New Secured Data Sharing Technique Using Elliptic Curve Cryptography and F5 Steganography

Sayantan Majumdar, Abhisek Maiti, Saptarshi Das, Achintya Das

Abstract- In the present study the authors have introduced a new technique for sharing secret data inside a JPEG image. An Android phone may be used to decode the data from the shared image. The secret message is first encrypted using AES [1] where ECDH [2] is used as the key agreement protocol. The original data is also digitally signed using ECDSA[3] where SHA-256[4] is used as the hash algorithm. The generated cipher, secret key, signature and public key files are compressed into a zip file. This zip file is further embedded in a JPEG image using F5 steganography[6]. The entire method has also been implemented for the Android mobile environment. This method may be used to transfer confidential message through Android mobile phone.

Index Terms- android, compression, cryptography, symmetric key, encryption, JPEG image, steganography

1 INTRODUCTION

In the presented paper the authors have used AES 192-bit encryption technique. Advanced Encryption Standard is based on a design principle known as a substitution-permutation network which is a combination of both substitution and permutation, and is fast in both software and hardware. So it is suitable for small scale devices like smartphone systems where computing resources are limited. Here the initialization vector (IV) is of 16-bytes and randomly generated. The IV is then padded to the secret key.

The ECDH key agreement protocol allows two parties, each having an elliptic curve public–private key pair, to establish a shared secret over an insecure channel. This shared secret may be directly used as a key, or to derive another key which can then be used to encrypt subsequent communications using a symmetric key cipher. The ecliptic curve used in this case is “brainpoolp256r1” [14].

For the digital signature generation, the authors have used 256-bit ECDSA encryption with prime256v1 [15] curve and SHA-256, which is fully compliant with NIST 800-57 security standard and FIPS 186-3 security standard.

Here the generated cipher, secret key, signature and public key files are all Base64 [10] encoded.

Efficient and secured steganography technique, F5 method is used to embed the encrypted message into container image file. Visual attacks on steganographic systems are based on essential information in the carrier medium that steganographic algorithms overwrite. Adaptive techniques (that bring the embedding rate in line with the carrier content) prevent visual attacks, however, they also reduce the proportion of steganographic information in a carrier medium. Lossy compressed image (JPEG) is originally adaptive and immune against visual attacks. Therefore, an altered image with slight variations in its colours will be indistinguishable from the original by a human being, just by looking at it.

2 ALGORITHMS USED

- *Sayantan Majumdar is currently pursuing masters degree program in Computer Science at St. Xavier's College, Kolkata, India, E-mail: monti.majumdar@gmail.com*
- *Abhisek Maiti is currently pursuing masters degree program in Computer Science at St. Xavier's College, Kolkata, India, E-mail: mail2abhisek.maiti@gmail.com*
- *Saptarshi Das is currently pursuing BTech degree program in Computer Science Engineering at Saroj Mohan Institute of Technology, Kolkata, India, E-mail: arkaforyou.1611@gmail.com*
- *Dr. Achintya Das is a PhD. holder in Electronics and Communication Engineering and currently heads the department of Electronics and Communication Engineering, Kalyani Govt. Engineering College, India, E-mail: achintya.das123@gmail.com*

In the present study the authors have introduced the following modules:

2.1 Encoding Algorithm:

1) Encryption of secret message using AES and ECDH.
2) Generation of digital signature using ECDSA and SHA-256.
3) Compressing the cipher, secret key, signature and public key files into a zip file
4) Embedding the zip file into a JPEG image using F5 steganography.

2.2 Decoding Algorithm:

1) Extraction of the cipher, secret key, signature and public key files from the embedded image
2) Deciphering the cipher.
3) Verification of the decrypted file.

3 CONCEPTUAL MODELS

3.1 Encoding:

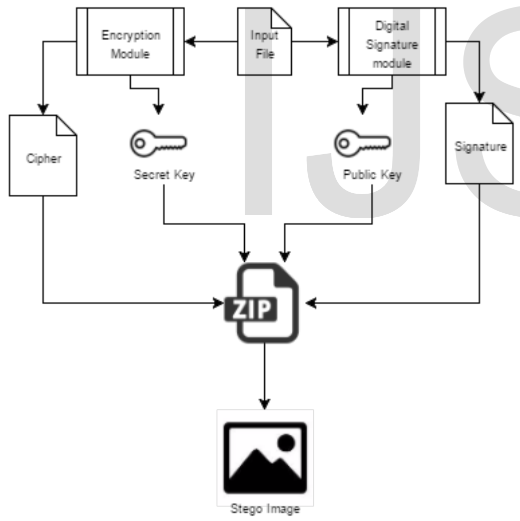


Fig. 1. Encoding Model

3.2 Decoding:

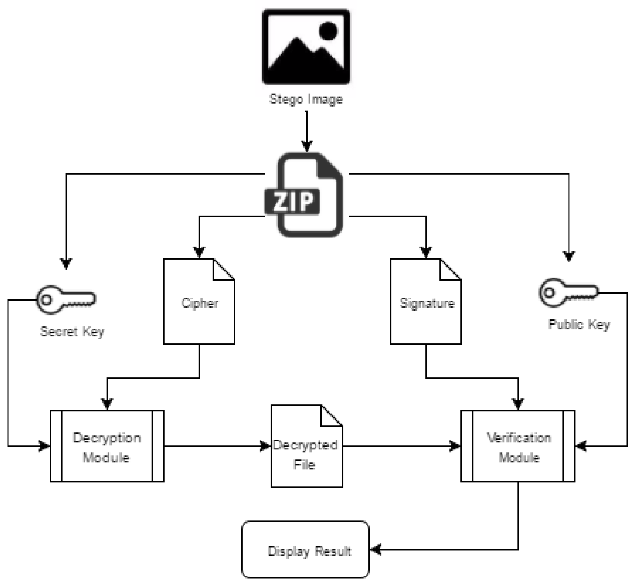


Fig. 2. Decoding Model

Above diagrams show the conceptual model of the project. Both of these models have been made to work in PC as well as Android mobile environment.

4 RESULTS & DISCUSSION

4.1 Input text file (441 Bytes)

The major cause of the software crisis is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem- Edsger Dijkstra, The Humble Programmer (EWD340), Communications of the ACM

4.1.1 Encryption:

1) Cipher (612 Bytes):

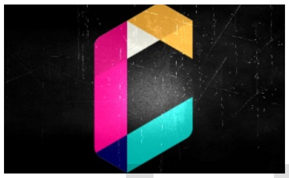
```
FfkdpnghWWWQslfRx5FQsvvd5btwT+QPx0tAGd39l
kcKEWWh0Id9HER+2WeILLOUvqjyf+UIPN4ylSzdSol
/MCzGiuAn+SYq2PSdbzeFNyWnB9++FN2m/AMHO
0GYncc4Q0gdMZow8uYNyBXgML7kmkot2r0qguaqK
2BWrpnai5CAaSFJWVwnQt0W0lPs5bRNawaNyO6M9
MdR7GpRe2l/SMnPKjlYutL3G+59JxSY0mBTaVejTb3jr
b7KwvPtoPRq2uE23rEceauLz48N28/uwkJ4OTC5sKoo
QH+xIz19WTq1pQgA6HfJmWkgVRDl8tWN9o+LPU2
/wXP/WkkybzPk0LMg+Hl55A2d7hXt/KIuhk0+La1P
pcBy6EpfBWBoEKNV5BAOwIybK1TPqSCeE8Vi+jct7
WxTYQaG9aSJWVXU/APxI+0+0KCjZnpAgnacKfaoM
50hAfcesidLjTULOfWNJnWe83i5Jid/5n3mdpXD/53M
SbZUZH8Bw1Svwbhhn96SAXy5e7xvSnZPAvjXZLOk3
byPxk/k+6TTLpl5Wg6jQBXk4nSyerLn3wVHKUHmo
NmezQ2lojsBlGXl6j2wEhf9WE86oAsav3IBCA==
```

2) Secret Key (64 Bytes):
ivN7AskIKCPIkjPIrYNDZURh0wV5CVQIxjmcin0za/TqVHWp3qlc16qdq4Bgl17n

3) Digital Signature (96 Bytes):
MEQCIB2jNcBX5srDwGa8YE3kDQOHyJX0cZ9bgaQfs+VjFXFcAiB7Gj/EFsMfoUIqp8DSD8bjyMKcx2R63neF+zbTPCNanQ==

4) Public Key (124 Bytes):
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEHFjqyYYMkLKZWhQkPOtILHgYy01a28IZ0EOnQpfqoqkzMA0PKuV0i1YXYNMDg0ZJxg3QKdkEFGZ3T3s3rpDcLA==

5) Original Image (79,879 Bytes):



6) Modified Image (15,286 Bytes):

4.1.2 Decryption:

1) Decrypted message (441 Bytes):

The major cause of the software crisis is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem- Edsger Dijkstra, The Humble Programmer (EWD340), Communications of the ACM

4.2 Input text file (120 Bytes):

"Program testing can be used to show the presence of bugs, but never to show their absence!"
— Edsger W. Dijkstra

4.2.1 Encryption:

1) Cipher (184 Bytes):
FOuAUHuIaBt66B5brt56qdIcFTOYvH+IKcA8x28DuJFasUCwLZUpuiRT/P3GXb1P2paydjDF5OSDcsF+qy+4BeUCl2dE8XL721rOHfd5oIwTs7BNbngvHEHWIKHPLHVqsqFbl3e0sUnEJA1s2cEEQt8UWTb9zX5lEBseuwq2QWcZnY7Y1i/y5g==

2) Secret Key (64 Bytes):
NQpqfB24iFXu3I1ikDXhA1ZSjxSdaG4GcCEz2Z7SzFyw6rMfeoy+cCRzKjKInh8/

3) Digital Signature (96 Bytes):
MEYCIQDA5ays10p6uKFqxkovkrChXK7Jgy1EvAaVZwKlZNVyJQIhAKUJXkP2HZmiVkw7k208cePKHSHWP1g5jAN8+TF4fokZ

4) Public Key (124 Bytes):
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE0nTFehrrHCd/BT62s1NuAwZclK2fEDgrukqHEZ79odquFFodFEAsPUF+5oXn3TExW+zq5bG+jNjmCnOvlO+2Ow==

5) Original Image (969,567 Bytes):

6) Modified Image (1,114,043 Bytes):

4.2.2 Decryption:

1) Decrypted text (120 Bytes):

“Program testing can be used to show the presence of bugs, but never to show their absence!”
— Edsger W. Dijkstra

4.3 Input PDF file (16316 Bytes):

Abhisek Maiti
Department of Computer Science,
St. Xavier's College, Kolkata
Email: xyz@gmail.com
Phone: +91 XXXXXXXXXX

Converted Text File (119 Bytes):
Abhisek Maiti

Department of Computer Science,
St. Xavier's College, Kolkata
Email: xyz@gmail.com
Phone: +91 XXXXXXXXXX

4.3.1 Encryption:

1) Cipher (180 Bytes):
58LIAeFdu39zZPyFj6t6L+sOouh+IEEVUf8g0cYzbjgp9f
rJdGFs1bzN7UmuVEc2Ka2zlE+wbbdAjU3+UBSe7sUq
+VSoRiN5t3QpD1Z7ZmGbQheU14EsJPTRuPRhCjMQ
CF3dYsLwRjm+RMCN7HxhcekoT7z6+PJLuf+UFZbxL
iiqmWN+qLSZ

2) Secret Key (64 Bytes):
gBtY4EsIm4Yr1YGbecIuezRoQv4C6AehsO+lPPK9XQ9
6ojoOnA/DzqND/NwzEm0O

3) Digital Signature (96 Bytes):
MEYCIQDC9FW5BZyf5Bl+4df00DLcZSdp5zj3G2T39O
J5JEMasQIhALANw+LpZIML/yw63IsL24/7KIAXLI7
dJFscbk7MDSTa

4) Public Key (124 Bytes):
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE59Ys
YdDqVtVknoYZiYhUszMDn5dJ12UDbWXH7gGXGyu
MMzrwWuPBxKyIO1k0TIMsLLSgC61Zal4WX29xsDtS
pQ==

5) Original Image (475,304 Bytes):

6) Modified Image (407,244 Bytes):

4.3.2 Decryption:

1) Decrypted text (119 Bytes):
Abhisek Maiti
Department of Computer Science,
St. Xavier's College, Kolkata
Email: xyz@gmail.com
Phone: +91 XXXXXXXXXX

4.4 Input text file (137 Bytes):

Trees sprout up just about everywhere in computer science...
Donald Knuth, Vol. IV - A, Combinatorial Algorithms, Section 4.2.1.6 (2011)

4.4.1 Encryption:

1) Cipher (204 Bytes):
6Hz38aJuMPSwErvQzYE4AUkoJVfOF/T3HpHzw48jJ
FbanBwRRghhDimdKFZnFQdYSRgLY2djEadLz1RGQ
7i9D0VQU48cAT3mHSJkgaZ+czXm85MQmFdkTyNH
tH/uEAND5UGSGSXECpTC/0wvh13xqNyEQ/U6l1+
74a3hMd7uoOjRiWl2tOo1zZhxAe/XwnYerEktCeU3ip
Uk

2) Secret Key (64 Bytes):
IUCsmLnlSqaD3eTBguArAR4tzZQVhHoi7xpRTZ1sqZ
dHUSTmCvjmfUBg8q1E8GS7

3) Digital Signature (96 Bytes):
MEUCIQCN2dTYXoyrlPbMvbF+BJm9RaV5yziJTPqUI
oPnCHqQJAIgaWrpEII2/wCwytY7WaqfPUpD2lGKy6
vPFwxDC8gOld4=

4) Public Key (124 Bytes):
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEOVE
G2SdBrcpe+gW4cKHpaaFfR/wYkZGv3sbK4w6IAy9z
/fRx5Kd+kC28SXGyqWA2G2ANJ1lzDt25d+Grok7F0
A==

5) Original Image (77,281 Bytes):

6) Modified Image ((51,500 Bytes):

4.4.2 Decryption:

1) Decrypted text (137 Bytes):
Trees sprout up just about everywhere in computer science...
Donald Knuth, Vol. IV - A, Combinatorial Algorithms, Section 4.2.1.6 (2011)

2) Verification: If we modify the decrypted file to-
Trees sprout up just about everywhere in Computer science...

Donald Knuth, Vol. IV - A, Combinatorial Algorithms, Section 4.2.1.6 (2011)

Verification Result: False

If we try the verification with more minor change, such as changing "Donald Knuth" to "**Donald Knuth**", that is,

Trees sprout up just about everywhere in computer science...

Donald Knuth, Vol. IV - A, Combinatorial Algorithms, Section 4.2.1.6 (2011)

Verification Result: False

Here the cipher, secret key, digital signature and the public key files are all Base64 encoded.

5 CONCLUSION AND FUTURE SCOPE

Small scale devices like smartphones often have less memory and slower CPUs, so they may be unable to use the same encryption methods as a traditional computer does, but that is no excuse for the lack of strong encryption. There are efficient cryptographic methods designed for small scale devices, such as Elliptic Curve Cryptography(ECC), which can be used. The primary benefit promised by ECC is a smaller key size, reducing storage and transmission requirements, i.e. an elliptic curve group could provide the same level of security afforded by an RSA-based system with a large modulus and correspondingly larger key: for example, a 256-bit ECC public key should provide comparable security to a 3072-bit RSA[5] public key. The hardest ECC scheme (publicly) broken to date had a 112-bit key for the prime field case and a 109-bit key for the binary field case. For the prime field case this was broken in July 2009 using a cluster of over 200 PlayStation 3 game console sand could have been finished in 3.5 months using this cluster when running continuously[11]. For the binary field case, it was broken in April 2004 using 2600 computers for 17 months[12]. In August 2013, it was revealed that bugs in some implementations of the Java class SecureRandom[13] sometimes generated collisions in the key value. This allowed a solution of the private key, in turn allowing stealing bitcoins[9] from the containing wallet on Android app implementations, which use Java and rely on ECDSA to authenticate transactions.

Many steganographic algorithms offer a high capacity for hidden messages, but are weak against visual and statistical attacks. Tools withstanding these attacks provide only a very small capacity. Matrix encoding and permutative straddling enable the user to decrease the necessary number of steganographic changes and to equalise the embedding rate in the steganogram. F5 accomplishes a steganographic proportion that exceeds 13% of the JPEG file size[7].

REFERENCES

[1] “Announcing the ADVANCED ENCRYPTION STANDARD (AES)”, Federal Information Processing Standards Publication 197, November 26, 2001

[2] Elaine Barker, Don Johnson, and Miles Smid, “Recommendation for Pair-Wise Key Establishment
Schemes Using Discrete Logarithm Cryptography (Revised)”, NIST Special Publication 800-56A March, 2007

[3] NIST Standard: http://csrc.nist.gov/publications/nistpubs/800-57/

[4] Secure Hash Standard (SHS), FIPS PUB 180-4

[5] R.L. Rivest, A. Shamir, and L. Adleman, “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”

[6] Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid, “Recommendation for Key Management- Part 1: General (Revision 3)”, NIST Special Publication 800-57

[7] Andreas Westfeld, “F5—A Steganographic Algorithm High Capacity Despite Better Steganalysis”, Technical University of Dresden, Institute for System Architecture, D-01062 Dresden, Germany

[8] Sayantan Majumdar, Abhisek Maiti, Asoke Nath , “New Secured Steganography Algorithm using Encrypted Secret Message inside QR^{TM} Code: System implemented in Android Phone”, published in International IEEE conference “Computational Intelligence and Communication Networks(CICN 2015)” held at Jabalpur, IEEE Explore page 1130-1134, Dec 12, 2015.

[9] https://bitcoin.org/en/alert/2013-08-11-android

[10] Base64: https://www.ietf.org/rfc/rfc3548.txt\

[11] http://lacal.epfl.ch/112bit_prime

[12] https://www.certicom.com/news-releases/300-solution-required-team-of-mathematicians-2600-computers-and-17-months-

[13] Java security libraries:
http://docs.oracle.com/javase/7/docs/api/java/security/package-summary.html

[14] ECC Brainpool Standard Curves and Curve Generation v. 1.0 19.10.2005

[15]http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf

[16] Sayantan Majumdar, Abhisek Maiti, Asoke Nath, "Secured Data Hiding Technique inside JPEG Image Embedded in QR^{TM} Code", published in International IEEE conference “Communication System and Network Technologies (CSNT 2016)” held at Chandigarh, India in 2016..