ISSN 2394-739X
International Journal for Research in Science Engineering and Technology

IMPROVE THE CLOUD DATA SECURITY ARCHITECTURE USING SYMMETRIC KEY ALGORITHM

Prof. P.S. KARTHIKEYAN[1]
[1]Assistant Professor,
[1]Department of Computer Science,
[1]Park's College,
[1]Tirupur –5.

Abstract

There exist several types of cryptographic systems. In symmetric cryptography, both sending and receiving parties share the same secret key. While symmetric cryptography is computationally efficient, it requires that the shared secret key is distributed to all recipients in a secure way. Public-key cryptography, also known as asymmetric cryptography, uses two keys: a public key that can be freely shared and a private key, which is mathematically tied to the public key. In public-key signature schemes, the creator of the content uses the private key to sign the content. Afterwards the public key and the signature are distributed along with the content, allowing the recipient to verify the content's authenticity.

Encryption algorithms play a main role in information security systems. On the other side, those algorithms consume a significant amount of computing resources such as CPU time, memory, and battery power. This thesis provides evaluation of six of the most common encryption algorithms namely: AES (Rijndael), DES, and Blowfish.

A comparison has been conducted for those encryption algorithms at different settings for each algorithm such as different sizes of data blocks, different data types, battery power consumption, different key size and finally encryption/decryption speed. Simulation results are given to demonstrate the effectiveness of each algorithm.

Keywords: - Cryptography, Symmetric Key, DES, AES, Blowfish, Encryption, Decryption.

1. INTRODUCTION

Cryptography has been derived from the Greek words: kryptós, "hidden", and gráphein, "to write" - or "hidden writing". People who study and develop cryptography are called cryptographers and the study of cryptography is called cryptanalysis, or code breaking. Cryptography and cryptanalysis are sometimes grouped together under the umbrella term cryptology, encompassing the entire subject.

Some cryptographic methods rely on the secrecy of the algorithms; such algorithms are only of historical interest and are not adequate for real-world needs. All modern algorithms use a key to control encryption and decryption; a message can be decrypted only if the key matches the encryption key.

There are two classes of key-based encryption algorithms, symmetric (or secret-key) and asymmetric (or public-key) algorithms. The difference is that symmetric algorithms use the same key for encryption and decryption (or the decryption key is easily derived from the encryption key), whereas asymmetric algorithms use a different

key for encryption and decryption, and the decryption key cannot be derived from the encryption key (each user has a public key and a private key; the public key is made public while the private key remains secret; encryption is performed with the public key while decryption is done with the private key).

Cryptography plays a very vital role in keeping the message safe as the data is in transit. It ensures that the message being sent at one end remains confidential and should be received only by the intended receiver at the other end. Cryptography converts the original message in to non readable format and sends the message over an insecure channel. The people who are unauthorized to read the message try to break the non readable message but it is hard to do it so. The authorized person has the capability to convert the non readable message to readable one. The original message or the actual message that the person wishes to communicate with the other is defined as Plain Text.

The message that cannot be understood by anyone or meaningless message is what we call as Cipher Text. Encryption is the process of converting plaintext into cipher text with a key. A Key is a numeric or alpha numeric text or may be a special symbol. A decryption is a reverse process of encryption in which original message is retrieved from the cipher text. Encryption takes place at the sender end and Decryption takes place at the receiver end. Figure 1.1 shows the encryption/decryption process of a plaintext message. The input to the encryption process is plain text and that of decryption process is cipher text. First the plaintext is passed through the encryption algorithm which encrypts the plaintext using a key and then the produced cipher text is transmitted.

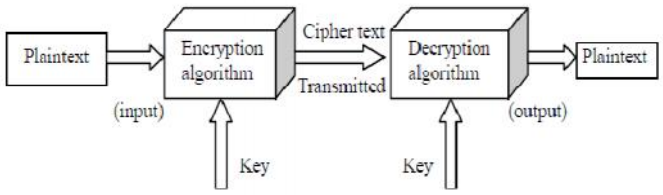

Figure 1: Encryption / Decryption Process

Symmetric algorithms can be divided into stream ciphers and block ciphers. Stream ciphers can encrypt a single bit of plaintext at a time, whereas block ciphers take a number of bits (typically 64 bits in modern ciphers), and encrypt them as a single unit. Asymmetric ciphers (also called public-key algorithms or generally public-key cryptography) permit the encryption key to be public (it can even be published in a newspaper), allowing anyone to encrypt with the key, whereas only the proper recipient (who knows the decryption key) can decrypt the message. The encryption key is also called the public key and the decryption key the private key or secret key.

2. CLOUD DATA SECURITY

Cloud computing is a distributed computing style which offer integration of web services and data centres. There are several major cloud computing providers including Amazon, Google, Yahoo, Microsoft and others that are providing cloud computing services. Amazon web services was first to provide an architecture for cloud based services in 2002 and after that advancements and new models for cloud architecture had been proposed and implemented. There have been many techniques of storing data on server storage. Such data storages provided by cloud service providers have to ensure client about Confidentiality, Integrity and Availability of data. Confidentiality: Confidentiality refers to keeping data private.

Cloud Storage: Cloud storage specifies the storage on cloud with almost inexpensive storage and backup option for small enterprise. The actual storage location may be on single storage environment or replicated to multiple server storage based on importance of data. Typical cloud storage system architecture includes a master control server and various clients. The mechanism model of cloud storage consists of four layers: storage layer which stores the data, basic management layer which ensures security and stability of cloud storage itself, application interface layer which provides application service platform, and access layer which provides the access platform. The basic cloud storage environment represented as follows:

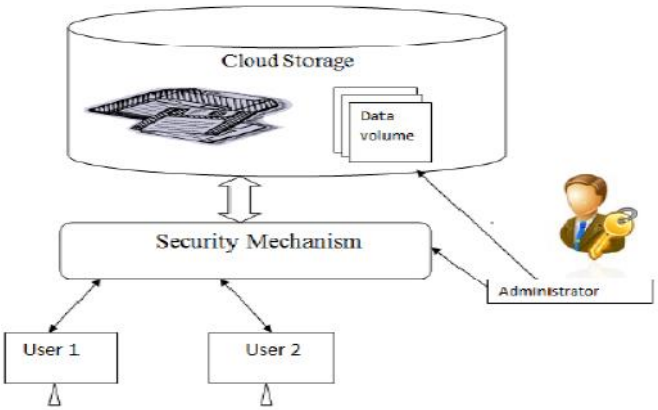

Figure 2: Cloud Storage Environment

Cloud computing offers a prominent service for data storage known as cloud storage. The flow and storage of data on the cloud environment in plain text format may be main security threat. So, it is the responsibility of cloud service providers to ensure privacy and security of data on storage as well as network level. The following three parameters confidentiality, integrity and availability decide whether security and privacy of data stored on cloud environment is maintained or not.

Cryptography application supports symmetric and asymmetric encryption algorithm to encrypt/decrypt data for uploading/downloading within cloud storage. A username and password based authentication mechanism for users and digital signature scheme for data authenticity are defined within cloud architecture.

Cloud computing provides on-demand resource access from a shared pool of computing resources such as; hardware and software for efficient manage. By outsourcing the user data to the public cloud environment, this decreases the control of data for data owner. To maintain the control of data in rest or data in motion within networks, offers more advantages for data security. Protecting data in the cloud, authentication and integrity, access control, encryption, integrity checking and data masking are some of the data protection techniques.

Cryptography is the one of the efficient method for data security in cloud computing. This includes the design and implementation of an efficient encryption and decryption algorithms. In symmetric cryptography, before outsourcing data to cloud server is encrypted into cipher text using secret key and later user decrypted using same shared secret key.

3. DES

Secrecy is at the heart of cryptography. Encryption is a practical means to achieve information secrecy. Modern encryption techniques are mathematical transformations (algorithms) which treat messages as numbers or algebraic elements in a space and transform them between a region of "meaningful messages" and a region of "unintelligible messages". A message in the meaningful region and input to an encryption algorithm is called clear text and the unintelligible output from the encryption algorithm is called ciphertext.

DES (and most of the other major symmetric ciphers) is based on a cipher known as the Feistel block cipher. This was a block cipher developed by the IBM cryptography researcher Horst Feistel in the early 70's. It consists of a number of rounds where each round contains bit-shuffling, non-linear substitutions (S-boxes) and exclusive OR operations. Most symmetric encryption schemes today are based on this structure (known as a feistel network).

As with most encryption schemes, DES expects two inputs - the plaintext to be encrypted and the secret key. The manner in which the plaintext is accepted and the key arrangement used for encryption and decryption, both determines the type of cipher it is. DES is therefore a symmetric, 64 bit block cipher as it uses the same key for both encryption and decryption and only operates on 64 bit blocks of data at a time5 (be they plaintext or ciphertext). The key size used is 56 bits, however a 64 bit (or eight-byte) key is actually input. The least significant bit of each byte is either used for parity (odd for DES) or set arbitrarily and does not increase the security in any way. All blocks are numbered from left to right which makes the eight bit of each byte the parity bit.

Once a plain-text message is received to be encrypted, it is arranged into 64 bit blocks required for input. If the number of bits in the message is not evenly divisible by 64, then the last block will be padded. Multiple permutations and substitutions are incorporated throughout in order to increase the difficulty of performing a cryptanalysis on the cipher. However, it is generally accepted that the initial and final permutations offer little or no

contribution to the security of DES and in fact some software implementations omit them.

It was adopted in 1977 by the National Bureau of Standards (NBS), now National Institute of Standards and Technology (NIST), as Federal Information Processing Standard 46 (FIPS PUB 46). In 1971, IBM's team under Horst Feistel leadership developed algorithm LUCIFER, operating on 64-bit blocks with 128-bit key. Further, IBM's team leaded by Walter Tuchman and Carl Meyer revised LUCIFER to make it more resistant to cryptanalysis, but they reduced key size to 56 bits. In 1973, NBS issued a request for proposals for a national cipher standard. IBM submitted results of its Tuchman-Meyer project. This was by far the best algorithm proposed and was adopted in 1977 as Data Encryption Standard. In 1994, NIST reaffirmed DES for federal use for another 5 years. In 1999, NIST issued a new version of its standard (FIPS PUB 46-3) that indicated that DES should only be used for legacy systems and that triple DES be used.

DES (Data Encryption Standard) algorithm purpose is to provide a standard method for protecting sensitive commercial and unclassified data. This same key used for encryption and decryption process. DES algorithm consists of the following steps

1. DES accepts an input of 64-bit long plaintext and 56-bitkey (8 bits of parity) and produce output of 64 bit block.
2. The plaintext block has to shift the bits around.
3. The 8 parity bits are removed from the key by subjecting the key to its Key Permutation.
4. The plaintext and key will processed by following
   - i. The key is split into two 28 halves.
   - ii. Each half of the key is shifted (rotated) by one or two bits, depending on the round.
   - iii. The halves are recombined and subject to a compression permutation to reduce the key from 56 bits to 48 bits. This compressed keys used to encrypt this round's plaintext block.
   - iv. The rotated key halves from step 2 are used in next round.
   - v. The data block is split into two 32-bit halves.
   - vi. One half is subject to an expansion permutation to increase its size to 48 bits.
   - vii. Output of step 6 is exclusive-OR'ed with the 48- it compressed key from step 3.
   - viii. Output of step 7 is fed into an S-box, which substitutes key bits and reduces the 48-bit block back down to 32-bits.
   - ix. Output of step 8 is subject to a P-box to permute the bits.

4. AES

Like DES, AES is a symmetric block cipher. This means that it uses the same key for both encryption and decryption. However, AES is quite different from DES in a number of ways. The algorithm Rijndael allows for a variety of block and key sizes and not just the 64 and 56 bits of DES' block and key size. The block and key can in fact be chosen independently from 128, 160, 192, 224, 256 bits and need not be the same. However, the AES standard states that the algorithm can only accept a block size of 128 bits and a choice of three keys - 128, 192, 256 bits. Depending on which version is used, the name of the standard is modified to AES-128, AES-192 or AES-256 respectively. As well as these differences AES differs from DES in that it is not a feistel structure. Recall that in a feistel structure, half of the data block is used to modify the other half of the data block and then the halves are swapped. In this case the entire data block is processed in parallel during each round using substitutions and permutations.

A number of AES parameters depend on the key length. For example, if the key size used is 128 then the number of rounds is 10 whereas it is 12 and 14 for 192 and 256 bits respectively. At present the most common key size likely to be used is the 128 bit key. This description of the AES algorithm therefore describes this particular implementation.

Rijndael was designed to have the following characteristics:

- Resistance against all known attacks.
- Speed and code compactness on a wide range of platforms.
- Design Simplicity.

In a first round of evaluation, 15 proposed algorithms were accepted. A 2nd round narrowed to 5 algorithms. NIST completed its evaluation process and published a final standard (FIPS PUB 197) in November, 2001. NIST selected Rijndael as the proposed AES algorithm. The 2 researches of AES are Dr. Joan Daemon and Dr. Vincent Rijmen from Belgium.

Advanced Encryption Standard (AES) algorithm is not only for security but also for great speed. Both hardware and software implementation are faster still. A New encryption standard recommended by NIST to replace DES is AES. Encrypts data blocks of 128 bits in 10, 12 and 14 round depending on key size. It can be implemented on various platforms especially in small devices. It is carefully tested for many securities applications.

Algorithm Steps: These steps used to encrypt 128-bit block

1. The set of round keys from the cipher key.
2. Initialize state array and add the initial round key to the starting state array.
3. Perform round = 1 to 9: Execute Usual Round.
4. Execute Final Round.
5. Corresponding cipher text chunk output of Final Round Step.

The algorithm begins with an Add round key stage followed by 9 rounds of four stages and a tenth round of three stages. This applies for both encryption and decryption with the exception that each stage of a round the decryption algorithm is the inverse of it's counterpart in the encryption algorithm. The four stages are as follows:

1. Substitute bytes
2. Shift rows
3. Mix Columns
4. Add Round Key

The tenth round simply leaves out the Mix Columns stage. The first nine rounds of the decryption algorithm consist of the following:

1. Inverse Shift rows
2. Inverse Substitute bytes
3. Inverse Add Round Key
4. Inverse Mix Columns

Again, the tenth round simply leaves out the Inverse Mix Columns stage.

5. BLOWFISH

The data transformation process for Pocket Brief uses the Blowfish Algorithm for Encryption and Decryption, respectively. The details and working of the algorithm are given below.

Blowfish is a symmetric block cipher that can be effectively used for encryption and safe guarding of data. It takes a variable-length key, from 32 bits to 448 bits, making it ideal for securing data. Blowfish was designed in 1993 by Bruce Schneier as a fast, free alternative to existing encryption algorithms. Blowfish is unpatented and license-free, and is available free for all uses.

Blowfish Algorithm is a Feistel Network, iterating a simple encryption function 16 times. The block size is 64 bits, and the key can be any length up to 448 bits. Although there is a complex initialization phase required before any encryption can take place, the actual encryption of data is very efficient on large microprocessors.

Blowfish is a variable-length key block cipher. It is suitable for applications where the key does not change often, like a communications link or an automatic file encryptor. It is significantly faster than most encryption algorithms when implemented on 32-bit microprocessors with large data caches.

The data transformation process for PocketBrief uses the Blowfish Algorithm for Encryption and Decryption, respectively. The details and working of the algorithm are given below.

Blowfish is a symmetric block cipher that can be effectively used for encryption and safeguarding of data. It takes a variable-length key, from 32 bits to 448 bits, making it ideal for securing data. Blowfish was designed in 1993 by Bruce Schneier as a fast, free alternative to existing encryption algorithms. Blowfish is unpatented and license-free, and is available free for all uses. Blowfish Algorithm is a Feistel Network, iterating a simple encryption function 16 times. The block size is 64 bits, and the key can be any length up to 448 bits. Although there is a complex initialization phase required before any encryption can take place, the actual encryption of data is very efficient on large microprocessors.

Blowfish is a variable-length key block cipher. It is suitable for applications where the key does not change often, like a communications link or an automatic file encrypt or. It is significantly faster than most encryption algorithms when

implemented on 32-bit microprocessors with large data caches.

The Blowfish Algorithm:

- Manipulates data in large blocks.
- Has a 64-bit block size.
- Has a scalable key, from 32 bits to at least 256 bits.
- Uses simple operations that are efficient on microprocessors.

e.g., exclusive-or, addition, table lookup, modular- multiplication. It does not use variable-length shifts or bit-wise permutations, or conditional jumps.

6. PERFORMANCE EVALUATION

Several performance metrics are collected: 1) Encryption time; 2) CPU process time; and 3) CPU clock cycles and battery power,4)Throughput,5)Different data types,6)Different size of data block.

- **Encryption Time**

Encryption Time is one of a performance metric which is defined as the amount of time required for converting plaintext message to cipher text at the time of encryption. Encryption time is used to calculate the throughput of an encryption scheme. It indicates the speed of encryption. The throughput of the encryption scheme is calculated as the total plaintext in bytes encrypted divided by the encryption time.

- **Decryption Time**

Decryption Time is one of a performance metric which is defined as the amount of time required for converting the cipher text into the plain text at the time of decryption.

- **Throughput**

The throughput of the encryption scheme is calculated by dividing the total plaintext in Megabytes encrypted on the total encryption time for each algorithm in.

$$\text{Throughput} = \frac{\text{Total Plaintext in MegaBytes}}{\text{Encryption Time}}$$

The higher the value of throughput more is the efficiency of encrypting any text with an encryption algorithm.

- **CPU Process Time**

The CPU process time is the time that a CPU is committed only to the particular process of calculations. It reflects the load of the CPU. The more CPU time is used in the encryption process, the higher is the load of the CPU. The CPU clock cycles are a metric, reflecting the energy consumption of the CPU while operating on encryption operations. Each cycle of CPU will consume a small amount of energy.

The following tasks that will be performed are shown as follows:

- A comparison is conducted between the results of the selected different encryption and decryption schemes in terms of the encryption time at two different encoding bases namely; hexadecimal base encoding and in base 64 encoding.
- A study is performed on the effect of changing packet size at power consumption during throughput for each selected cryptography algorithm. Key Length is an easy objective, numeric metric to adopt since key size is universally expressed as a number of bits.

In fact, every extra key bit generally doubles the number of possible keys and therefore increases the effort required for a successful brute force attack against most symmetric algorithms.

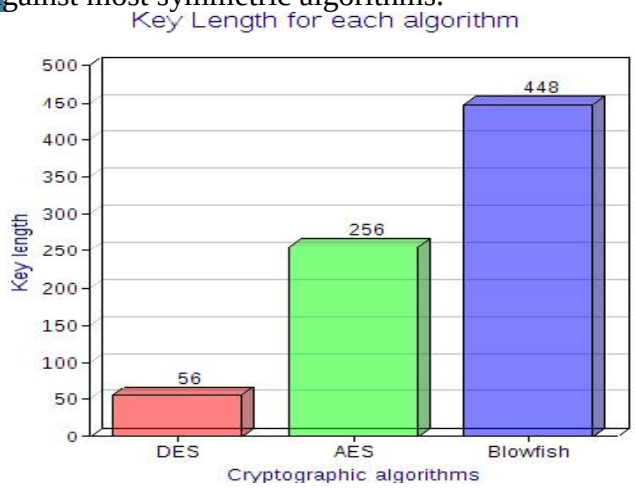

Figure 3: Key Length comparison for each cryptographic algorithm

Attack Steps is defined as the number of steps required to perform the best known attack. The number of steps helps determine the time that might be required for a successful attack, using a particular processor, without having to actually run the attack on the algorithm, which may not be feasible.

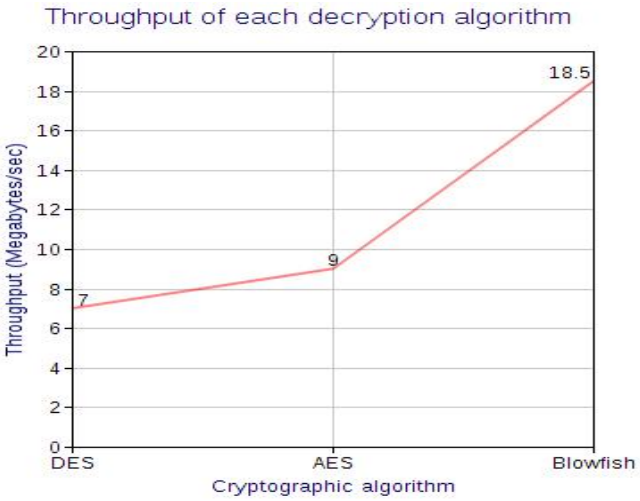

Figure 4: Throughput for each cryptographic algorithm

The performance matrices are encryption and decryption time. The encryption time is defined as the time that an encryption algorithm takes to generate a cipher text from plain text and decryption time is defined as the time that an encryption algorithm takes to generate plain text from cipher text.

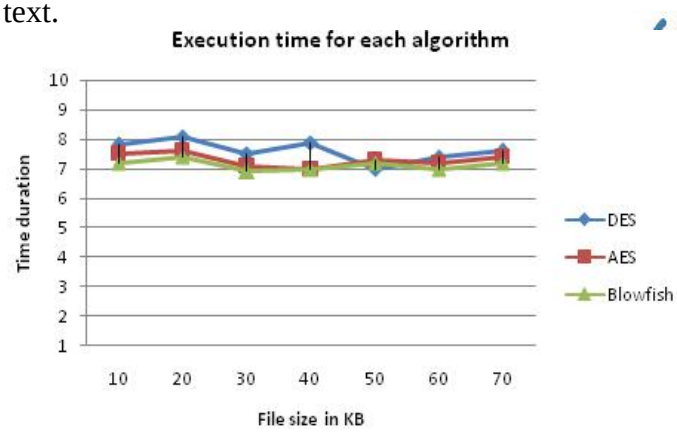

Figure 5: Execution time for each cryptographic algorithm

Algorithm strength was chosen as the name of a scale developed for expressing the overall measurement of a cryptographic algorithm's strength, worth or value, even though the scale has to be defined and expressed in subjective, adjectival terms. This is the only subjective, adjectival characteristic scale for algorithm specification that was developed during this pilot. The Algorithm Strength (AS) metric is intended for use by experienced cryptographers to specify, or express an evaluation of, algorithm strength values.

A determination of algorithm strength must take into consideration the best known methods of attack and the length of time required to carry out those attacks using current technology.

7. CONCLUSION

The objective of the paper is to provide a performance analysis between symmetric key cryptography algorithms: DES, AES and Blowfish. The analysis has been conducted by running several encryption settings to process different sizes of data blocks to evaluate the algorithm's speed for encryption and decryption. Each algorithm is designed and executed in these modes. The variation is provided in data size given by the user. The data is retrieved from various text files to calculate the time consumed by each algorithm to process the retrieved data.

The presented simulation results showed that Blowfish has a better performance than other common encryption algorithms used. Since Blowfish has not any known security weak points so far, this makes it an excellent candidate to be considered as a standard encryption algorithm. AES showed poor performance results compared to other algorithms since it requires more processing power.

REFERENCES

[1] M. Abdalla, M. Bellare, and P. Rogaway, "The oracle Diffie-Hellman assumptions and an analysis of DHIES," in *Topics in Cryptology – RSA Conference Cryptographers' Track (RSA-CT 2001)*, vol. 2020 of *Lecture Notes in Computer Science*, pp. 143–158, Springer-Verlag, 2001.

[2] G. Anastasi, A. Falchi, A. Passarella, M. Conti, and E. Gregori, "Performance measurements of motes sensor networks," in *Proceedings of the 7th International Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2004)*, pp. 174–181, ACM Press, 2004.

[3] R. Anderson and M. Kuhn, "Tamper resistance–a cautionary note," in *Proceedings of the 2nd USENIX Workshop on Electronic Commerce*, pp. 1–11, USENIX Association, 1996.

[4] "Anonymity and Privacy in Electronic Services (APES), IWT/STWW Project." *h*https://www.cosic.esat.kuleuven.ac.be/apes/*i*.

[5] D. Balfanz, G. Durfee, N. Shankar, D. K. Smetters, J. Staddon, and H. C. Wong,

"Secret handshakes from pairing-based key agreements," in *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pp. 180–196, IEEE, 2003.

[6] P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott, "Efficient algorithms for pairing-based cryptosystems," in *Advances in Cryptology - CRYPTO 2002*, vol. 2442 of *Lecture Notes in Computer Science*, pp. 354–368, Springer-Verlag, 2002.

[7] D. J. Barrett, R. E. Silverman, and R. G. Byrnes, *SSH, The Secure Shell: The Definitive Guide, Second Edition*. O'Reilly, 2005.

[8] L. Batina, J. Lano, N. Mentens, B. Preneel, I. Verbauwhede, and S. B. ¨ Ors, "Energy, performance, area versus security trade-offs for stream ciphers," in *ECRYPT Workshops – The State of the Art of Stream Ciphers (SASC 2004)*, pp. 302–310, 2004.

[9] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proceedings of the 1th ACM Conference on Computer and Communications Security (CCS 1993)*, pp. 62–73, ACM Press, 1993.

[10] M. Bellare and P. Rogaway, "Optimal asymmetric encryption," in *Advances in Cryptology - EUROCRYPT 1994*, vol. 950 of *Lecture Notes in Computer Science*, pp. 92–111, Springer-Verlag, 1995.

[11] M. Bellare and P. Rogaway, "The exact security of digital signatures: How to sign with RSA and Rabin," in *Advances in Cryptology – EUROCRYPT 1996*, vol. 1070 of *Lecture Notes in Computer Science*, pp. 399–414, Springer-Verlag, 1996.

[12] M. Bellare and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm," in *Advances in Cryptology - ASIACRYPT 2000*, vol. 1976 of *Lecture Notes in Computer Science*, pp. 531–545, Springer-Verlag, 2000.

[13] M. Bellare, P. Rogaway, and D. Wagner, "The EAX mode of operation," in *Fast Software Encryption (FSE 2004)*, vol. 3017 of *Lecture Notes in Computer Science*, pp. 389–407, Springer-Verlag, 2004.

[14] F. Bennett, D. Clarke, J. B. Evans, A. Hopper, A. Jones, and D. Leask, "Piconet: embedded mobile networking," *IEEE Personal Communications*, vol. 4, no. 5, pp. 8–15, 1997.