Secured Encryption - Concept and Challenge

Govind Singh Tanwar
Department of Information Technology
Govt. Engineering College
Bikaner, Bikaner (Raj), India

Ganesh Singh
Department of Master of Computer Application
Govt. Engineering College
Bikaner, Bikaner (Raj), India

Vishal Gaur
Department of Master of Computer Application
Govt. Engineering College
Bikaner, Bikaner (Raj), India

ABSTRACT
This paper contains a complications related to access systems and their functionality. Several cryptographic algorithms were implemented using the public library Lib-Tom-Crypt and benchmarked on an ARM7-processor platform. The common coding schemes in use were ECC and RSA (asymmetric coding schemes), AES, 3DES and two fish (symmetric algorithms). The benchmark considered both code size and speed of the algorithms. The two asymmetric algorithms, ECC and RSA, are possible to be used in an ARM7 based access system. Although, both technologies can be configured to finish the calculations within a reasonable time-frame of 10 Sec.

Key Words
Cryptography, Public ARM7-processor, ECC, RSA.

1. INTRODUCTION

When designing and implementing a multi-user access system there are several factors to consider, of which *key management* is one of the most important and most difficult to solve. Key management is the problem of making sure that each user has the correct key with the proper security level, at the right time. The access system has to distribute and keep track of the keys, making sure that no keys are lost or compromised. The simplest and most common access system is the traditional mechanical lock cylinder, which is a simple and very reliable system. However, key management in a mechanical multi-user system is a momentous task and therefore several electronic solutions exist solving this problem. These electronic solutions are mostly based on *online* networked lock-terminals communicating with a centrally managed key server. The decision of who to grant access is made by the server. This is practical and secure when all lock units are gathered in the same geographical area, such as a building or within company grounds. The mechanical lock system lacks several important features compared to its electronic counterpart. The system is not suited for a multi-user environment since all authorized users have a copy of the same key (from the lock unit's viewpoint). Lost keys cannot be blocked and have unlimited lifespan. The keys can also easily be copied by corrupt users with physical access to the key. Or even within visual range of the key, according to a recent thesis work in Linköping University [1], which showed that it is possible to reproduce a key from a single photo. Some companies deal with hundreds of keys daily, which results in a highly complex key management solution requiring large resources, both in additional work done by employees and in maintenance costs. This also often leads to security flaws due to the complexity of the system, mostly related to the human factor. The system has to determine which employee to give which key, and often multiple employees have to access the same location. It also has to detect lost keys in the system and issue replacement cylinders whenever needed. If a key is knowingly compromised this results in a time consuming and expensive task changing the cylinder in the lock and distributing new keys to each user. When there is a security breach due to a compromised key in this mechanical system there is no trace to which employee was responsible, since everyone has the same key and the keys can easily be copied. Therefore the focus of this thesis work will be on the problems associated with access systems with geographically distributed objects and locations.

1.1 Limitations of resources

There are two project members. The project ranges over a period of 20 weeks with a contracted deadline on November 30 in 2007. The budget is approximately 10.000 SEK which is invested by Combitech AB. The project has four mentors, two provided by Combitech and two from the University of Linköping. Combitech AB provides a workplace equipped with computers, stationery and lab equipment.

2. ACCESS SYSTEMS

The description of general access systems and their functions. Furthermore are current access system solutions presented and how to improve their functionality. The suggestive improvements presented as *concept requirements* are based on the market investigation and personal experiences. The aim of this chapter is to present information of the requirements of a new system.

2.1 The principle of an access system

When discussing access systems, it is important to clarify their intended function and purpose. An access system is a system meant to protect locations within a domain against unauthorized access, while still granting access to authorized users.

2.1.1 Basic functionality

The primary function of an access system is, as mentioned, to keep unauthorized people out and to grant access to valid users. To make difference if the user is valid or not, the system must use some form of authentication. The authentication is

done by analyzing information provided by the user. This information can be divided into three different categories, *identity codes*, which are presented as circles in *Figure* 2.1. These are often used in different combinations and the four most common are described below. Although the use of passport and driving licenses are common, they are not considered within this description of general access systems.

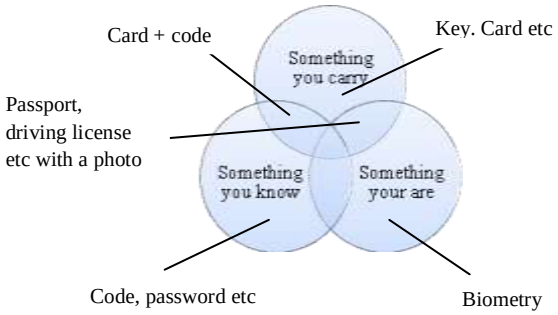

Figure 2.1: *The figure is describing the three different identity methods used in access systems. More information about these methods can be found in a Securitas document [2]*

a.) *Memory code* - In this case the user memorizes the code and only the code is required to get access. This type of identity control is weak and is only used in areas where there are many users who need to be able to delegate access to others, such as an entrance to an apartment building.

b.) *Carried code* - In this system the user brings a physical token, often a key card, although other solutions exist. The token contains information loaded by the access system to grant access.

c.) *Carried code with memory code* - This method is a combination of the two previous methods. The token's information includes the memory code which the user has to provide each time an access is requested.

d.) Biometric code - The most common form of biometric codes is the visual inspection of the photograph on an identification card, such as a driving license.

2.1.2 Additional functions

Some additional functions which are desirable in an access system include:

a.) *Time-lock* - Limiting the validity period of the key. A user can be restricted to certain hours of the day, or the key lifespan could be limited.

b.) *Logging* - An important feature in access systems, which has authentication, is to keep a log-file of all access events by all users. This provides tracking functionality if a key and/or a user is compromised.

c.) *Authorization level* - This enables the system to have a hierarchy among the users. A certain security clearance can be required to access a certain location. This can also be implemented as a system where users can be mapped to which locations they have access to.

2.1.2 Security

The security is a measurement of how difficult it is to break into the object which the system protects. Due to the complexity of all the possible factors affecting the total security, it is only feasible to estimate the security of the specific access system. The goal for an attacker of an access system is to get access. This could be done by going around the actual access system and through some other weaker point of the total defense, e.g. breaking a window or subverting a valid user. A way of modeling the threats is to use an *attack tree* [3].

2.2 Available solutions

Available solutions based on the four methods used for identification are presented in this chapter. The solutions can be categorized in three different groups of access systems; unintelligent off-line systems, intelligent off-line systems and online systems.

2.2.1 Unintelligent off-line systems

These systems are mostly based on the ordinary lock cylinder system widely used within many different areas. Another access system within this category is the key pad.

a.) Mechanical lock cylinder - The mechanical lock cylinder is as mentioned frequently used in geographical distributed areas. The primary advantage of such systems is the dependability. It is robust and well tested and it works in all kinds of weather in contrast to other systems. Other systems often need a backup system and they often require electricity. The mechanical lock cylinder is also very easy to use, since it is the most common system; generally everybody knows how to use it. Disadvantages of this system are for example the maintenance of the system.

b.) Keypad - The keypad is another unintelligent off-line system which is quite common. It has the same verification problem though it only controls the dialed code.

2.2.2 On-line systems

These systems are connected to some central unit of intelligence. The most common use of such a system is the use of the entry cards. This system is more suited for multi-user environments than the other solutions. There are several solutions which are based on the same concept, e.g. radio-frequency identification (RFID) tags, although the entry cards is the most common and thereby used for this description.

3. CRYPTOGRAPHY

Cryptography is a large subject which can be confusing at best sometimes. The term cryptography (or cryptology derived from Greek *kryptós* "hidden" and *gráfo* "write") is the study of message secrecy. The opposite is cryptanalysis which is the study of methods of how to reverse the encrypted message. This chapter aims to give some background on the encryption techniques and application areas considered during the design process of the system.

3.1 Basic Cryptography

There is a tradition within the area of cryptography of using the names *Alice*, *Bob* and *Eve* to represent the different roles played by the communicating devices on a communication channel. By definition *Alice* sends messages to *Bob* and *Eve* is assumed to be eavesdropping on all messages sent on the communication channel.

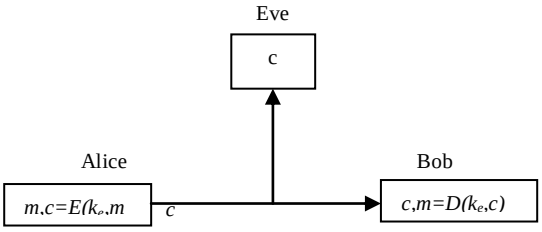

Figure 3.1: The figure describes the relationship between Alice, Bob and Eve and the generic settings for encryption. These roles, representing the different parts affecting the communication, are common within the area of cryptography. More figures and description about the different roles can be found in "Practical Cryptography" by Bruce Schneier and Niels Ferguson [4].

3.1.1 Encryption

Encryption is used to communicate securely over an insecure communication channel. Consider *Alice* communicating with *Bob*. Any message from *Alice* to *Bob* is also received by *Eve*. To prevent *Eve* from understanding the message an encryption function $E(K_{enc}, m)$ is used to transform the so called *Plaintext*, *m*, into the unreadable *Ciphertext*, *c*, where K_{enc} represents the encryption key which is to be known only by the authorized communicants and not by *Eve*. In order for *Bob* to be able to read the message, a decryption function $D(K_{enc}, c)$ is used to make the reverse transformation from *Ciphertext* into *Plaintext*, see figure 3.1. Both these transformations require a cipher which is an algorithm used for performing encryption and decryption. The key is as mentioned to be kept secret although the algorithm can and should be public.

3.1.2 Authentication

Using hash functions in communication enables the recipient to verify the integrity of the message against the hash-digest sent along with the message. However, how can *Bob* be sure the message really is from *Alice*? *Eve* could have changed the message and recalculated a new hash-digest. A solution is to use *message authentication codes*, MAC, which is basically a hash function with an authentication key, K_{auth}. The fixed length MAC-digest is calculated using the MAC function $h(K_{auth}, m)$ and is sent together with the message. When *Alice* wants to send a message she computes the MAC, $a = h(K_{auth}, m)$ and sends the complete message as $(m||a)$. When *Bob* receives the message $(m_{rcv}||a_{rcv})$ he calculates his own MAC $a_{bob} = h(K_{auth}, m_{rcv})$ and verifies $a_{bob} = a_{rcv}$. If the codes are different he discards the message. When *Bob* does this he verifies the integrity and the authenticity of the message. Since K_{auth} is a shared secret, *Bob* can verify the authenticity of the message. This means he can verify the sender really is *Alice*, since only she has the other key.

3.1.3 Asymmetric encryption

Previous sections have described symmetric encryption, where *Alice* and *Bob* share the same secret key, K_{enc}. However, they can not send the key over the communication channel. Since *Eve* is listening in, they have to meet in person to synchronize keys. Keys have a limited lifespan and *Alice* may have many people to communicate with, so exchanging keys is a tedious task. A solution to the problem of key distribution is asymmetric cryptography, commonly known as Public-Key cryptography. In asymmetric encryption both *Alice* and *Bob* have two paired keys, a public key, *P*, with a corresponding secret or private key, *S*. Both publish their public keys somewhere for everyone to see, while they keep the private key secret. The encryption technique is basically a one-way function, so a message encrypted with a public key can only be decrypted with the corresponding private key.

3.1.4 Digital signatures

A digital signature is the way to check data integrity with asymmetric cryptography. It works in a similar way as *message authentication codes*, MAC. The signing process uses a hash function producing a fixed sized hash-digest. The signing function encrypts the hash-digest using the private key into a signature and the signature can be verified by anyone with the sender's public key. When *Alice* signs a message for *Bob*, she uses an appropriate hash function to generate the hash-digest, $h_d = HASH(m)$. She then signs h_d using her private key, S_{Alice}, giving $s = \sigma(S_{Alice}, h_d)$ and transmits the signature, *s*, together with the message. *Bob* can then verify the signature by computing his own hash-digest of the received message, $h_{d,comp} = HASH(mrcv)$. The result should be the same as returned from the signature verification algorithm, $h_{d,rcv} = v(P_{Alice,} s_{rcv})$. If $h_{d,comp}$ equals $h_{d,rcv}$ the message integrity is verified, otherwise it has been tampered with. Since the message is signed with a private key and a message is encrypted with a public key the same key-pair should never be used for both applications. Each user has to have at least two separate key-pairs, one for encrypting messages and one for signing them.

3.1.5 Public key infrastructure

The problem with asymmetric keys is authentication. How can *Alice* find *Bob*'s public key when she wants to send him a message? *Alice* may never have met *Bob*, but she wants to send him a secure message. How can she be sure that the public key she finds really belongs to *Bob* and not *Eve* posing as *Bob*? They have to rely on a trusted third party, which both trust, to supply them

with the correct keys. The trusted third party is called a *certificate authority*, CA, and maintains a *public key infrastructure*, PKI. The CA collects user information and the public key from a user it recognizes into a file which is signed by the CA. This signed file is called a *certificate*.

3.1.6 Bits of security

To be able to compare the security level between different cryptography technologies, the concept of *work factor* is defined and is measured in *bits of security*. It describes the amount of work the fastest currently available attack would require on the algorithm with the specific key. The fastest attack on an algorithm with *N* bits of security would require 2*N* calculated steps.

3.2 Symmetric Cryptography

Symmetric key encryption was briefly mentioned in the previous section 3.1. This section describes the symmetric encryption, its algorithms and variations in more detail. Furthermore, the ciphers and their modes will be presented and data encryption standards will be described. In symmetric key encryption the sender and the receiver use the same key, *Kenc* (or rarely different keys, but related in an easily computable way). Other names for symmetric key encryption are one-key, single-key and private-key encryption.

3.2.1 Stream ciphers

Stream ciphers are used for encrypting a continuing stream of data for transmission on a communication channel. In the stream cipher the bit stream of the *Plaintext* is ciphered using a stream of key bits. The output of the cipher depends on the internal state of the cipher algorithm. Therefore the same text string will result in different *Ciphertext* every time it's encrypted. The advantages of this cipher are high speed and low hardware complexity.

3.2.2 One-time-pad ciphers

In general a symmetric key cipher is considered secure if the most effective attack has approximately the same workload as a brute force attack. However, they can be broken since the same key is used multiple times. The one-time-pad ciphers solve this problem and give perfect secrecy to the messages sent. This is done by always encrypting the message using a fresh new random key. For example a four letter message, encrypted using a one-time pad, is impossible for the attacker to decrypt since every possible four letter *Plaintext* could be the true message. The true message is just as likely to be "fast", "kiss" or "stop" from the attacker's viewpoint [5].

3.2.3 Block ciphers

Block ciphers encrypt the *Plaintext* by dividing the data into fixed sized blocks and processing each block at a time. Each block is processed with a block cipher encryption algorithm. The algorithm processes the fixed size *Plaintext* block of length *n*, together with a fixed size key and sometimes along with an initialization vector. The output is a fixed size *Ciphertext* block of the same length *n*. The substitution box, s-box, is commonly used in block ciphers and are therefore briefly described. S-boxes are tables which present the specific substitute that is to be done to encrypt the input.

A simple example is a 3 × 2 S-box table found in table 3.1. Given a 3 bit input, the 2 bit output is found by selecting the row using the outer bit, and the column by using the inner 2 bits.

Table 3.1: S-BOX

		Inner 2 bits			
		00	**01**	**10**	**11**
Outer bit	0	10	11	01	00
	1	11	10	00	01

The table shows a simple example of an S-box. It takes a 3 bit input and substitutes it to 2 bits which is the output.

In this section some block ciphers which are relevant to this thesis will be described.

3.2.3.1 Data encryption standard

Data Encryption Standard, DES is a block cipher and was selected in United States in November 1976 to be a Federal Information Processing Standard (FIPS). DES uses a block size of 64 bits. These days DES is considered to be insecure for many applications. DES key sizes of 56 bit have been broken in less than 24 hours [6]. Therefore not be used for any new application. One attempt to enhance the security was to upgrade the algorithm to 3DES, TDES or TDEA (Triple Data Encryption Algorithm).

3.2.3.2 Advanced encryption standard

The Advanced Encryption Standard, AES, was the new standard replacing DES. The AES was decided by a design-contest were several candidates where contributed. The contest was held by The National Institute of Standards and Technology, NIST, and resulted in the new AES standard on October 2, 2000 [7]. The final five candidates were; Rijndael, Twofish, RC6, Serpent and MARS. The final winner was the Rijndael cipher, invented by Joan Daemen and Vincent Rijmen. Rijndael can be specified by a key and block size of any multiple of 32 bits from 128 bits to 256 bits. However, AES is specified only to operate on a fixed block size of 128 bits using a key size of 128, 192 or 256 bits [8]. The CTR mode was standardized in 2001 [9] although it has been around since the DES came in 1980.

3.2.3.3 Twofish

Twofish, one of the five finalists in the AES-contest, is a block cipher with a block size of 128 bits supporting key sizes up to 256 bits. It is related to the earlier Blowfish. Instead of using fixed tables as S-boxes, values are generated dynamically using information from the key. One half of the key is used in encryption

and the other is used generating the S-boxes. It is slower than Rijndael using a key of 128 bits, although faster when using a 256 bit key.

4. SYMMETRIC ALGORITHMS

The symmetric algorithms tested on the test platform were AES, DES, 3DES and Twofish. The measurements of the different algorithms done using 10 KiB data and will be presented in the unit [*ms*/*KiB*].

4.1 Symmetric Encryption And Decryption

Both encryption and decryption routines were implemented on the board for the measurements on the symmetric algorithms. The test was performed using several key sizes for the algorithms, wherever this was possible. The software algorithm test was based on the cryptographic library Lib Tom Crypt and did not require any mathematical library. The library is configured for size rather than speed wherever possible.

Table 4.1: Symmetric Encryption And Decryption

			LibTomCrypt		Hardware module	
Algorithm	**Block size** *[bits]*	**Key size** *[bits]*	**Encryption** *[bits]*	**Decryption** *[bits]*	**Encryption** *[bits]*	**decryption** *[bits]*
AES	128	128	92	9.2	0.57	0.57
AES	128	256	115	11.5		
DES	64	56	150	15.0	0.55	0.55
3DES	64	168	37.6	37.6	0.88	0.88
Two fish	128	128	9.1	9.1		
Two fish	128	256	9.1	9.1		

The table shows the encryption- and decryption time taken, for the listed symmetric ciphers, to compute 1 KiB.

Table 4.2: Symmetric Algorithm's Different Memory Requirements

Symmetric cipher	**Memory requirements of the algorithm's code** *[Byte]*	**Memory requirements of the algorithm's tables** *[Byte]*	**Total FALASH required** *[Byte]*
AES	7304	4436	11740
DES/3DES	6028	2588	8616
Twofish	9888	392	10280

The table show symmetric algorithm's different memory requirements.

For AES, DES and 3DES it was possible to use the hardware acceleration modules on the processor. This was also tested and the results are listed as a comparison in table 4.1. Note that by using the hardware implementation in all three algorithms, there was a large improvement in speed. The AES cipher was 16.1 times faster than the software implementation, and 3DES was 42.7 times faster.

4.2 Asymmetric Algorithms

In the test of the asymmetric algorithms, RSA was compared to ECC. Due to the large amount of memory required by the asymmetric cryptography there was insufficient memory left to implement a PRNG, which is required to perform encrypt and signature-operations. Therefore the implemented operations were limited to decryption and verification. The two different algorithms were benchmarked with the previously determined key sizes. ECC was also implemented with the two different mathematical libraries, LibTomMath and TomsFastMath. The TomsFastMath library could not be used in combination with RSA due to lack of memory.

4.2.1 Asymmetric decryption

The two following tables, 4.4 and 4.5, shows the different decryption times for ECC and RSA respectively using their private keys. The data used in this test was a symmetric key of 128 bits, encrypted by the corresponding public key. In the first table, describing ECC decryption, both LibTomMath and TomsFastMath libraries are used and compared to each other. The improvement factor of implementing the TomsFastMath library is increasing with the security level. The largest improvement is therefore with the 521 bit key, resulting in an improvement factor of nearly 4 times faster decryption.

Table 4.4: Decryption In Ecc

Key size	**With LibTomMath** *Decryption time*	**With TomFastMath** *Decryption time*	***Improved by factor***
[bits]	[Sec]	[Sec]	
ECC 112	0.808	0.765	1.06
ECC 128	0.980	0.834	1.18
ECC 160	1.233	0.938	1.31
ECC 192	1.570	1.064	1.48
ECC 224	1.972	1.187	1.66
ECC 256	2.829	1.375	2.06
ECC 384	6.329	2.277	2.78
ECC 521	12.940	3.248	3.98

The table shows the decryption time taken for different key sizes in both LibTomMath and TomsFastMath.

CONCLUSIONS

This paper presents the consisted of implementing cryptographic algorithms on an ARM7 based test platform. Here the conclusions from the benchmarks of the cryptographic algorithms will be presented, including a summary of the constraints related to the limited performance of the embedded

system. The aim of this paper was to answer the question of weather it is feasible to implement advanced cryptography on an embedded platform to be used in an access system and if the resulting performance would be usable. The comparison of ECC and RSA in this implementation showed that it is feasible to implement asymmetric cryptography algorithms with very high security, RSA key sizes up to 3076 bits and ECC key sizes up to 521 bits, on an embedded platform. The ECC algorithm outperformed the RSA implementation with a large margin. ECC with a 521 bit key was able to do both decryption and verification in a very reasonable time of 10 seconds. The same operations on RSA with a 3076 bit key required 45.7 seconds, this compares in security to a 256 bit ECC key which only required 3.3 seconds.

REFERENCES

[1] Linus Fredriksson and Martin Gyllensten. Modellering av delvis kända bilder med hjälpav bilder. URL:http://www.divaportal.org/diva/getDocument?urn_nbn_se_liu_diva-67241__fulltext.pdf, 2007.

[2] Securitas AB. Passersystem - generelltom passersystem. URL: http://www.securitassystems.se/97888027-1404-4b55-b007-5c51bb545b26.fodoc, 2007.

[3] Bruce Schneier. Attack trees - modeling security threats. URL:http://www.schneier.com/paper-ttacktrees-ddj-ft.html, 1999.

[4] Bruce Schneier Niels Ferguson. Practical Cryptography. Number ISBN 0-471-22357-3.

[5] David Kahn. The Code-Breakers. Number ISBN 0-684-83130-9.

[6] Keshava P. Subramanya. Brute force searches in cryptography. URL:http://www.cs.ucsb.edu/∼keshava/bruteforce/bruteforce.html, 2007.

[7] National Institute of Standards, Technology, and the agency of Commerce Department´s Technology Administration. Commerce department announces winner of global information security competition. URL: http://www.nist.gov/public_affairs/releases/g00-176.htm, 2007.

[8] Federal Information Processing Standards Publications. Announcing the advanced encryption standard (aes), fips 197. URL:http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf, 2001.

[9] National Institute of Standards and Morris Dworkin Technology. Recommendation for block cipher modes of operation methods and techniques, nist special publication 800-38a. URL: http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf, 2007.