ISSN 2319-2720

Volume 1, No.2, September – October 2012
International Journal of Computing,Communications and Networking
Available Online at http://warse.org/pdfs/ijccn02122012.pdf

Analysis on AES Algorithm using symmetric cryptography

Amit Chaturvedi[1] Damodar Tiwari[2]
[1] M.Tech (CSE), BIST, Bhopal (M.P.), chat150478@gmail.com
[2]Dept. of CSE, BIST, Bhopal (M.P.), damodarptiwari21@gmail.com

ABSTRACT

In this paper we present the protection of long lived private key & public key using cryptographic schemes is one of the most important issues for information security. Any cryptographic scheme that reveals private key and public key will soon its security absolutely disintegrates. Public key are used in two different based, i.e. RSA and elliptic curve cryptography both are modular based arithmetic and private key is along random bit string and should be stored securely , some cryptographic are implemented in hardware such as an smart card IC(integrated circuit) needed to store the private key. Unfortunately, the security system in digital signature system, anyone obtains the victim's private key, authenticity and non repudiation can no longer be claimed. Next an Overview is given of the different hardware architectures which have been proposed in the literature.

1. INTRODUCTION

The rapid growth of networks, in terms of number and size, encourages and forces the linking together of more computers in order to share various kinds of data and exchange huge amount of information. Growing demands for security are characterizing the vast majority of communication and computer systems. The public key is a collection of technologies, process and organizational policies that support public key cryptography application. Public key cryptography system requires two separate keys, one of which is public and other is secure. The algorithms used for public key cryptography are based on mathematical relationships that have no efficient solution. To secure a system, various cryptographic techniques are used. These techniques are used to implement in hardware and software.

Cryptographic is a piece of parameter and determine the functional output Keys are also used in other cryptographic algorithms, such as digital signature schemes and message authentication codes. Private Key is confidential at all times and stored securely it is critical concept of to all public and private key that must be understood. Because the private key in public key cryptography for decrypting the code or signing messages is a long binary string it cannot be memorized by human beings. The storage area of is generally referred to as a key store.

- Public-key encryption: a message encrypted with a recipient's public key cannot be decrypted by anyone except a possessor of the matching private key - it is presumed that this will be the owner of that key and the person associated with the public key used. This is used to attempt to ensure confidentially

- Digital signatures: a message signed with a sender's private key can be verified by anyone who has access to the sender's public key, thereby proving that the sender had access to the private key and, therefore, is likely to be the person associated with the public key used. This also ensures that the message has not been tampered with (on the question of authenticity, see also message digest.

- Private Key: a private key is a fundamental component of any public key implementation. The network technique and password features are combined to dynamically generate the private key. Even if the adversaries obtain the storage device or the password, the probability of revealing the user's private key remains very difficult. The scheme is able to reduce the vulnerable password schemes when the password is revealed then the private key is lost. When a user enter password, the password are fed in neural network and then to generate the user's private key. Private Key is not stored in storage.

2. SYMMETRIC CRYPTOGRAPHY IN THE SECURE MEASUREMENT

Cryptographic algorithms that can be used in the measurement system consist in two operations. The former, encryption, is changing the easily-readable measurement data (plaintext) into the cipher, readable only by the authorized users. The latter, decryption, transforms the cipher into the plaintext. To be able to

read the encrypted data, the receiving node must know the key used to create the cipher. Depending on the number and application of the keys, the cryptography can be divided into symmetric and asymmetric. In the former, the same key is used for encryption and decryption. The latter requires separate keys for encryption and decryption. Asymmetric systems are currently very popular and their application in DMS should be considered. The most popular system is currently RSA, which, unfortunately is vulnerable to the attack using the quantum computer. Because the symmetric cryptography is fast, efficient and invulnerable to the brute force attacks, it seems to be a better choice to ensure the security of the measurement system. Its main disadvantage is the problem with the secure key distribution in the widely accessible environment, such as the Internet. When the environment is closed (which is the case in most industrial sites), the key may be distributed using the communication network. Therefore in the presented experiments, the symmetric cryptography was used. The main symmetric algorithms are Blowfish, DES and AES. Their security relies on the length of the key. Therefore currently the most popular is AES, which replaced the rest of the algorithms in most applications. Its usefulness will increase as the newest Intel processors have this system implemented as the machine instruction. The asymmetric structure of the MS with respect to the computational power imposes different requirements to the particular nodes, which is the main consideration in the paper. The measurement nodes are responsible for gathering the data from the external environment and sending them to the server. Therefore required processing power for the encryption and decryption is relatively low, as only one data vector must be encrypted and sent at the same time. Measurement server is responsible for simultaneous processing (decrypting) of multiple data streams sent from various nodes. Therefore it requires much more power and speed. Sending and receiving control information is usually less absorbing because of the small amount of information to send.

The simultaneous data processing in the server-side must be supported by both hardware and software. The former is realized by the multiprocessor and multi-core systems, which enable true parallel processing of the independent data streams. The latter are currently a standard and inexpensive solution present on the market, allowing execution of at least two different sets of instructions by separate cores. Also, the parallel computing is supported by the modern multiprocessor graphic cards (so-called GPGPU technology), which can be used to perform complex calculations (using CUDA architecture). The software support covers the design of the multithreaded applications – a single program can be decomposed into separate, independent fragments that are run simultaneously.
Modern programming languages, such as C++, C# or Java widely support multithreaded programming.

It is often important to acquire the data from the sensors and process them within the predefined time limits. To ensure such determinism, the RT system is needed. The latter can easily be implemented on the server (personal computer), and operate the measurement application designed in the integrated programming environments.

3. THERE ARE TWO ALGORITHMS TO IMPLEMENT ENCRYPTION.

In cryptography, protecting a private key is an important issue in any cryptographic scheme. In our experiment the private key is employed using 2048 bit RSA. RSA is a algorithm for public key cryptography. It is the first algorithm known to be suitable for signing as well as encryption and one of the first great advances. In public key cryptography RSA is widely used in electronic rules and is to be secure, given sufficiently long keys, and using up-to-date implementation. The following is the RSA key generation algorithm.

3.1 RSA key generation algorithm

1. Generate to Prime numbers a and b approx equal size;
2. 2. Compute n=ab and phi(n)=(a-1)(b-1);
3. Choose a integer e,1<e<phi such that gcd(e,phi);

4. Compute the secret exponent *d*, 1 < d < phi, such that ed ≡ 1 (mod phi);
5. The public key is (n, e) and the private key (d, a, b). Keep all the values d, p, q and phi secret. [We prefer sometimes to write the private key as (n, d) because you need the value of n when using d.]

- n is known as the *modulus.*
- e is known as the *public exponent* or *encryption exponent* or just the *exponent.*
- d is known as the *secret exponent* or *decryption exponent.*

Encryption

Sender A does the following:-

1. Obtains the recipient B's public key (n, e).
2. Represents the plaintext message as a positive integer *m*, $1 < m < n$
3. Computes the cipher text $c = m^e \bmod n$.
4. Sends the cipher text *c* to B.

Decryption
Recipient B does the following:-
Uses his private key (n, d)
to compute $m = c^d \bmod n$.
Extracts the plaintext from the message representative *m*.

Digital signing
Sender A does the following:-
Creates a *message digest* of the information to be sent. Represents this digest as an integer *m* between 1 and *n*-1.
Uses her *private* key (n, d) to compute the signature
$s = m^d \bmod n$.
Sends this signature *s* to the recipient, B.

Signature verification
Recipient B does the following:-
Uses sender A's public key (n, e) to compute integer
$v = s^e \bmod n$.
Extracts the message digest from this integer. Independently computes the message digest of the information that has been signed. If both message digests are identical, the signature is valid.

3.2 The RSA cryptosystem

Key generation

- Choose two distinct primes p and q of approximately equal size so that their product n = ab is of the required length.
- Compute $\varphi(n) = (a-1)(b-1)$.
- Choose a public exponent e, $1 < e < \varphi(n)$, which is co prime to $\varphi(n)$, that is, $\gcd(e, \varphi(n))=1$.
- Compute a private exponent d that satisfies the congruence $ed \equiv 1 \pmod{\&phi(n)}$.
- Make the public key (n, e) available to others. Keep the private values d, p, q, and $\varphi(n)$ secret.

RSA Encryption scheme

Encryption rule: cipher text,
$c = \text{RsaPublic}(m) = m^e \bmod n$,
$1 < m < n-1$

Decryption rule: plaintext,
$m = \text{RsaPrivate}(c) = c^d \bmod n$
Inverse transformation: m = RsaPrivate(RsaPublic(m))

RSA Signature scheme

Signing: signature,
$s = \text{RsaPrivate}(m) = m^d \bmod n$,
$1 < m < n-1$
Verification: check,
$v = \text{RsaPublic}(s) = s^e \bmod n$
Inverse transformation:
m = RsaPublic (RsaPrivate (m))

3.3 AES algorithm description

To implement the secure DMS in the presented research, the AES was used. It is the block cipher, approved by the National Institute of Standards and Technology in 2001 [6]. The purpose of the algorithm is to replace the older and less reliable algorithms, such as Data Encryption Standard (DES).
The algorithm operates on the 128-bit (16-byte) data blocks (plaintext, i.e. acquired samples, arrays and control instructions) and uses 128-, 192-, or 256-bit key to obtain the 128-bit cipher. Its hardware and software requirements are relatively small [9], so it can be successfully used in both types of the described measurement nodes of the system.

The implementation of the algorithm is based on the substitution– permutation network. Three algorithms are parts of the system: encryption, decryption and key expansion, presented in the following subsections. Because the main purpose of the paper is to present the parallel algorithm execution, it will be presented from the server point of view. However, the modifications can also be used in other nodes of the DMS equipped with multiple processing units.

3.4 AES encryption scheme
The encryption algorithm is used in the server to encrypt the control instructions and measurement parameters for the DAQ nodes. It transforms the input data (IN – measurements) into the cipher (OUT) using the expanded key W.

3.5 Pseudo code of the AES encryption

For both its Cipher and Inverse Cipher, the AES algorithm uses a round function that is composed of four different byte-oriented transformations: SubBytes, ShiftRows, MixColumns and AddRoundKey.

Inputs and outputs: The input and output for the AES algorithm each consists of sequences of 128 bits. The Cipher Key for the AES algorithm is a sequence of 128, 192 or 256 bits. The basic unit for processing in the AES algorithm is a byte (a sequence of eight bits), so the input bit sequence is first transformed into byte sequence. In the next step a two-dimensional array of bytes (called the State) is built. The State array consists of four rows of bytes, each containing Nb bytes, where Nb is the block size divided by 32 (number of words). All internal operations (Cipher and Inverse Cipher) of the AES algorithms are then performed on the State array, after which its final value is copied to the output (State array is transformed back to the bit sequence).

```
AEScipher(IN(4·Nb), OUT(4·Nb), W[Nb·(Nr+1)])
begin
    state[4,Nb] = IN
    AddRoundKey(state, W[0, Nb-1])
    for i = 1 to Nr-1
        SubBytes(state)
        ShiftRows(state)
        MixColumns(state)
        AddRoundKey(state, W[i·Nb, (i+1)·Nb-1])
    end
    SubBytes(state)
    ShiftRows(state)
    AddRoundKey(state, W[Nr·Nb, (Nr+1)·Nb-1])
    OUT = state
end
```

.

Cipher: Using round function, which is composed of four different byte-oriented transformations, the Cipher converts input data (the input data is first copied to the State array) to an unintelligible form called ciphertext. After an initial Round Key addition, the State array is transformed by implementing a round function with the final round differing slightly from the first Nr−1 rounds.

The round function is parameterized using a key schedule that consists of a one-dimensional array of four-byte words (Round Key) derived using the Key Expansion routine.

All Nr rounds (see Table 1) are identical with the exception of the final round, which does not include the Mix Columns transformation. Key Schedule: The AES algorithm takes the Cipher Key and performs a Key Expansion routine to generate a Key Schedule. The Key Expansion generates a total Nb(Nr + 1) words (Nr + 1 Round Keys).

Inverse Cipher: At the start of the Inverse Cipher, the input (ciphertext) is copied to the State array. After Round Key addition (the last Round Key is added), the State array is transformed by implementing a round function, that is composed of three different inverse transformations and AddRoundKey transformation (Round Keys are applied in the reverse order when decrypting), with the final round differing slightly from the first Nr – 1 rounds. So this procedure converts cipher text back to its original form called plaintext.

All Nr rounds are identical with the exception of the final round, which does not include the Inverse Mix-Columns transformation.

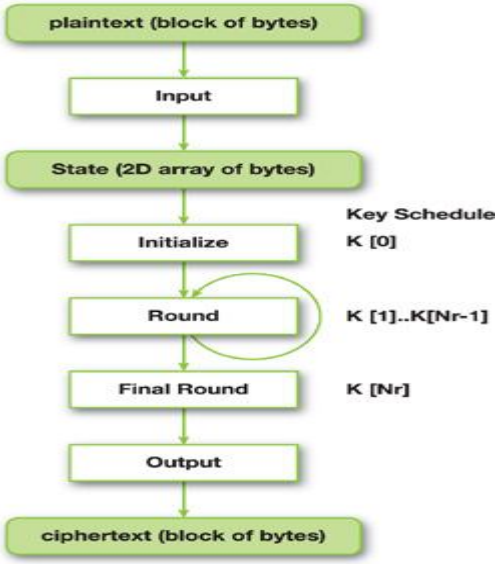

Figure 1: Process of Encryption.

4. RESULTS AND ANALYSIS

The password based schemes for protecting private key. The proposed schemes involve password features. To evaluate the system accuracy, two kinds of error rates through human test are applied.

Table 1: Speed of different versions of the AES encryption and decryption for the single plain text block and different key lengths.

Key length	Basic (ms)	ICA (ms)	ECA (ms)
AES-128	75.35	45.78	118.74
AES-192	96.74	51.59	150.36
AES-256	118.65	57.35	181.55

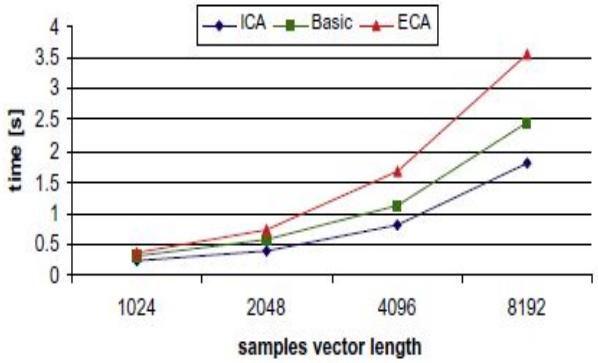

Figure 2: Time of the AES scheme execution for different lengths of samples vectors (128-bit key)

The influence of the waveform length on the encryption and decryption efficiency is presented in Fig. 2. The most efficient is the ICA version of the algorithm, which can be used in practical applications. Other versions are not that efficient, although ECA allows to fully control the cores.

When focusing on speed, ICA is the only justified modification. This configuration was also tested thoroughly for various key and waveform vector lengths.

The software solution is more flexible, as the multi-core processor can be used for multiple purposes in the measurement system. Therefore the cost of the proposed approach is related to the software design only. Moreover, the main task of RTOS is to ensure reliability, while speed is always a secondary issue.

Expressing the computation efficiency in processor cycles is difficult in such systems, as the implemented software was created in a high level programming language. A more popular practice is presenting the capacity in bps, which facilitates comparing efficiency of proposed approaches. Note that the speed of the AES strongly depends on the hardware it is run on. The processor used is a rather basic model. Newer and faster units ensure higher processing speeds. The structure of the modified AES indicates that the processor for the task should have four cores.

5. CONCLUSION

This paper enhanced the security of protecting a private key. This method based on specific key stroke feature biometrics and the non-linear handling ability of networks to dynamically generate the private key. We have presented an overview of the wide variety of architectures which have been designed to implement Public Key Cryptography. Creating a working implementation was a significant challenge in the 1980s; the number of hardware implementations that made it to prototype or production phase was very limited. In the 1990s, we have seen significant progress due to a combination of better algorithms and advances in VLSI technology. In addition, Elliptic Curve Cryptography may allow more compact implementations. Cryptographic hardware accelerator modules are now a commodity for Virtual Private Networks (VPNs) and e-commerce transactions; they can even be found in smartcard co-processors. In the area of smartcards, we have seen an increasing number of compact yet per formative co-processors for Public Key Cryptography.

In this paper, we are proposed two algorithm to implement the cryptographic i.e. RSA and AES algorithms, Because our proposed scheme is constructed using neural networks, the property of protection is reserved and a series of experiments showed that the brute force attacks by computer have low probability to generate the valid private key. As the same in the on line password guessing attacks, the attacks mounted by human or computers can be prevented easily by limiting the number of failed runs for generating the valid private key.

REFERENCES

1. Canberra in Lithuania – Securing the Maisiagala Waste Repository. <http://www.canberra.com/literature/438277.asp>.

2. AES on FPGA from the fastest to the smallest, cryptographic hardware and embedded systems – CHES 2005, Lecture Notes in Computer Science, vol. 3659/2005, pp. 427–440.

3. S. Mangard, M. Aigner and S. Dominikus. **A highly regular and scalable AES hardware architecture**, IEEE Transactions on Computers 52 (4) (2003), pp. 483–491.

4. S.A. Manavski. **Cuda compatible GPU as an efficient hardware accelerator for AES cryptography**, Proceedings IEEE International Conference on Signal Processing and Communications, Dubai, United Arab Emirates, 24–27 November 2007, pp. 65–68.

5. A.S. Tannenbaum and M. van Steen. **Distributed Systems- Principles and Paradigms**, Prentice Hall Inc., 2002.

6. Announcing the Advanced Encryption Standard (AES). <www.nist.gov>.

7. Advanced Encryption Standard (AES) Instructions Set Rev 2. <http://software.intel.com/en-us/articles/advanced-encryption-standard-aesinstructions-set/>.

8. W. B. Langdon. **A fast high quality pseudo random number generator for nVidia CUDA**, Proceedings of Genetic and Evolutionary Computation Conference, Montreal, Canada, 2009, pp. 2511–2514.

9. T. Good and M. Benaissa. **AES on FPGA: from the fastest to the smallest**, Proceedings CHES, Edinburgh, UK, August 29–September 1, 2005, pp. 427–440.

10. H. Kasahara and S. Narita. **Practical multiprocessor scheduling algorithms for efficient parallel processing**, IEEE Transactions on Computers 33 (11) (1984), pp. 1023–1029.

11.Yamanouchi. **AES Encryption and Decryption on the GPU**, GPU Gems3. <http://http.developer.nvidia.com/ GPUGems3/ gpugems3_ch36.html>.

12. D. Jinwala, D. Patel and K. Dasgupta. **Optimizing the Block Cipher and Modes of Operations Overhead at the Link Layer Security Framework in the Wireless Sensor Networks**, Information Systems Security, Springer, 2008. pp. 258–272.

13. Vartor Crypto-G library. <http://www.vartortech.com/cryptog.html>.

14. W. Winiecki and P. Bilski. **Multi-core programming approach in the real-time virtual instrumentation**, Proceedings IEEE I2MTC, Victoria, British Columbia, Canada, 12–15 May, 2008, pp. 1031-1036.

15. **Multicore Programming with Lab VIEW Technical Resource.** Guide.<http://ftp.ni.com/evaluation/labview/ekit/multicore_programming_resource_guide.pdf>.

16. P. Bilski and W. Winiecki. **Distributed real-time measurement system using time-triggered network approach**, International Journal of Computing 7 (2008), pp. 22–29.

17. P. Bilski and W. Winiecki. **Technika programowania wielordzeniowegow wirtualnych przyrza¸dach pomiarowych**, Przegla¸d Elektrotechniczny, No. 5/2008, pp. 269–272 (in Polish).

18. National Instruments Announces PXI-8110 3U Quad-Core Embedded Controller for PXI System. <http://embeddedsystemnews.com>.