A user's private and public keys are x and y, respectively.
They are normally fixed for a period of time. Parameters x and k are used for signature generation only, and must be kept secret. Parameter k must be regenerated for each signature.
DSA Signature Generation
The signature of a message M is the pair of numbers r and s computed according to the equations below:
r = (g^k mod p) mod q and
s = (k^-1(SHA-1(M) + xr)) mod q.

In the above, k-1 is the multiplicative inverse of k, mod q; i.e., (k^-1 k) mod q = 1 and 0 < k^-1 < q. The value of SHA-1(M) is a 160-bit string output by the Secure Hash Algorithm For use in computing s, this string must be converted to an integer. As an option, one may wish to check if r = 0 or s = 0. If either r = 0 or s = 0, a new value of k should be generated and the signature should be recalculated (it is extremely unlikely that r = 0 or s = 0 if signatures are generated properly). The signature is transmitted along with the message to the verifier.

DSA Signature Verification

Prior to verifying the signature in a signed message, p, q and g plus the sender's public key and identity are made available to the verifier in an authenticated manner. Let M', r', and s' be the received versions of M, r, and s, respectively, and let y be the public key of the signatory. To verify the signature, the verifier first checks to see that 0 < r' < q and 0 < s' < q; if either condition is violated the signature shall be rejected. If these two conditions are satisfied, the verifier computes

w = (s')-1 mod q
u1 = ((SHA-1(M'))w) mod q
u2 = ((r')w) mod q
v = (((g)u1 (y)u2) mod p) mod q.
If v = r', then the signature is verified and the verifier can have high confidence that the received message

was sent by the party holding the secret key x corresponding to y. For a proof that v = r' when M' = M, r' = r, and s' = s

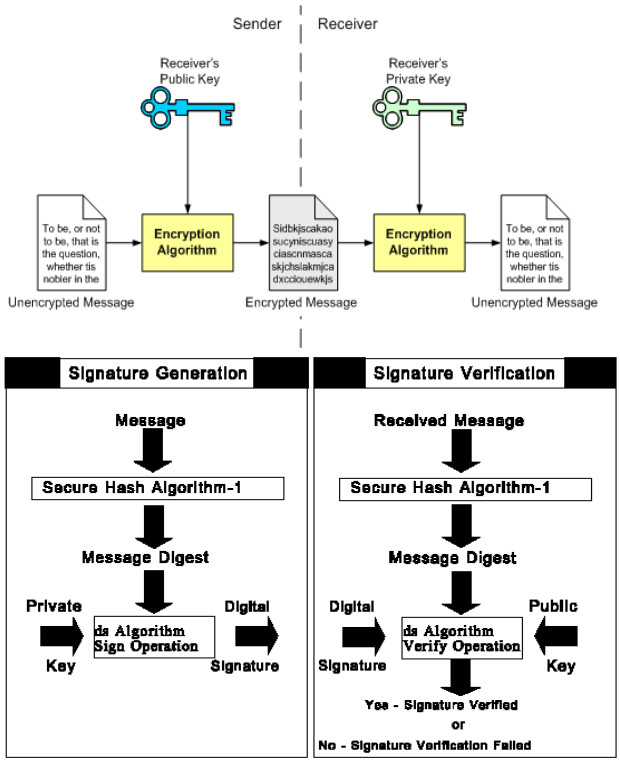


Fig. 1 Digital Signature & Verification

If v does not equal r', then the message may have been modified, the message may have been incorrectly signed by the signatory, or the message may have been signed by an impostor. The message should be considered invalid.

Hash functions: One of the fundamental primitives in modern cryptography is the cryptographic hash function, often informally called a one-way hash function. A simplified definition for the present. The most common cryptographic uses of hash functions are with digital signatures and for data integrity. With digital signatures, a long message is usually hashed (using a publicly available hash function) and only the hash-value is signed. The party receiving the message then hashes the received message, and verifies that the received signature is correct for this hash-value. This saves both time and space compared to signing the message directly, which would typically involve splitting the message into appropriate-sized blocks and signing each block individually.

Hash Function Algorithm

Take input message with maximum length less than 2^{64} bit .Divide it with 512 bits as block length f message is less than 512 then pad with 0'sAppend a block of 64 bit to the message // It contains the length of original message

Initialize MD buffer. Buffer can represented as five 32 bit registers. (A, B, C, D, E)Process message in 512 bit blocks

After all 512 bit blocks have been processed, the output from the last stage is 160 bit message digest.

Hash Function Example

The computation is described using two buffers, each consisting of five 32-bit words, and a sequence of eighty 32-bit words. The words of the first 5-word buffer are labeled A,B,C,D,E. The words of the second 5-word buffer are labeled H0, H1, H2, H3, H4. The words of the 80-word sequence are labeled W(0), W(1),..., W(79). A single word buffer TEMP is also employed. To generate the message digest, the 16-word blocks M(1), M(2),..., M(n) are processed in order. The processing of each M(i) involves 80 steps.Before processing any blocks, the H's are initialized as follows: in hex,

H0 = 67452301, H1 = EFCDAB89,H2 = 98BADCFE,H3 = 10325476, H4 = C3D2E1F0.

Now M(1), M(2), ... , M(n) are processed. To process M(i), we proceed as follows:

Divide M(i) into 16 words W(0), W(1), ... , W(15), where W(0) is the left-most word.

For t = 16 to 79 let, W(t) = S^1(W(t-3) XOR W(t-8) XOR W(t-14) XOR W(t-16)).

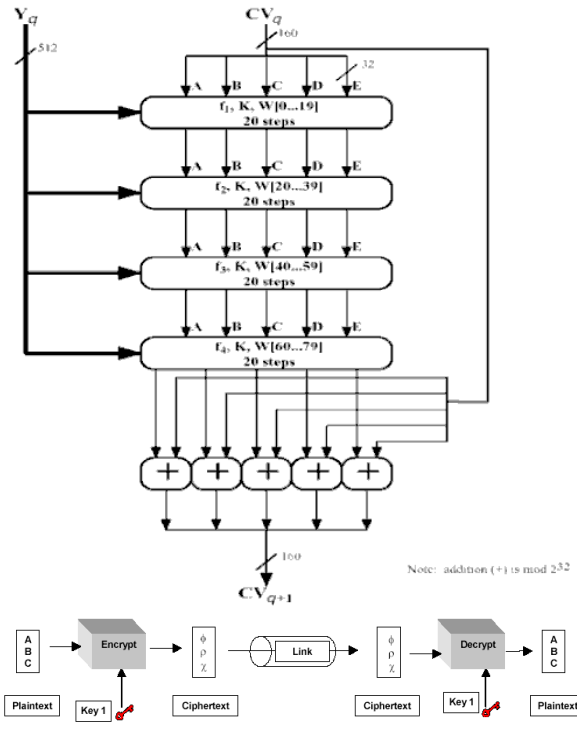
Let A = H0, B = H1, C = H2, D = H3, E = H4.

For t = 0 to 79 do TEMP = S^5(A) + f(t;B,C,D) + E + W(t) + K(t);

E = D; D = C; C = S^30(B); B = A; A = TEMP;

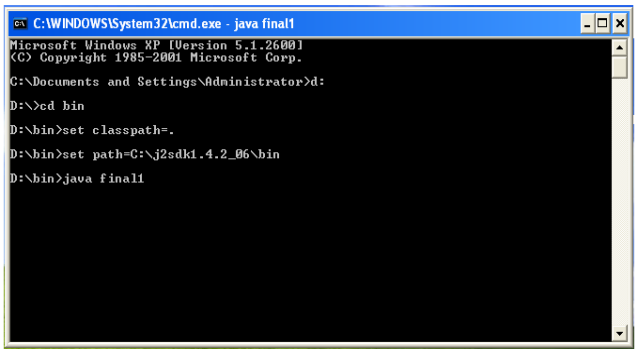
Let H0 = H0 + A, H1 = H1 + B, H2 = H2 + C, H3 = H3 + D, H4 = H4+ E.

After processing M(n), the message digest is the 160-bit string represented by the 5 words H0 H1 H2 H3 H4.

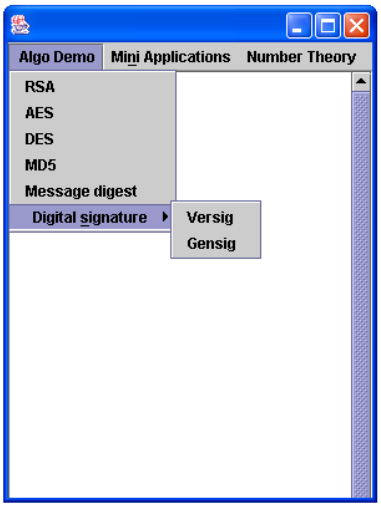


Input and Output Screens

How to run the program...

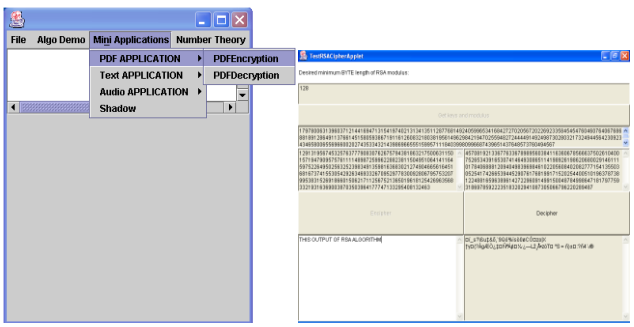


Screen 1:-This Screen shows list of algorithm we have implemented and used in our research applications in JAVA Version 2.

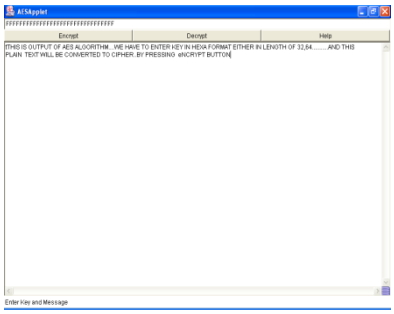


Screen 2:-This Screen is Just overview of the menus, how the research is designed Mini applications menu shows the different application option created using cryptography algorithm in java

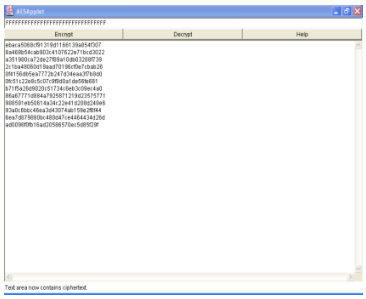
Screen 3:- This Applet is Created Using RSA Algorithm, You will see that if you encipher the same plaintext multiple times, you will receive a different cipher text,



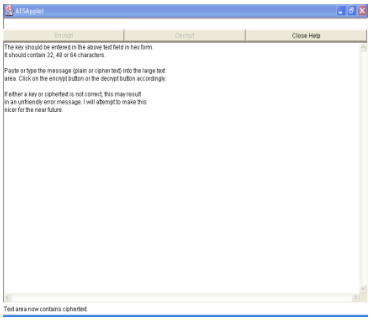
Screen 4:-This is applet is created using AES algorithm, in the upper text column Compulsorily write in multiple of 16,32 and in hex format as algorithm required. It is the sample for input of AES algo.



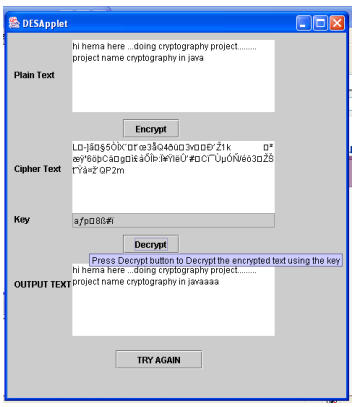
Screen 5:-This is output of AES Applet by encrypting the input text



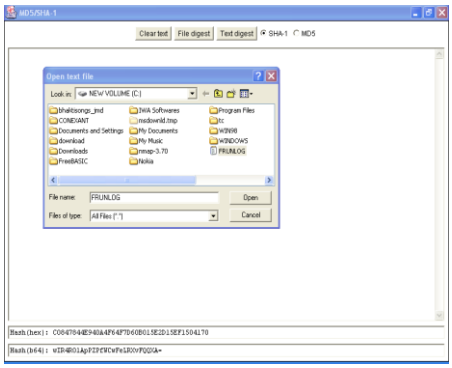
Screen 6:-This HELP screen for AES applet. The Text will guide to novice
User how to use the applet without any knowledge of cryptography



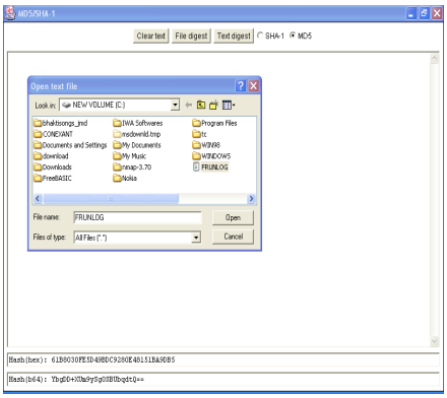
Screen 7:-This output screen generated by using DES applet. Key is automatically Generated using DES algorithm.



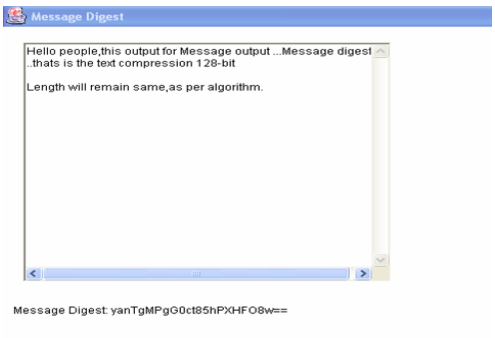
Screen 8:-This Screen is used to generate HASH function of text file, we Have two options available with this applet, SHA-1 & MD5



Screen 9:-This screen is generating HASH function of text file FRUNLOG.TXT output with MD5 algorithm. HASH in two format b64 and HEX



Screen 10:-This screen is generating Message Digest with byte length 128 of only text i/p it can be used for compression



Conclusion and Future Work

The JCA framework gives you a full range of cryptographic services that are flexible and easy to use. Each service can be used alone or in combination with one or more other services to create the level of cryptographic security you need. You can request a cryptographic service by specifying the algorithm you want and let the framework find the provider, or specify the algorithm and provider you want. It is easy to add new provider packages so you readily have the service and algorithm implementations that work best for your application requirements. Due to time constraint and due deep knowledge required by the cryptography subject itself we had restricted our self to this much application software One can continue with this theme and can enhance some applications like, SMS encryption and decryption ,Encryption on network settings, Office Security application n using these algorithms, Can further build Groups, RINGS,FIELDS in number theory.

References

David bishop (2003), Introduction to Cryptography with JAVA, Narosa Publications
Scott Oaks (2001), Java Security, Oreilly publication.
Jonathan Knudsen(1998), Java Cryptography, Oreilly publication.
Bruce Schneier (1996), Applied Cryptography: Protocols, Algorithms, and Source Code in C, Second, John Wiley
David Hook (1996), Beginning Cryptography in Java, PaperBack
William Stallings, Cryptography and Network Security, PHI Publications
Herbert Schildt (2004), Java(tm)2: A Beginner's Guide, Paperback
Steven Holzner, Java 2 Black Book by Steven Holzner
http://www.nsa.gov/museum/index.html
http://www.mirror.org/crypt/index.html
http://dir.yahoo.com/Computers_and_Internet/secrity_and_encryption/cryptography/
http://www.ciphersbyritter.com/LEARNING.HTM
http://www.itl.nist.gov/fipspubs/fip46-2.htm
http://theory.lcs.mit.edu/~rivest/crypto-security.html
http://www.vcnet.com/~rossde/pgp_encrypt.html
http://java.sun.com/products/jce
http://java.sun.com/j2se/1.4.2/docs/guide/security/index.html
http://javaboutique.internet.com
http://sourceforge.net
http://www.planet-source-code.com