

私立東海大學資訊工程學系研究所

碩士論文

指導教授:林祝興博士

共同指導教授:劉榮春博士

移動裝置上智慧卡執行密碼演算法的耗能分析
研究

Energy Consumption Analysis for Cryptographic
Algorithms on Smart Cards in Mobile Devices

研究生:陳冠翰

中 華 民 國 一 〇 〇 年 六 月

東海大學碩士學位論文考試審定書

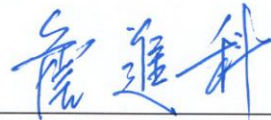
東海大學資訊工程學系 研究所

研究生 陳冠翰 所提之論文

移動裝置上智慧卡執行密碼演算法的耗能分析
研究

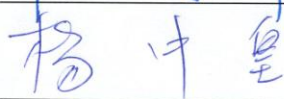
經本委員會審查，符合碩士學位論文標準。

學位考試委員會
召集人

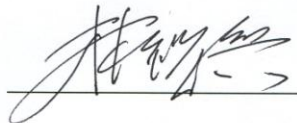


簽章

委員



指導教授



簽章

中華民國 100 年 6 月 28 日

Abstract

In mobile devices, smart cards become the trend and are used in numerous applications and services. Since mobile devices are battery-powered, it is important to find energy saving solutions to reduce energy consumption. Programmers have to handle the power consumption issues when they develop applications. Previously, contact smart cards are connected to a terminal or a computer, which provide sufficient power and so applications developed at that time may not fully consider the power consumption issue. However, when these application programs are ported from computers to battery-powered mobile appliances, the following three criteria, i.e. performance, security, and power consumption should be handled with care.

This thesis studies a smart card embedded in a security chip with. In this study, we use LabView 2010 and related hardware to build a power measurement environment. And we use cryptographic algorithms of OpenSSL project to measurement. Two sets of experiments are conducted in this study. In fixed clock experiment, we analyze the effect in the energy consumption and the execution time due to the clock change. It shows that the algorithm complexity affect the power saving and increase execution time, meanwhile, reducing the clock rate has more impact to the execution time than the energy consumption. In random clock experiment, we use a random clock to focus energy consumption distribution of different key. Therefore, it increases the number samples that the attacker need and reduce the success rate of side channel attacks and increase the security of smart cards. Programmers can use results of this thesis as reference when developing applications.

Keywords: Cryptographic, LabView, Smart card, CPU clock, Energy consumption.

摘要

在移動裝置上需要智慧卡的應用軟體和服務愈來愈多。因為移動裝置使用電池電力，電力消耗的程度很重要。程式設計師開發軟體時必須考慮裝置的電力消耗。過去接觸式智慧卡是連接在終端機或是電腦上使用，可以獲得充足電力，電力消耗並不是開發應用軟體時必須考慮的問題。然而，當我們由電腦移植軟體到移動設備上，就必須考慮三個問題-效能、安全性和能量消耗。

這篇論文研究使用可作為智慧卡的安全晶片並在晶片上實作多項的密碼演算法。使用 LabView 2010 和相關的硬體建立能量測量環境。使用 OpenSSL 專案中的密碼驗算法做測量。本論文有兩個實驗。在固定時脈實驗，分析時脈改變時對能量消耗和執行時間的影響。一方面發現演算法的複雜度會影響降低時脈的節能效果和拉長執行時間;另一方面發現降低時脈對執行時間的影響比能量消耗大。在隨機時脈實驗，利用隨機時脈使不同金鑰的耗電分布集中，增加旁道攻擊的所需的樣本數，降低旁道攻擊的成功率，增加智慧卡的安全性。程式設計師可以觀看本文的結果作為開發時的參考。

關鍵字: 密碼學, LabView, 智慧卡, 處理器時脈, 能量消耗.

Contents

Abstract.....	I
摘要.....	II
Figure List.....	IV
Table List	V
1. Introduction.....	1
2. Background.....	4
2.1 Symmetric Encryption.....	4
2.2 Asymmetric Encryption	5
2.3 Hash algorithms.....	7
3. Side channel attacks.....	8
3.1 Timing analysis.....	8
3.2 Simple power analysis	8
4. Experimental environment.....	10
4.1 Hardware and software	10
4.2 Experimental environment setup.....	11
4.3 Energy consumption measurement steps	11
5. Experimental results.....	14
5.1 Energy analysis of cryptographic algorithms using different clock rates	14
5.1.1 Symmetric algorithm-AES.....	14
5.1.1 Asymmetric algorithm-RSA	21
5.1.2 Hash algorithms-MD5, SHA1, and SHA256.....	24
5.2 Using random clock against SPA.....	28
6. Conclusions.....	31
Bibliography	33

Figure List

Figure 1 SSDC architecture at the top of SD Slot, mobile devices at the bottom of SD Slot...	2
Figure 2 The application of SSDC in VoIP.	3
Figure 3 Experimental environment.....	11
Figure 4 ECB mode to reduce the clock and increase the rate of execution time.	17
Figure 5 CBC mode to reduce the clock and increase the rate of execution time.....	17
Figure 6 OFB mode to reduce the clock and increase the rate of execution time.	18
Figure 7 CFB mode to reduce the clock and increase the rate of execution time.	18
Figure 8 ECB mode to reduce clock and reduce the rate of power consumption.	19
Figure 9 CBC mode to reduce clock and reduce the rate of power consumption.	19
Figure 10 OFB mode to reduce clock and reduce the rate of power consumption.	20
Figure 11 CFB mode to reduce clock and reduce the rate of power consumption.	20
Figure 12 Shown to the impact of reducing clock to execution time.....	22
Figure 13 Shown to the impact of reducing clock to power consumption.....	23
Figure 14 Shown to the impact of reducing clock to execution time.....	26
Figure 15 Shown to the impact of reducing clock to power consumption.....	27
Figure 16 Power consumption distribution of RSA uses 25 MHz.....	29
Figure 17 Power consumption distribution of RSA uses 25/12.5 MHz.....	29
Figure 18 Power consumption distribution of RSA uses 12.5 MHz.....	30

Table List

Table 1 Experiment parameters	13
Table 2 Power consumptions and execution times of AES using 25MHz.....	15
Table 3 Power consumptions and execution times of AES using 12.5 MHz.....	15
Table 4 Power consumptions and execution times of RSA	22
Table 5 Power consumption and execution time of hash algorithms using 25 MHz	26
Table 6 Energy consumption and execution time of hash algorithms using 12.5 MHz.....	26

1. Introduction

In the past, financial transactions are usually carried out with a personal computer. With the fast development of the 3G network and mobile devices in recent years, applications such as login network banking, online shopping, and financial transactions only used on a personal computer in the past can be ported in mobile devices now [1][2][3][8]. Since e-commerce activities in an insecure network has the risk of leaking information, therefore, service providers often use smart cards to protect consumer information. Smart cards can be used as tokens for identity authentication and can be used to encrypt transferred data. To avoid being stolen by Trojans viruses, a secret key can be stored in the card and the encryption and decryption processes are executed completely inside the card the secret key. On the other hand, people often use mobile devices to watch TV or listen to online music. If the service provider's software on mobile devices is cracked, people can enjoy services without paying, which will hurt the rights and interests of service providers. In order to protect their interests, service providers can encrypt meaningful information through the smart card.

How dose a smart card work on mobile devices? At practice, the smart card reader can access the smart card through USB. But this way is considerably troublesome, since people must carry the reader along. Another approach is to access smart card through the microSD slot on mobile devices, as shown in Figure 1 [15]. In Figure 1, the microSD card, known as the Security microSD Card (SSDC), consists of a smart chip and flash memory, with both storage and security services functions [15]. By SSDC, mobile devices use smart card services without the need of extra external devices. By SSDC, programmer can develop many applications. Figure 2 shows one application of SSDC. Designers can encrypt VoIP packets through the SSDC. Except the intended receiver, other people cannot recover the true voice. The call cannot be tapped by outsiders and the user's privacy is protected.

Most mobile devices are powered by batteries with limited power capacity. In order to protect information during the communication process, security system designers use many cryptographic algorithms to protect user data. The encryption technology will consume considerable power. Without suitable limitation, excess use of cryptographic algorithms will exhaust the power very soon. Cryptographic algorithms according to different security levels consume different amount of power, for example, asymmetric encryption methods consume more power than symmetric encryption methods.

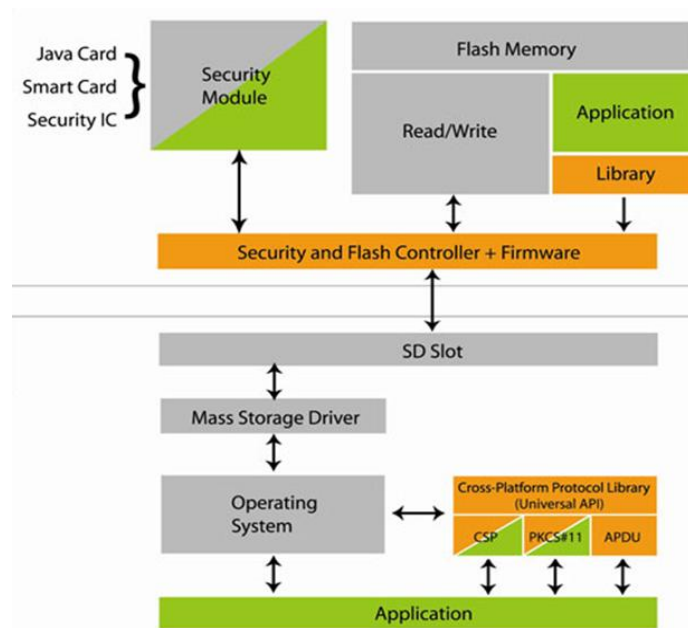


Figure 1 The Security microSD Card (SSDC). From top to bottom: SSDC architecture, SD Slot, and mobile devices.

On mobile devices running the security protocols, the ratio of cryptographic algorithms in total power consumption is not the largest, but instead, the environment power consumption is the most important [11]. Therefore, to measure the power consumption of cryptographic algorithms running on a smart card, we need to add the power consumption of the card environment to that of cryptographic algorithms.

When smart cards are in use, some useful information is leaked, such as the power consumption, and execution time and easily obtained by attackers. Using timing attack or simple power analysis, the attacker can execute side channel attacks and acquire the secret key in the card [9][12][14]. Kocher et al. in 1998 describe differential power analysis (DPA) [12]. The attack success rate is higher, the threat to the smart card is greater.

The rest of this thesis is organized as follows: Chapter 2 describes the background knowledge of the cryptographic algorithm. Chapter 3 describes the side channel attacks include time analysis, simple power analysis, and differential power analysis. Chapter 4 describes the experimental environment settings, and Chapter 5 presents experimental results. Chapter 6 concludes this study.

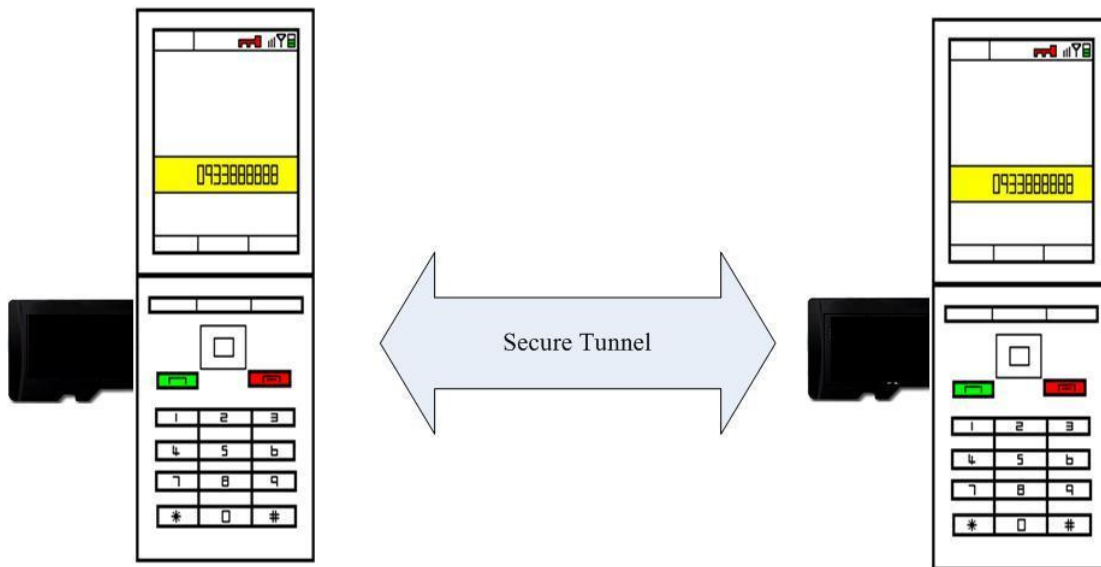


Figure 2 The application of SSDC in VoIP.

2. Background

This section will describe the symmetric, asymmetric, and hash algorithm used in this thesis.

2.1 Symmetric Encryption

Before the advent of the asymmetric encryption method, the symmetric encryption method is the only encryption method. Compared to the asymmetric encryption, the symmetric encryption has faster encryption and decryption processes, shorter key length, and lower computational cost. Currently the symmetric encryption is used more than the asymmetric encryption.

One feature of the symmetric encryption and decryption method [4] is to use of the same key. The communicating parties must use the same key to correctly transmit messages. The symmetric encryption method can be divided into two types, i.e. stream and block cipher encryption. The stream encryption, such as RC4, selects a byte or a bit to encrypt each time, and so it encrypts data instantly without the need to wait until the encryption block is filled with data. On the other hand, the block cipher encryption, such as DES [18], 3DES, AES [17], has to wait until the block is filled with data, and then starts to do encryption and decryption. For communications, a pre-shared key or agreement key is used to encrypt data. The same pre-shared key is used for encryption all the time if no other mechanism is used to change it. Thus, the session key is cracked easily and the communication has a high risk of eavesdropping. By using a key agreement, a key is randomly generated. Although the encryption key for each session is different, without careful analysis of the key agreement, the attacker may find a loophole in the existing agreement and be able to get the key to tap messages. Since symmetric encryption algorithms have security problems, the asymmetric encryption algorithms are used to improve them. However, considering

the encryption and decryption speed and low resource requirements, the symmetric encryption algorithms are still in use today.

AES by NIST replaces DES and has become the most widely used encryption algorithm. The inventor of the AES, Rijndael, submitted the original version, which can designate their respective block length and key length of 128,192 or 256 bits to have nine combinations. But the final version of the AES key length has the three choices, but the block length is fixed as 128 bits.

According to the key length, AES is executed 10, 12, or 14 rounds. Each round is generally divided into four steps: SubBytes, ShiftRows, MixColumns, and AddRoundKey [18]. However, not each round has the four steps. AES can be performed in 8 bit, 16-bit, 32-bit and 64-bit computing systems. Because of the parallel design of AES, when the key length becomes long and the number of rounds increase, it still has high performance and efficient use of resources. AES performs best in a 32-bit machine. Using the 32-bit word length as a unit, part results of each AES round can be made as tables. For encryption and decryption operations, each time only a small number of operations plus some table look-ups are needed, so the computation time is reduced a lot.

2.2 Asymmetric Encryption

The asymmetric encryption algorithm, or the public key encryption method, uses different keys, namely the public key and private key for encryption and decryption [16] [19]. There are problems by using the symmetric encryption such as the key sharing. In order to ensure the key be known only by the two communication parties, symmetric key encryption must use a secure protocol to send the key. Public key cryptography solves the key distribution problem by using two paired keys, one used to encrypt and the other used to decrypt. The public key cryptographic algorithm has

characteristics that it is not feasible find the private key when the public key is only known.

Most public key cryptographic theories are based on the number theory. Calculation unsolvable problems are based on the number theory, such as the integer factorization and discrete logarithm problems. To crack public key cryptographic algorithms, in addition to find hidden loopholes, the most direct way is to solve difficult number theory problems to obtain the key. Public key algorithm uses irreversible mathematical functions, for which the computational cost grows nonlinearly with the key length. It is almost impossible to crack the key in an effective time, so the public key of sufficient length is almost unbreakable.

The public key scheme not only can be used to encrypt confidential information, but also can be used for identity authentication and non-repudiation functions [13]. The identity of the receiver can be determined, because the only the legitimate recipient holds the unique private key. In the same way, the private key scheme can be used to do data signatures with non-repudiation [7] [13].

RSA[19] is an asymmetric encryption method and widely used in security systems. RSA encryption algorithm is easy to understand. The difficulty lies in the key pair generation. To generate RSA key, two large prime numbers, p and q are chosen first to obtain $n = pq$. Then choose an integer e that is primed with n . The third step is to obtain the private key by computing d . Users will publish (e, n) as a public key and retain d as a private key. And d also can be used to sign data to ensure the legitimate sources of information.

2.3 Hash algorithms

Hash algorithm [28][29][30] can output input messages of any length to a fixed length hash code. A small change of the input message, such as 1 bit or 1 byte, will change the output of the hash algorithm. Thus, hash algorithms are often used to test whether a message has been tampered with. Common hash algorithms are MD5, SHA1 and SHA256. For MD5, a message of arbitrary length can be entered to output 128-bit message digest. During the hash code generation process, the input message is divided into several 512 bits blocks. The whole process is divided into four steps. First, add additional bits in the message, making the remainder of the total length divided by 512 equal to 448, which is used in the second step. Next, use 64 bit to represent the original length of the message and append it behind the added additional bits and make the message length become a multiple of 512. Third, setup four 32-bit register and fill in their initial values. Those registers will be used to store middle values. Finally, all of the 512-bit message blocks will put through the MD5 compression module, and output a 128-bit message digest.

SHA1 has similar structures with MD5. SHA1 output a 160 bits message digest and is more secure than MD5. SHA256 structure and internal processes are very similar to the SHA1; SHA256's message digest length is 256 bits.

3. Side channel attacks

To obtain the data in the smart card, the attacker can use invasive attacks to directly access the data in the smart card. But the smart card chips can use security check to resist invasive attacks. The other method, known as side channel attacks, does not directly try to read the card information, but rather to observe the behavior of the card and analyze it. By observing the operation of the smart card to find its characteristics, one has the opportunity to get more information. Side channel attacks [10] allow the attacker to observe and find relevant information of smart cards, analyze, and then get the desired information.

Here are two examples of side channel attacks:

3.1 Timing attack

An attacker can use different parameters and observe the execution time to collect enough specimens to have the opportunity to infer the secret key [14]. To prevent this attack, one can execute each instruction with the same execution time to increase the difficulty for an attacker to observe information and to decrease the attack success rate.

3.2 Simple power analysis

Simple power analysis (SPA) approach is to collect and observe the power consumption information of microprocessors implemented with cryptographic algorithms [9][10][12]. Because each processor instruction set in the implementation consumes different power, the attacker who have the opportunity to collect data to infer the key [12]. For example, the implementation of one cryptographic algorithm is repeated multiplication, addition and shift operations. These operations will consume different power consumption on the microprocessor and the attacker can collect these

informations to infer the key. To prevent SPA algorithm, one can improve the operation flow to increase the difficulty of observation, although it may cause performance loss.

4. Experimental environment

In this section we show the experimental setup and describe the hardware and software used in the experiment. And in this section we also explain the method used to measure energy consumption.

4.1 Hardware and software

We used National Instruments LabVIEW (NI LabVIEW) 2010 software and related hardware to build the power measurement system. Through graphical lines and diagrams, NI LabVIEW is used to establish flow charts to develop a complete measurement, testing, and control system. LabVIEW [26] integrate thousands of hardware devices and built libraries to work for high-level analysis and present information, provide a strong virtual instrumentation functions. LabVIEW platform can work on variety of operating systems.

The security chip used in the experiment is ST33F1M [25] produced by STMicroelectronics. ST33F1M include SecurCore® SC300 CPU Core designed by ARM, and the core is 32-bit RISC core based on Cortex^l M3 core with high performance. The CPU clock is up to 25 MHz and the chip has 30 Kbytes user RAM and 1280 Kbytes user flash memory. The flash memory has 10-year data retention, 100,000 erase/write cycles. The ST33F1M can be used in mobile communications, multimedia, and banking. The PC used in this experiment has a 2.60GHz Intel Pentium Dual-Core CPU E5300, and 4 GB of RAM, and running the Windows XP OS. The security cryptographic algorithms are provided from OpenSSL 0.9.8.o [20] project (The latest version is 1.0.0.d).

4.2 Experimental environment setup

As shown in Figure 3, the Vcc pin of SSDC is used to connect a power supply. A 2 ohm resistor is in series with the 3 voltage power supply [21][22][23]. BNC-2110 [26] measures the voltage drop across the resistor and connects to a data acquisition card [26]. LabView on the PC performs data analysis and statistics.

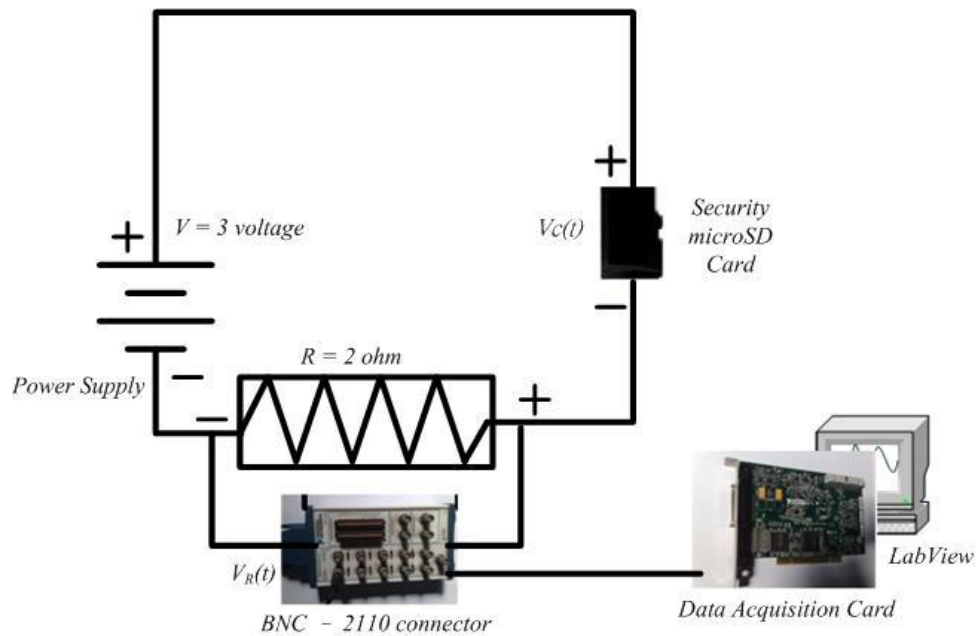


Figure 3 Experimental environment

4.3 Energy consumption measurement steps

In the experiment, the voltage drop across the resistor is measured [21][22], which can be used to find the voltage over the security chip and to calculate its energy consumption.

Table 1 lists the experiment parameters. First, run cryptographic algorithms on the security chip and obtain n samples of voltage drops across the resistor, $V_R(t)$. From the Kirchhoff voltage law, the voltage over the SSDC is calculated as:

$$V_c(t) = V - V_R(t) \quad (1)$$

By the Ohm's law, the current through the SSDC is:

$$I(t) = \frac{V_R(t)}{R} \quad (2)$$

The instantaneous energy consumptions on the SSDC are:

$$J(t) = I(t) * V_c(t) \quad (3)$$

We are summed up by sampling frequency to find the total energy consumption J .

Table 1 Experiment parameters

Notation	Description
V	Input voltage.
$V_R(t)$	The t-th sampling point, the voltage across the resistor.
$V_C(t)$	The t-th sampling point, voltage on security chip, $V_C(t) = V - V_R(t)$
$I(t)$	The t-th sampling point, the ampere across the resistor, $I(t) = \frac{V_R(t)}{R}$
n	The total number of sampling points.
R	Resistor, in this thesis is 2 ohms.
$J(t)$	The t-th sampling point, security chip energy consumption, $J(t) = I(t) * V_C(t)$
J	Total energy consumption, $\frac{\sum_{t=1}^n J(t)}{rate}$, rate is sampling rates that 250 KHz. On the other hand, each second has 250000 sampling point.

5. Experimental results

The cryptographic algorithm is executed in the experimental platform and actual data are retrieved to find the effect on the execution time and energy consumption by reducing the clock rate in section 5.1. Furthermore, in section 5.2, we randomly change the processor clock to reduce the energy consumption variations due to different keys used in RSA in order to decrease the success rate of SPA.

5.1 Energy analysis of cryptographic algorithms using different clock rates

Two CPU clock rates: 25 MHz and 12.5 MHz are used to run the cryptographic algorithm to find their effect on the energy consumption and execution time.

In the AES phase, three key lengths and four operation modes are used. Each key length generates 100 random keys, which are executed 100 times in the four modes and the average results of the execution time (in ms) and energy consumption (in μJ) of each byte are recorded. Encrypted data is a binary file of 31124 bytes, and its content is the U.S. Declaration of Independence [27]. In the RSA phase, two key lengths of 1024 bites and 2048 bites are used, and the public key is $e = 65537$. In order to see significant energy consumption change by using signature when the key length is changed, we use the private key. In each key length 100 different keys are generated and each of them is executed 100 times. The encrypted 128 bytes data are values of $0x00 \sim 0xff$. In the hash algorithm phase, MD5 [28], SHA1 [29], SHA256 [30] are used and the encrypted data are the same as AES.

5.1.1 Symmetric algorithm-AES

Table 2 shows the execution time and energy consumption for AES by using the 25 MHz clock rate in different key lengths and four operation modes. Table 3 shows

the execution time and energy consumption for AES using the 12.5 MHz clock in different key lengths and the four operation modes. Among the four modes, the ECB mode does not use an initial vector, so its energy consumption is the lowest. In three key lengths, the AES-256 has most computation time and energy consumption.

Table 2 Energy consumptions and execution times of AES using 25MHz

25 MHz								
Mode	ECB		CBC		OFB		CFB	
Length	T(ms)	E (μJ)	T(ms)	E (μJ)	T(ms)	E (μJ)	T(ms)	E(μJ)
AES-128	485	0.449	515	0.495	516	0.471	516	0.471
AES-192	531	0.500	578	0.526	546	0.508	547	0.520
AES-256	578	0.527	609	0.548	594	0.547	594	0.548

Table 3 Energy consumptions and execution times of AES using 12.5 MHz

12.5 MHz								
Mode	ECB		CBC		OFB		CFB	
Length	T(ms)	E (μJ)	T(ms)	E (μJ)	T(ms)	E (μJ)	T(ms)	E (μJ)
AES128	641	0.384	719	0.421	687	0.405	703	0.406
AES192	734	0.421	781	0.457	765	0.442	781	0.452
AES256	781	0.464	859	0.503	844	0.489	843	0.489

Figure 4 shows the result in the ECB mode by reducing the clock rate, in which the execution time is increased. Due to different key lengths the execution time are increased by 32.16%, 33.52%, and 35.12%. Figure 5 shows the result in the CBC mode by reducing the clock rate. Due to different key lengths the execution time are increased by 39.61%, 39.96%, and 41.05%. Figure 6 shows the result in the ECB mode by reducing the clock rate. Due to different key lengths the execution time are

increased by 33.14%, 36.45%, and 42.09% . Figure 7 shows the result in the CFB mode by reducing the clock rate. Due to different key lengths the execution time are increased by 36.24%, 37.29%, and 41.92% . From Figure 4 to Figure 7 we observe that when the clock rate is reduced the execution time is increased faster when the needed amount of computations rises.

Figure 8 shows the result in the ECB mode by reducing the clock rate, in which the energy consumption is reduced. Due to different key lengths the energy consumption are reduced by 14.48%, 12.84%, and 11.95%. Figure 9 shows the result in the CBC mode by reducing the clock rate. Due to different key lengths the energy consumption are reduced by 14.95%, 13.12%, and 8.21%. Figure 10 shows results in the OFB mode by reducing the clock rate. Due to different key lengths the energy consumption are reduced by 14.01%, 12.99%, and 10.60%. Figure 11 shows the result in the CFB mode by reducing the clock rate. Due to different key lengths the energy consumption are reduced by 13.80%, 13.08%, 10.77%. From Figure 8 to Figure 11 we observe that by reducing the clock rate the energy consumption will reduce slower when the needed amount of computations rises.

From Figure 4 to Figure 11 we also observe that by reducing the clock rate in the four modes of AES, the increase of the execution time is faster than the decrease of the energy consumption. For example, AES-256 in the CBC mode with the reduced clock rate increases the execution time by 41.05% and decrease the energy consumption only by 8.21%.

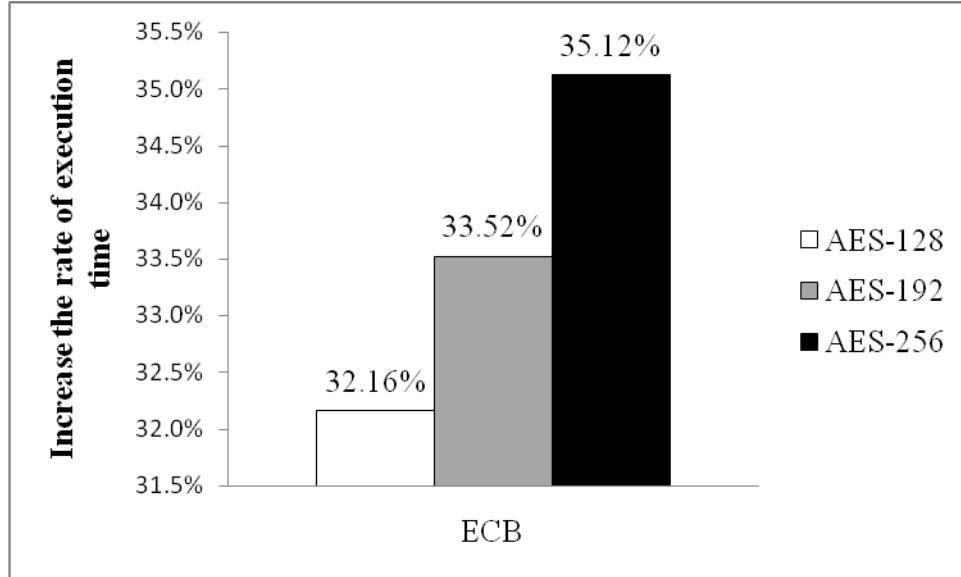


Figure 4 ECB mode to reduce the clock and increase the rate of execution time.

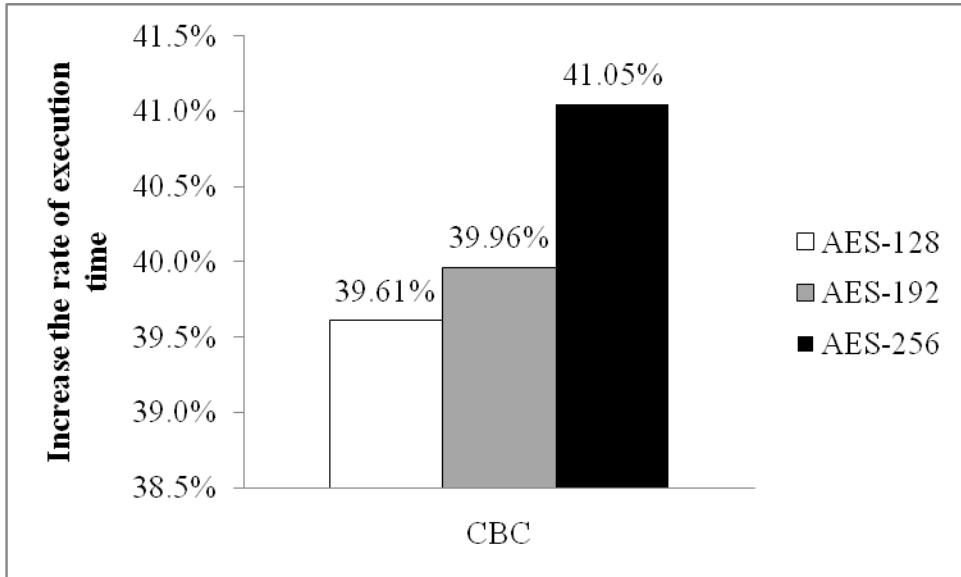


Figure 5 CBC mode to reduce the clock and increase the rate of execution time.

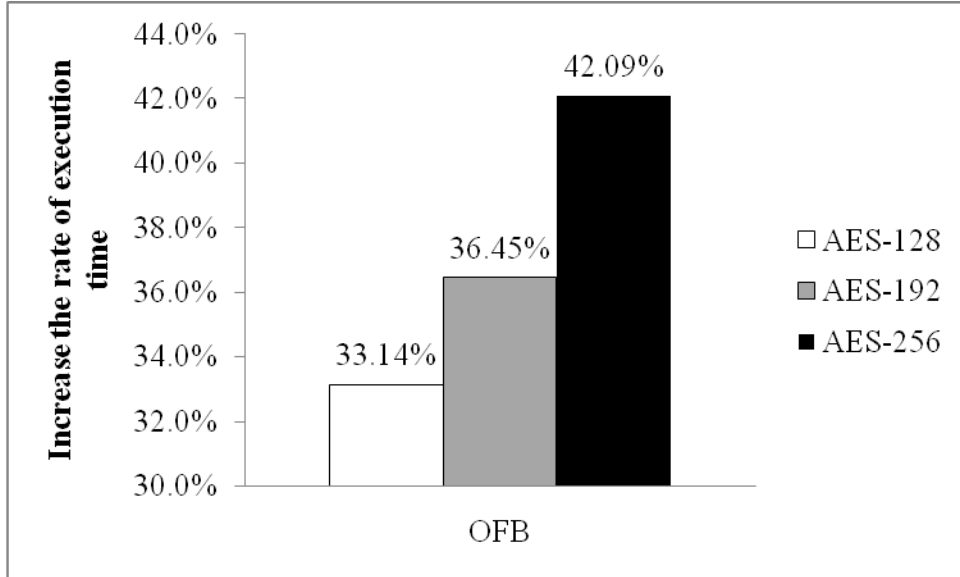


Figure 6 OFB mode to reduce the clock and increase the rate of execution time.

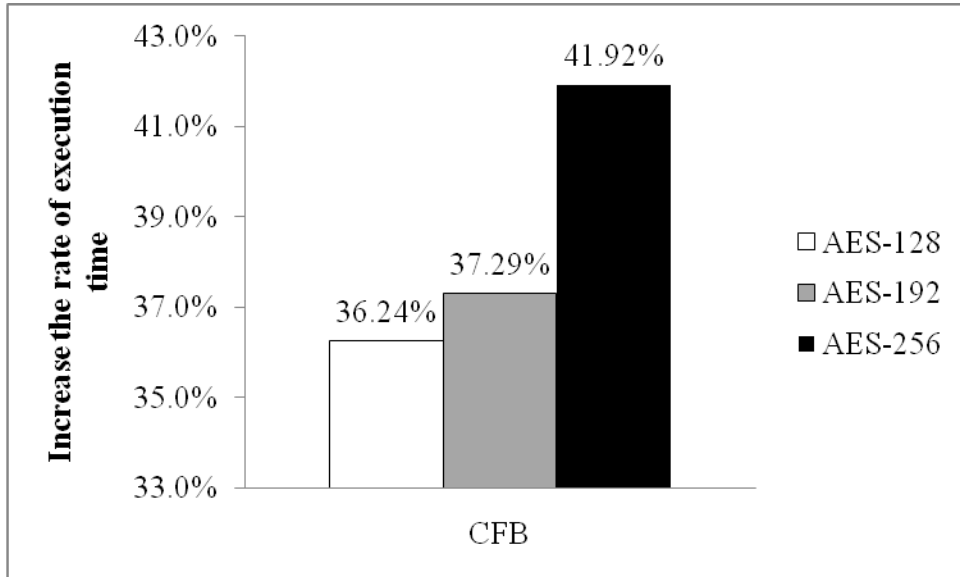


Figure 7 CFB mode to reduce the clock and increase the rate of execution time.

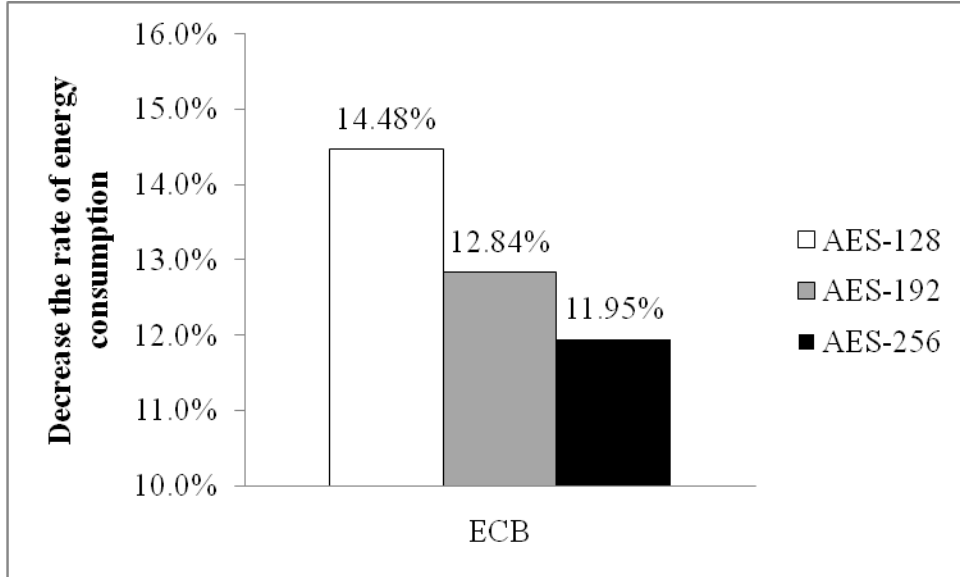


Figure 8 ECB mode to reduce clock and reduce the rate of energy consumption.

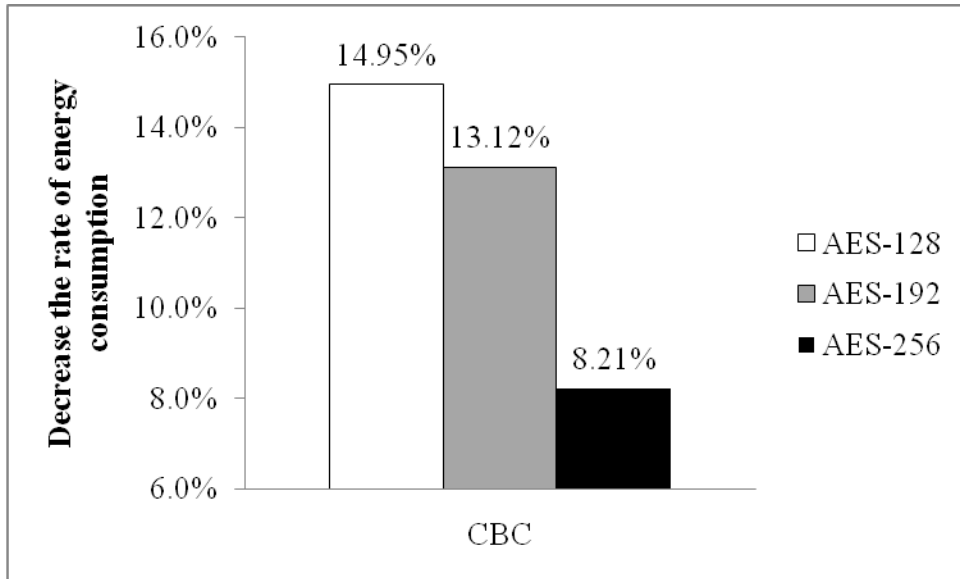


Figure 9 CBC mode to reduce clock and reduce the rate of energy consumption.

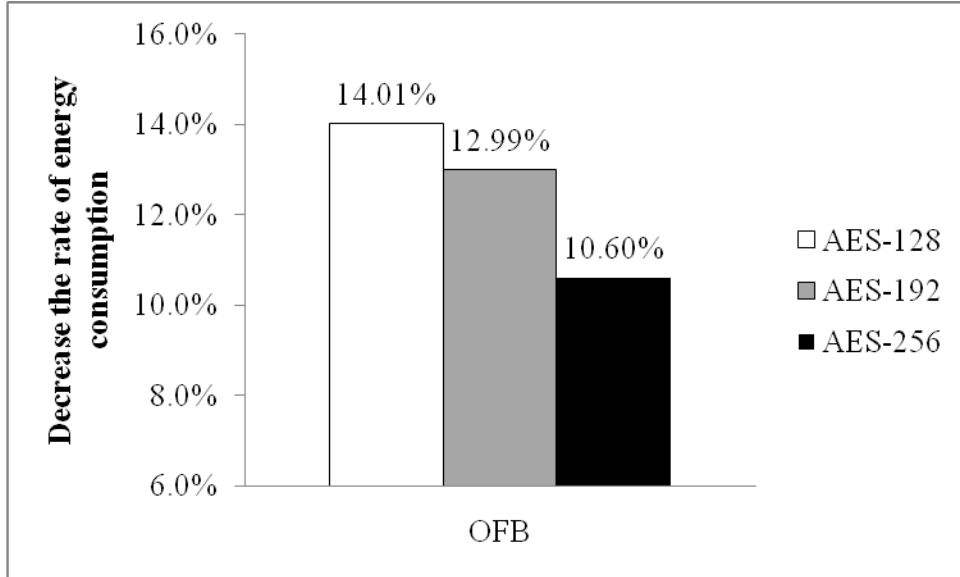


Figure 10 OFB mode to reduce clock and reduce the rate of energy consumption.

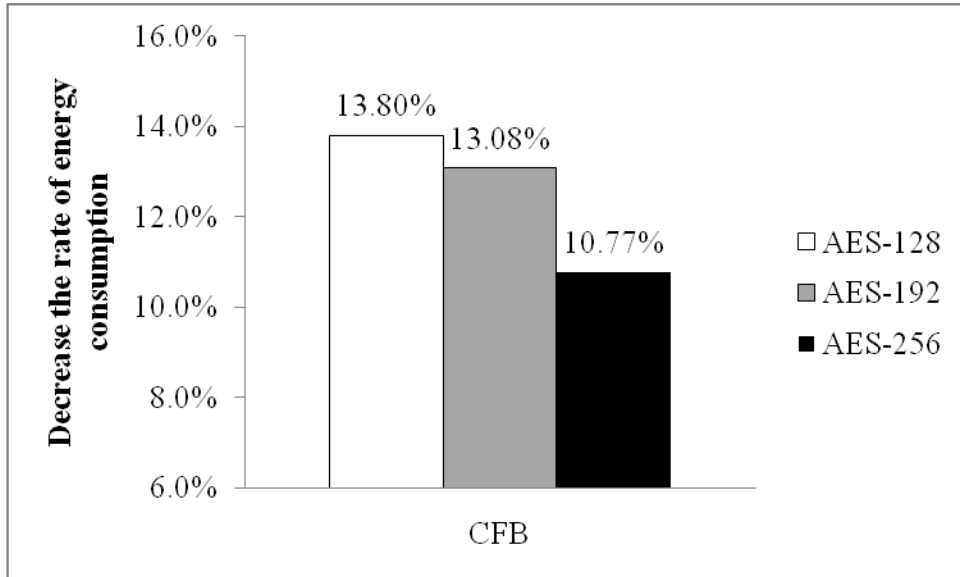


Figure 11 CFB mode to reduce clock and reduce the rate of energy consumption.

5.1.1 Asymmetric algorithm-RSA

Table 4 shows the execution time and energy consumption of RSA with signature operations. RSA-2048 needs more computation than RSA-1024. Using 1024 bits at 25 MHz, the execution time is 78 ms, and the energy consumption is 3.205 mJ. Using 1024 bits at 12.5 MHz, the execution time is 110 ms, and the energy consumption is 1.913 mJ. Using 2048 bits at 25 MHz, the execution time is 324 ms, and the energy consumption is 20.681 mJ. Using 2048 bits at 12.5 MHz, the execution time is 578 ms, and the energy consumption is 13.242 mJ.

Figure 12 shows the effect on the execution time by reducing the clock rate for the two key lengths cases. The execution time of 1024 bits is increased by 40% and the execution time of 2048 bits is increased by 70%. It shows that by reducing the clock rate the execution times increase faster when the needed amount of computations rises.

Figure 13 shows the effect on the energy consumption by reducing the clock rate for the key lengths cases. The energy consumption of 1024 bits is reduced by 41% and the energy consumption of 2048 bits is reduced by 36%. It shows that by reducing the clock rate the energy consumption is reduced slower when the needed amount of computations rises.

In Figure 12 for RSA-1024, by reducing the clock rate, the increase of the execution time is 41.03% and the decrease of the energy consumption is 40.31% ; while in Figure 13 for RSA-2048, the increase of the execution time is 78.40% and the decrease of the energy consumption is 35.97%. We conclude that the effect by reducing the clock rate has more impact on the execution time than the energy consumption.

Table 4 Energy consumptions and execution times of RSA

Clock Length	25MHz		12.5MHz	
	T(ms)	E (mJ)	T(ms)	E (mJ)
RSA-1024	78	3.205	110	1.913
RSA-2048	324	20.681	578	13.242

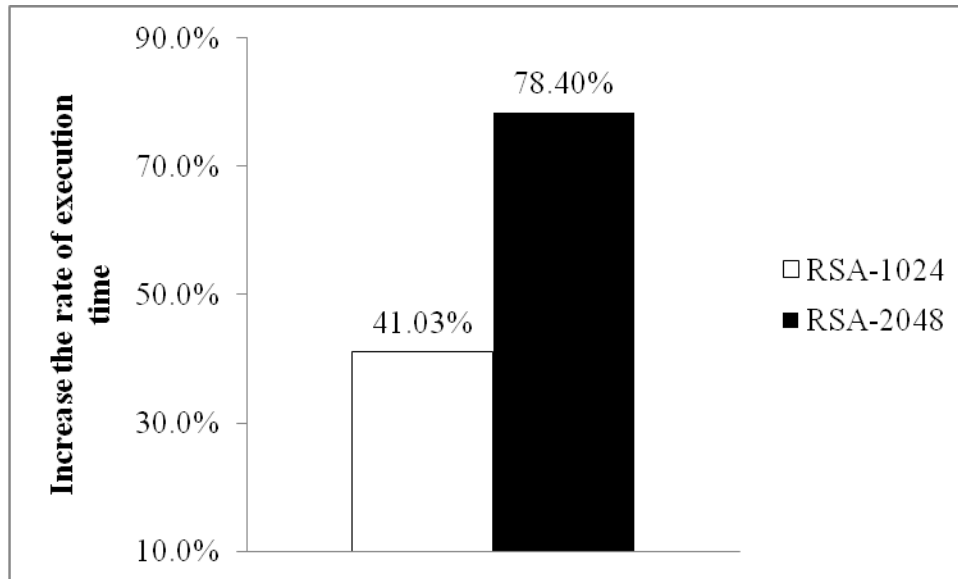


Figure 12 Shown to the impact of reducing clock to execution time

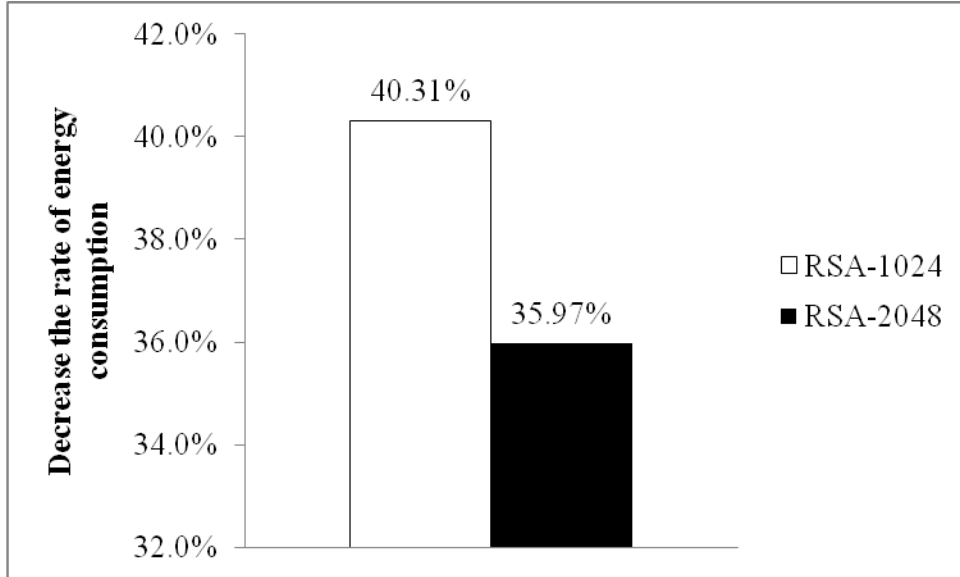


Figure 13 Shown to the impact of reducing clock to energy consumption

5.1.2 Hash algorithms-MD5, SHA1, and SHA256

Figure 15 shows the impact on the energy consumption by reducing the clock rate: for MD5 the energy consumption is reduced by 24.59%; for SHA1, reduced by 17.12%; and for SHA256, reduced by 8.88%. It also shows that by reducing the clock rate, the energy consumption is decreased slower when the needed amount of computations rises.

Figure 14 shows the impact on the execution time by reducing the clock rate: for MD5 the execution time is increased by 26.79%; for SHA1, increased by 29.07%; for SHA256, increased by 40.60%. It also shows that by reducing the clock rate, the execution time is increased faster when the needed amount of computations rises.

In Figure 15 for MD5, by reducing the clock rate, the execution time is increased by 26.79% and the energy consumption is decreased by 24.59%; while In Figure 14 for SHA 256, the execution time is increased by 40.60% and the energy consumption is decreased by 8.88%. We conclude that by reducing the clock rate, the increase of the execution time is faster than the decrease of the energy consumption.

Table 5 shows the execution time and energy consumption using the 25 MHz clock rate to execute hash algorithms. SHA256 needs the most computation. The execution time of MD5 is 321 ms, and the energy consumption of each input byte is 0.305 μJ . The execution time of SHA1 is 375 ms, and the energy consumption of each input byte is 0.333 μJ . The execution time of SHA256 is 500 ms, and the energy consumption of each input byte is 0.439 μJ .

Table 6 shows the execution time and energy consumption using the 12.5 MHz clock rate to execute hash algorithms. The execution time of MD5 is 407 ms, and the energy consumption of each input byte is 0.230 μJ . The execution time of SHA1 is 484 ms, and the energy consumption of each input byte is 0.276 μJ . The execution time of SHA256 is 703 ms, and the energy consumption of each input byte 0.400 μJ .

Figure 15 shows the impact on the energy consumption by reducing the clock rate: for MD5 the energy consumption is reduced by 24.59%; for SHA1, reduced by 17.12%; and for SHA256, reduced by 8.88%. It also shows that by reducing the clock rate, the energy consumption is decreased slower when the needed amount of computations rises.

Figure 14 shows the impact on the execution time by reducing the clock rate: for MD5 the execution time is increased by 26.79%; for SHA1, increased by 29.07%; for SHA256, increased by 40.60%. It also shows that by reducing the clock rate, the execution time is increased faster when the needed amount of computations rises.

In Figure 15 for MD5, by reducing the clock rate, the execution time is increased by 26.79% and the energy consumption is decreased by 24.59%; while In Figure 14 for SHA 256, the execution time is increased by 40.60% and the energy consumption is decreased by 8.88%. We conclude that by reducing the clock rate, the increase of the execution time is faster than the decrease of the energy consumption.

Table 5 Energy consumption and execution time of hash algorithms using 25 MHz

25 MHz		
Algorithm	T (ms)	E(μJ)
MD5	321	0.305
SHA1	375	0.333
SHA256	500	0.439

Table 6 Energy consumption and execution time of hash algorithms using 12.5 MHz

12.5 MHz		
Algorithm	T(ms)	E(μJ)
MD5	407	0.230
SHA1	484	0.276
SHA256	703	0.400

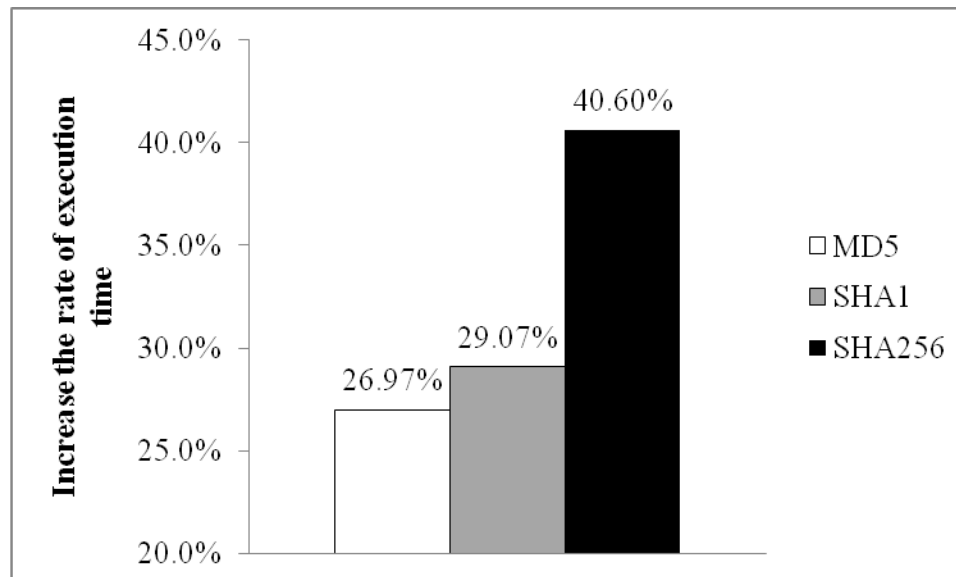


Figure 14 Shown to the impact of reducing clock to execution time

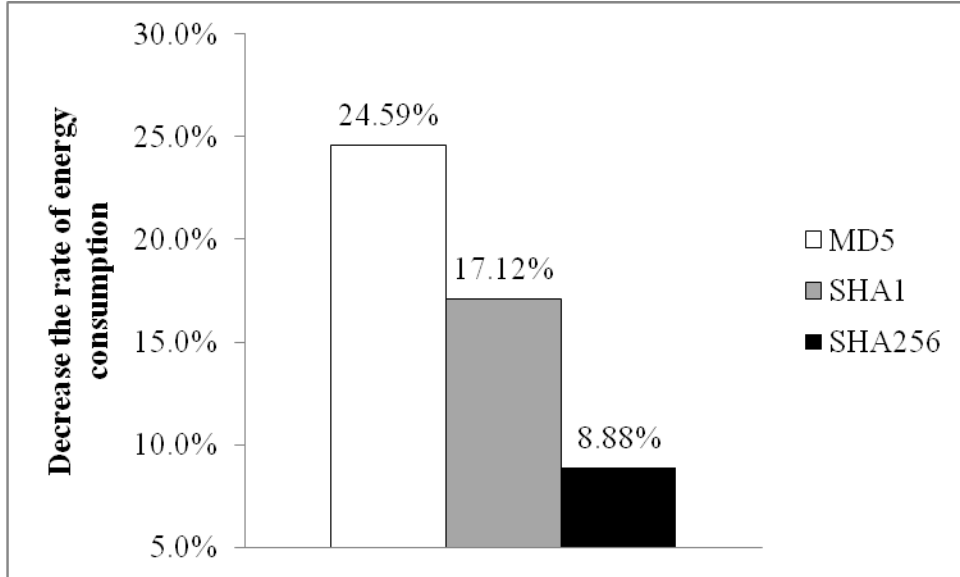


Figure 15 Shown to the impact of reducing clock to energy consumption

5.2 Using random clock against SPA

As mentioned in the Chapter 3, SPA is used to observe the changes of energy consumption to infer the secret key. To prevent SPA attackers from obtaining the secret key, one can make it hard to observe variations of the energy consumption when executing cryptographic algorithms on the device. If the energy consumptions of two different secret keys used to encrypt same data are similar, the attacker will not be able to distinguish the two keys easily.

In this experiment we use a random clock to have similar energy consumption distribution of cryptographic algorithms using different secret key to encrypt same data. Results of the energy consumption distribution by using the random clock will be compared to the fixed clock to show that the random clock is helpful against the SPA attack.

In the experiment we used two kinds of fixed clock, i.e. 25MHz, and 12.5MHz, and one random clock, 25/12.5 MHz, in which 25 MHz or 12.5 MHz is randomly selected in each time. The energy consumption of the random clock case is between two fixed clock cases. In the experiment we used an RSA algorithm with key length of 2048 bits. We randomly generated 1000 keys, and executed RSA 1000 times.

Figure 16 shows the case of using 25 MHz clock, the average energy consumption is 20.921 mJ. Figure 17 shows the case of using 25/12.5 MHz clock randomly, the average energy consumption is 16.469 mJ. Figure 18 shows the case of using 12.5 MHz clock, the average energy consumption is 13.171 mJ. It can be seen that the energy consumption distribution of Figure 17 is in-between of Figure 16 and Figure 18. In Figure 17, the percentage of the most occurrence is 58%, it is more than the results which are 40% and 38.8%, respectively in Figure 16 and 18.

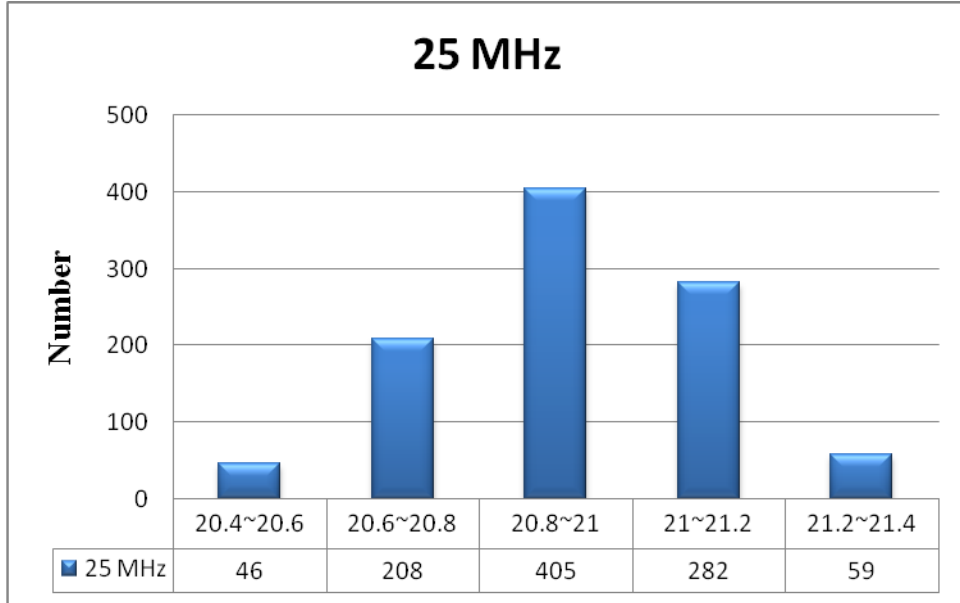


Figure 16 Energy consumption distribution of RSA uses 25 MHz

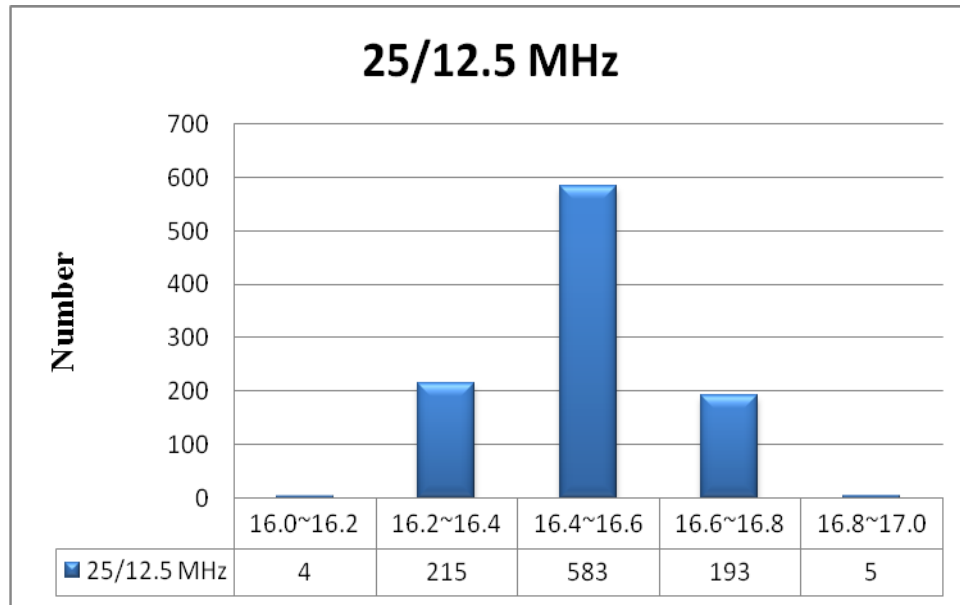


Figure 17 Energy consumption distribution of RSA uses 25/12.5 MHz

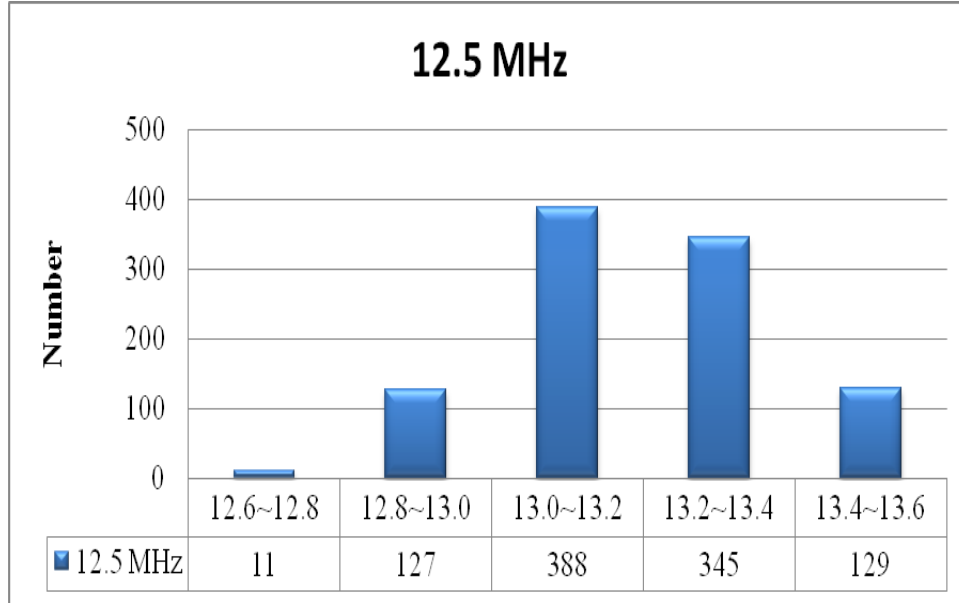


Figure 18 Energy consumption distribution of RSA uses 12.5 MHz

6. Conclusions

In this thesis, we measure energy consumptions of smart chip that embedded in microSD card. This approach adds many extra energy consumptions; we can determine that the energy consumption result is not real value that executes encryption algorithms. However, we determine that this approach is approximately practical applications. The practical applications exists extra energy consumptions of SSDC. Those extra energy consumptions need to be considered.

In different clock experiment, we use three types of cryptographic algorithms to study the relation of energy consumption, execution time and CPU clock. In random clock experiment, we use three different CPU clock to study that defend SPA attacks. From the experimental results we conclude that:

- For three types of cryptographic algorithms on SSDC, by reducing the clock rate, the increase of the execution time is faster than the decrease of the energy consumption.
- Reducing the CPU clock rate will have an impact on the execution time. In particular, the execution time increases faster when cryptographic algorithms need more amount of computations.
- Reducing the CPU clock rate will have an impact on the energy consumption. In particular, the energy consumption decreases slower when cryptographic algorithms need more amount of computations.
- By observing Figure 16, Figure 17, and Figure 18 of the energy consumption plots of using various clock schemes, we find that the random clock can be used to affect the energy consumption distribution and so is effective to defend SPA attacks.

From experimental results, the contributions of this thesis:

- The programmer can compute reality energy consumption to design agreements and applications.
- Select appropriate encryption to achieve the goal of saving energy consumption.
- The programmer can use random clock to resist SPA.

Bibliography

- [1] Chun I Fan, Yung Cheng Chan, and Zhi Kai Zhang, “Robust Remote Authentication Scheme with Smart Cards,” *Computers & Security*, vol. 24, issue 8, Nov. 2005, pp. 619-628.
- [2] Rongxing Lu, Zhenfu Cao, “Efficient Remote User Authentication Scheme Using Smart Card,” *Computer Networks*, vol. 49, issue 4, 15 Nov. 2005, pp. 535-540.
- [3] Nam-Yih Lee, Yu-Chung Chiu, “Improved Remote Authentication Scheme with Smart Card,” *Computer Standards & Interfaces*, vol. 27, issue 2, Jan. 2005, pp. 177-180.
- [4] J. Burke, J. McDonald, and T. Austin, “Architectural Support for Fast Symmetric-Key Cryptography,” *Proc. Intl. Conf. Architectural Support for Programming Languages and Operating Systems*, Nov. 2000, pp. 178-189.
- [5] Jing Xua, Wen-Tao Zhub, and Deng-Guo Feng, “An Efficient Mutual Authentication and Key Agreement Protocol Preserving User Anonymity in Mobile Networks,” *Computer Communications*, vol. 34, issue 3, Mar. 2011, pp. 319-325.
- [6] Kyungah Shim, “Cryptanalysis of Mutual Authentication and Key Exchange for Low Power Wireless Communications,” *IEEE Communications Letters*, vol. 7, issue 5, 2003, pp 248–250.

- [7] Qiu Liang Xu, Tzer Shyong Chen, "An Efficient Threshold RSA Digital Signature Scheme," *Applied Mathematics and Computation*, vol. 166, issue 1, July. 2005, pp. 25-34.
- [8] Yung Fu Changa, C.S. Chenb, and Hao Zhou, "Smart Phone for Mobile Commerce," *Computer Standards & Interfaces*, vol. 31, issue 4, June. 2009, pp. 740-747.
- [9] Lu Xiaoa, Howard M. Heys, "A Simple Power Analysis Attack Against the Key Schedule of the Camellia Block Cipher," *Information Processing Letters*, vol. 95, issue 3, Aug. 2005, pp. 409-412.
- [10] Adam Matthews, "Side-Channel Attacks on Smart Cards," *Network Security*, vol. 2006, issue 12, Dec. 2006, pp. 18-20.
- [11] N.R. Potlapally, S. Ravi, A. Raghunathan, and N.K. Jha, "A Study of the Energy Consumption Characteristics of Cryptographic Algorithms and Security Protocols," *IEEE Transl. Mobile Computing*, vol. 5, pp. 128, Feb. 2006.
- [12] P. Kocher, J. Jaffe, and B. Jun., "Differential Power Analysis," In: *Advances in Cryptology – CRYPTO'99*, Santa Barbara, California, USA, vol. 1666; Aug. 1999. pp. 388-397.
- [13] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems," *Communications of the ACM*, vol. 21, issue 2, Feb. 1978, pp. 120-128.
- [14] Paul Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems," *Advances in Cryptology*, 1996, pp. 104-113.

- [15] Security MicroSD Card, <http://www.go-trust.com/>
- [16] RFC 2631:Diffie–Hellman Key Agreement Method, June 1999.
- [17] FIPS 46-3: Data Encryption Standard, Oct. 1999.
- [18] FIPS 197: Advanced Encryption Standard, Nov. 2001.
- [19] IEEE 1363: Standard Specifications for Public-Key Cryptography.
- [20] OpenSSL Project, <http://www.openssl.org>
- [21] Chu Hsing Lin, Jung Chun Liu, and Chun Wei Liao, “Energy Analysis of Multimedia Video Decoding on Mobile Handheld Device,” 2007 International Conference on Multimedia and Ubiquitous Engineering, Seoul Korea, Apr. 2007, pp. 120-125.
- [22] Chu Hsing Lin, Jung Chun Liu, and Chun Wei Liao, “Power Consumption Analysis of Audio Application on Mobile Handheld Devices,” The IEEE TENCON 2007, Oct. 2007.
- [23] Chu Hsing Lin, Jung Chun Liu, and Mao Hua Cheng, “Energy Analysis of Multimedia Video Decoding on Embedded Systems,” The 22nd edition of the International Conference on Information Networking, Jan. 2008.
- [24] ISO/IEC 18092, <http://www.iso.org>
- [25] ST33F1M, <http://www.st.com>
- [26] National Instruments Corp, <http://www.ni.com>
- [27] <http://www.gutenberg.org/ebooks/1>
- [28] RFC 1321: The MD5 Message-Digest Algorithm, Apr. 1992.
- [29] RFC 3174: US Secure Hash Algorithm 1 (SHA1), Sep. 2001.

[30] FIPS 180-3: Secure Hash Standard, Oct. 2008.