THE REPUBLIC OF TURKEY
BAHÇEŞEHİR UNIVERSITY

IMPLEMENTING THE PAILLIER CRYPTOSYSTEM ON CONSTRAINED MICROCONTROLLERS FOR SMART GRID APPLICATIONS

Master's Thesis

ABDELRAHMAN ALKHODARY

İSTANBUL, 2016

THE REPUBLIC OF TURKEY
BAHÇEŞEHİR UNIVERSITY

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
ELECTRICAL AND ELECTRONICS ENGINEERING

IMPLEMENTING THE PAILLIER CRYPTOSYSTEM ON CONSTRAINED MICROCONTROLLERS FOR SMART GRID APPLICATIONS

Master's Thesis

ABDELRAHMAN ALKHODARY

Supervisor: SELÇUK BAKTIR

İSTANBUL, 2016

THE REPUBLIC OF TURKEY
BAHÇEŞEHİR UNIVERSITY
The Graduate School of Natural and Applied Sciences
Electrical and Electronics Engineering

Title of the Master's Thesis	:	Implementing the Paillier Cryptosystem on Constrained Microcontrollers for Smart Grid Applications
Name/Last Name of the Student	:	Abdelrahman ALKHODARY
Date of Thesis Defense	:	31 August, 2016

The thesis has been approved by The Graduate School of Natural and Applied Sciences.

Assoc. Prof. Dr. Nafiz ARICA
Graduate School Director

I certify that this thesis meets all the requirements as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Ayça Yalçin ÖZKUMUR
Program Coordinator

This is to certify that we have read this thesis and that we find it fully adequate in scope, quality and content, as a thesis for the degree of Master of Science.

Examining Commitee Members:	Signature
Asst. Prof. Dr. Selçuk BAKTIR (Supervisor)	: ....
Prof. Dr. Çiğdem Eroğlu ERDEM	: ....
Asst. Prof. Dr. Alptekin KÜPÇÜ	: ....

ACKNOWLEDGEMENTS

Praise be to ALLAH, Creator of the heavens and earth and peace be upon His messenger Muhammad (S.A.W).

First I want to thank my family, my wonderful and loving parents, without their support, belief and guidance I would not be what I am today. May Allah grant me the ability to always make them very proud. Next I want to thank my Supervisor Asst. Prof. Dr. Selçuk BAKTIR, Department of Computer Engineering, Bahcesehir University, your help and guidance over the period of my research has been unmeasurable, you have been a great supervisor. A special thanks to my colleague and friend Utku GÜLEN for his help. Also I want to thank Prof. Dr. Çiğdem Eroğlu ERDEM for her guidance and willingness to always help.

İstanbul, 2016 Abdelrahman ALKHODARY

ABSTRACT

IMPLEMENTING THE PAILLIER CRYPTOSYSTEM ON CONSTRAINED MICROCONTROLLERS FOR SMART GRID APPLICATIONS

Abdelrahman Alkhodary

Electrical and Electronics Engineering
Supervisor: Asst. Prof. Dr. Selçuk BAKTIR

August 2016, 34 Pages

Smart grid is the updated version of the electricity grid, as it will merge two-way communication to it along with other features. One of this features is the smart meters. Smart meter is one of the main components that defines smart grids. It provides advantages for utility providers and consumers. The information collected by smart meters is very valuable, that is why preserving the privacy of this information is very important for the security of smart grid. The privacy preserving of the data that is collected by smart meters is achieved by using homomorphic encryption scheme. In this thesis we provide an implementation results for the homomorphic encryption Paillier scheme for smart meters on constrained microcontroller MSP430.

Keywords: Paillier Encryption, Smart Grid, Smart Meter, MSP430 Microcontroller

ÖZET

Akıllı Şebeke Uygulamaları için Kısıtlı Microcontrollers
üzerinde paillier şifrelemesi uygulanması

Abdelrahman Alkhodary

Elektrik-Elektronik Mühendisliği
Tez Danışmanı: Yrd. Doç. Dr. Selçuk BAKTIR

Ağustos 2016, 34 Sayfa

Akıllı şebekeler elektrik ve bilginin iki yönlü iletilebilmesine olanak sağlayan ve geleneksel elektrik şebekesinin yerine geçmesi beklenilen modern elektrik iletim altyapılarıdır. Akıllı şebekelerin en önemli yapıtaşı akıllı sayaçlardır. Akıllı sayaçlar hem elektrik şirketlerine hem de kullanıcılarına sayısız avantajlar sağlar. Tüketilen elektrik miktarının akıllı sayaçlar tarafından elektrik şirketlerine iletilmesi elektrik üretim ve dağıtımının optimizasyonu açısından çok değerlidir, bununla birlikte bu hassas bilginin mahremiyetinin korunması akıllı şebeke güvenliği açısından büyük önem taşımaktadır. Bu tez ile akıllı sayaçlar tarafından toplanan elektrik tüketim bilgisinin mahremiyeti homomorfik bir şifreleme algoritması kullanılarak sağlanmaktadır. Bu tezde homomorfik Paillier şifreleme algoritmasının akıllı sayaçlarda yaygın olarak kullanılan kısıtlı mikrodenetleciler üzerinde performans etkin gerçeklemeleri yapılmıştır. Hedef mikrodenetleyici olarak MSP430 kullanılmıştır.

Anahtar Kelimeler: Paillier şifreleme algoritması, Akıllı Şebeke, Akıllı Sayaç, MSP430, Mikrodenetleyici

CONTENTS

TABLES

FIGURES

ABBREVIATIONS

HAN	:	Home Area Network
BAN	:	Buseiness Area Network
NAN	:	Neighbor Area Network
RSA	:	Rivest-Shamir-Adleman
HW	:	Hardware

1. INTRODUCTION

1.1 Smart Grid

The word GRID is usually used to describe the system that does four operations: electricity transmission, electricity generation, electricity control, and electricity distribution. ***Smart Grid*** is the updated version of the electricity grid that has been around for decades as it will inherit it by merging two-way communication and the capabilities of computing for more efficiency, reliability, safety and control [1]. The electricity is delivered between consumption and production points using two-way communication instead of one way communication from the consumer to the production point. It manages smart devices at consumers' buildings to reduce cost, save power and boost efficiency and reliability. The smart grid consists of traditional central-generator and/or renewable distributed-generator [2]. So, we can describe the smart grid as the electricity system that uses cyber-security, two-way communication technologies, and computational intelligence in an integrated way across distribution, transmission, generation and consumption of electricity to accomplish a system that is resilient, reliable, secure, safe, clean, sustainable, and efficient [3]. There are many survey papers that have been written for smart grids [4], [5], [6], [7].

Many technologies that we can find in the smart grid are used in other areas, for instance in manufacturing sensor networks and in telecommunication wireless networks. Figure 1.1 illustrates the content of communication infrastructure of a smart grid for instance Home Area Network (HAN), Neighbour Area Network (NAN), Business Area Network (BAN), substation automation integration systems and data center [8]. The communication infrastructure of the smart grid provides smart metering and monitoring techniques that can give feedback of energy consumption in real time and supply the required energy from utilities [9]. Network operations center can get online market pricing and customer's electricity usage data from data centers to generate and distribute the electricity more efficiently corresponding to consumers' energy demand.

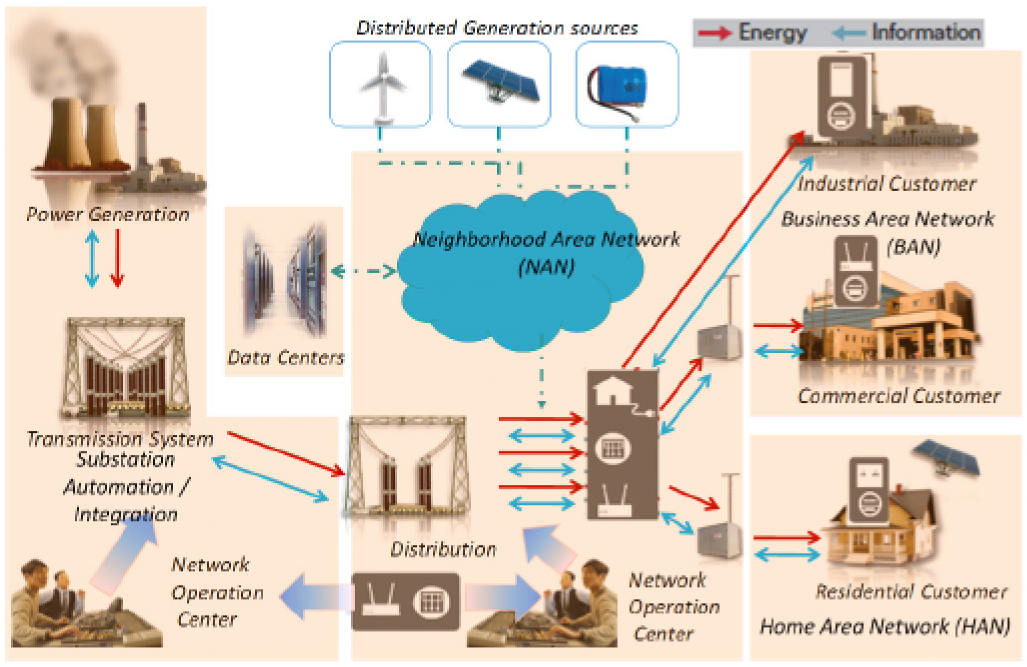


Figure 1.1: Communication Infrastructure Parts of Smart Grid [8]

2. SECURITY IN SMART GRIDS

2.1 Problem Definition

The reliance of smart grid on the interconnectivity between its elements make it more vulnerable to attacks. The main issue that we are trying to solve in this thesis is the problem of preserving the privacy of electricity consumption data by smart meters communicated in a smart grid.

2.1.1 Smart meter

Smart meter is considered to be one of the main components in smart grid, it is used to collect data about power consumption in households. The adoption of using smart meters provides advantages for utility providers and consumers. For utility providers, smart meters have better monitoring capabilities of electricity consumption than normal meters. As for the consumer, they can observe the pricing options and thus they can choose to use the high power consuming devices in non-peak times.

The information collected by smart meters is very valuable as it indicates the habits of the people by observing the energy consumption load profile, such as if they are at home or not. Also the devices they use can be detected by analyzing electricity consumption data.

2.1.2 Data aggregator

Smart meters are gathered in clusters, which encapsulate households from connected areas, e.g. houses from the same district area. Every cluster contains one data aggregator which can be physically connected or separated from one of the smart meters in the cluster. Smart meters in one cluster send their information to their data aggregator in their cluster. The data aggregator will collect all these information and send it to the power company. The power company will then receive information about energy consumption in cluster level and not in household level.

2.2 Related Work

Li et al. presented an approach for data aggregation for smart meters in smart grids that collects the data from the source units to the receiver unit [26]. To protect the user privacy, they used a homomorphic encryption scheme, namely the Paillier cryptosystem to secure the data enroute. The explanation of the system model is as follows.

According to the infrastructure of communication in the smart grid, the wired wireless multilayer architecture is very common. The smart meters are connected to the aggregator through a mesh wireless network. The aggregator is connected to the central management unit through a wired network. With limited coverage area, the aggregator may not be connected directly with all smart meters in its cluster, it is assumed that each smart meter will have more than one path to the aggregator through other smart meters in the cluster. Figure 2.1 shows a communication infrastructure with 15 homes in the cluster. Here, the aggregator collects the the smart meters' data in the cluster and computes the aggregation data and sends it to the central management unit. Each smart meter in the cluster will establish a connection to the aggregator, so it can report its data to it. In this approach, aggregation tree will be constructed virtually based on a mesh network topology. From down to top, every node in that tree will collect its children's data and calculates the aggregation data then sends it to the aggregator. Each of this data will be encrypted so that it is not to be exposed to other smart meters. As advantage of using the homomorphic cryptosystem is that one can make mathematical operations on the ciphertext without having to decrypt it first. For the given model there could be more than one spanning tree, so the presented protocol follows some rules to make sure to the get the best formation :

- the length of the aggregation tree should be small to shorten the distance between children smart meters and the aggregator unite.

- to prevent overload of the communication each path in the tree should not contain many nodes.

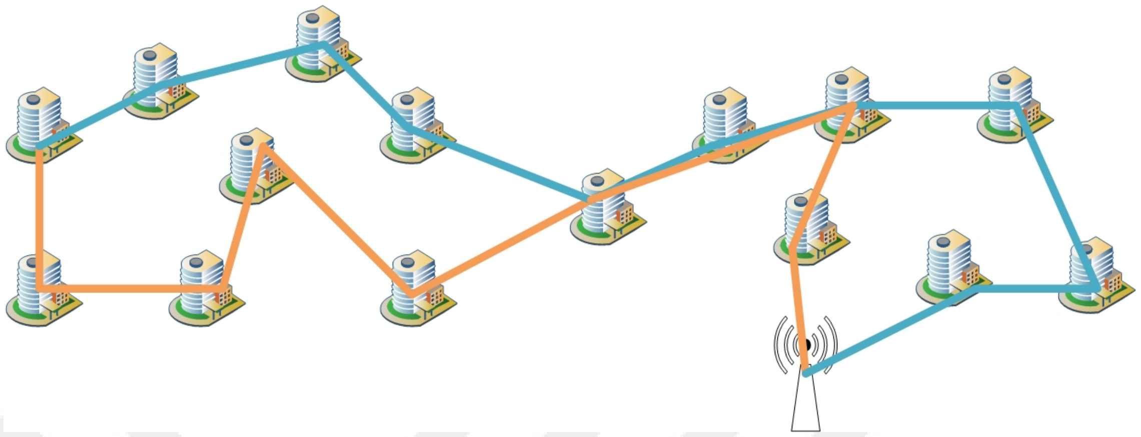


Figure 2.1: Connection between smart meters and data aggregator

In another paper, by Leontiadis et al., the authors proposed a solution for preserving the privacy of the data collected by every smart meter [28]. They took into account the operation of identifying the smart meter's maximum consumption as an intriguing factor for energy providers, as they can use it to predict the amount of energy that needs to be provided in advance. But in their solution they did not use the Paillier encryption scheme as they used the order preserved encryption scheme [27] by preserving the numerical data order in the space of the ciphertext.

In this thesis, we implemented the Paillier encryption scheme on the constrained microcontroller MSP430 to provide privacy-preserving for data communication for smart grids.

3. PAILLIER CRYPTOSYSTEM

3.1 Introduction to Encryption

Encryption schemes are originally designed to ensure the confidentiality of data. According to the Kerckhoffs principle [10,11], the security of any encryption scheme should not depend on the code obfuscation but it should rely on the decryption key secrecy. There are two distinguishable types of the encryption schemes, the asymmetric and symmetric schemes.

3.1.1 Symmetric Encryption Schemes

The word symmetric refers to the fact that decryption and encryption operations use the same key. So, the sender of the data and the receiver both of them have to agree on a key to be used in encryption and decryption for any communication between them. Usually a key is sent in a secure communication channel. This also means they have to share a symmetric key with any one they desire to connect with. Examples of symmetric schemes are block ciphers such as AES [12,13] and stream ciphers such as One Time Pad [14].

3.1.2 Asymmetric Encryption Scheme

The principle of asymmetric encryption schemes is different from the symmetric ones. Here, there are two keys, the public key that is advertised to everyone and the private key that is kept secret by the message receiver. Let us have Alice and Bob who want to communicate with each other. If Bob wants to communicate with Alice, he has to encrypt his message with her public key. This message can only be decrypted by Alice using her private key. Examples for asymmetric schemes are RSA [15] and ElGamal [16].

3.2 Homomorphic Encryption

A homomorphic cryptosystem is an encryption system that allows to perform mathematical operations such as sum/product on ciphertext messages without having to decrypt them, and the result will be the sum/product of the messages after decrypting them. This feature enables performing calculations on the cipher text without exposing the real information. Homomorphic encryption is usually used if there is an untrusted part in the computation process.

In homomorphic encryption the following equality holds, where $*$ represents the sum or product depending on whether the encryption algorithm is additive or multiplicative homomorphic.

$$E_k(X * Y) = E_k(X) * E_k(Y)$$

This feature was discovered accidentally by Rivest [17] soon after the creation of RSA. RSA is a multiplicative homomorphic encryption scheme, which considered partially homomorphic. Since then some solutions have been proposed to numerous applications using homomorphic encryption such as zero knowledge proofs [18], threshold scheme [19], electronic voting [20] and so on.

There are few partially homomorphic encryption schemes : ElGamal [16] and Paillier [21] encryption schemes are some of these. There is only one fully homomorphic encryption scheme invented by Gentry in his PhD Thesis in 2009 [22]. In smart grids for in-network data aggregation the additive homomorphic property is desired, that is why the Paillier encryption scheme is preferred for smart meters.

3.3 Paillier Cryptosystem

The Paillier cryptosystem was invented in 1999 by Pascal Paillier [21]. It is a probabilistic algorithm which means for a specific message, there is more than one ciphertext, hence the

Paillier cryptosystem is resilient against dictionary attacks. It is a public key encryption scheme with its security based on finding the n^{th} residue which is considered to be difficult computationally.

Key Generation : Private Key (q, p), where q,p are large primes

Public Key (g, n) where $n = p \times q$ and $g \in \mathbb{Z}^*_{n^2}$ is random

Encryption : Message $m < n$

Select a random number $r < n$

Ciphertext $c = g^m \times r^n \; mod \; n^2$

Decryption : Ciphertext $c \; < \; n^2$

Message $m = \frac{L(c^\lambda \; mod \; n^2)}{L(g^\lambda \; mod \; n^2)} \; mod \; n$

where $\lambda = lcm(p-1, q-1)$ and $L(t) = \left[\frac{t-1}{n}\right]$

The Paillier cryptosystem is additive homomorphic and hence the following equalities hold true for it:

$1)\; D(E(x_1).E(x_2) \; mod \; n^2) = x_1 + x_2 \; mod \; n,$

$2)\; D(E(x_1)x_2 \; mod \; n^2) = x_1 + x_2 \; mod \; n,$

$3)\; D(E(x_1)k \; mod \; n^2) = x_1.k \; mod \; n.$

The original Paillier cryptosystem has been used in various schemes after extending it. Ivan Damgard et al. submitted a scheme with moduli $n^t(t > 2)$ that can be used in electronic voting systems [23]. The Paillier cryptosystem is extended to a scheme over elliptic curves by Galbraith in [24]. Choi et al. submitted a way to eliminate the modular inversion in the decryption operation of the Paillier cryptosystem by changing the public key g generation [25]. The modified key satisfies the relation

$$g^{\lambda} = 1 + n \, mod \, n^2$$

In this thesis we are interested in achieving Paillier encryption efficiently on constrained microcontrollers in smart meters.

4. ALGORITHMS USED IN OUR IMPLEMENTATION

4.1 Multiplication

For many of the theoretic problems, including certain public key cryptosystems for instance RSA and the Paillier encryption schemes, the large integer multiplication is the essential operation of multi-precision integer arithmetic. The literature about the multiplication operation includes the standard pencil-and-paper method taught in grade school which is also known as the schoolbook method or the operand scanning method, the hybrid methods and operand caching. There are also asymptotically faster multiplication algorithms such as Karatsuba multiplication. Most of the known techniques for instance Karatsuba multiplication and operand caching are “divide and conquer” tools: they decrease the computation to smaller multiplication problems instead of one large multiplication problem. We will refer here to just the operand scanning, operand caching and Karatsuba methods as we implemented them in this thesis.

4.1.1 Operand Scanning

The operand scanning method is a reorganization of the standard paper schoolbook method. Let B and A be unsigned integers with length $n + 1$ and $t + 1$ respectively. The idea depends on using two nested loops, one of them is used to load from the first array $B[\,]$ and the second is used to load from the other $A[\,]$. The length of the product of the two integers will be at most the summation of their lengths $n + t + 2$.

Algorithm 4.1 Multiple-precision multiplication

Input: Two positive integers A, B with length $n+1, t+1$ respectively

Output: The product $A \cdot B = C_{n+t+1}....C_1C_0$

```
1:  for $i\ from\ 0\ to\ (n+t+1)$ do
2:      $C_i = 0$
3:  for $i\ from\ 0\ to\ t$ do
4:      $carry = 0$
5:      for $j\ from\ 0\ to\ n$ do
6:          $(u\ v)\ =\ C_{i+j}\ +\ A_j \cdot B_i\ +\ carry$
7:          $C_{i+j}\ =\ v$
8:          $carry\ =\ u$
9:      $C_{i+n+1}\ =\ u$
10:     return $(C_{n+t+1}....C_1C_0)$
```

Reference: [$Handbook\ of\ Applied\ Cryptography$, chapter14, page595][29]

also as illustrated in the figure 4.1

4.1.2 Operand Caching

The operand caching method was submitted by Hutter et al. [30]. The main idea behind this method is to minimize the number of access to the memory by caching the operand words. It is possible to save a significant number of load operations by performing a certain number of store operations for storing operand words from memory to registers and reusing them. There is a parameter for number of individual rows $r = \lfloor n/e \rfloor$ in operand caching where n is the words number in one operand. In our work, a 256-bits operand is spread over 16-bit registers, so $n = 16$, and $e = 5$, the number of registers for each A and B, that means five registers are preserved to store five words of A and another five for B, $r = 3$ which are r_1, r_2 and r_3 and one remaining point is the initialization block

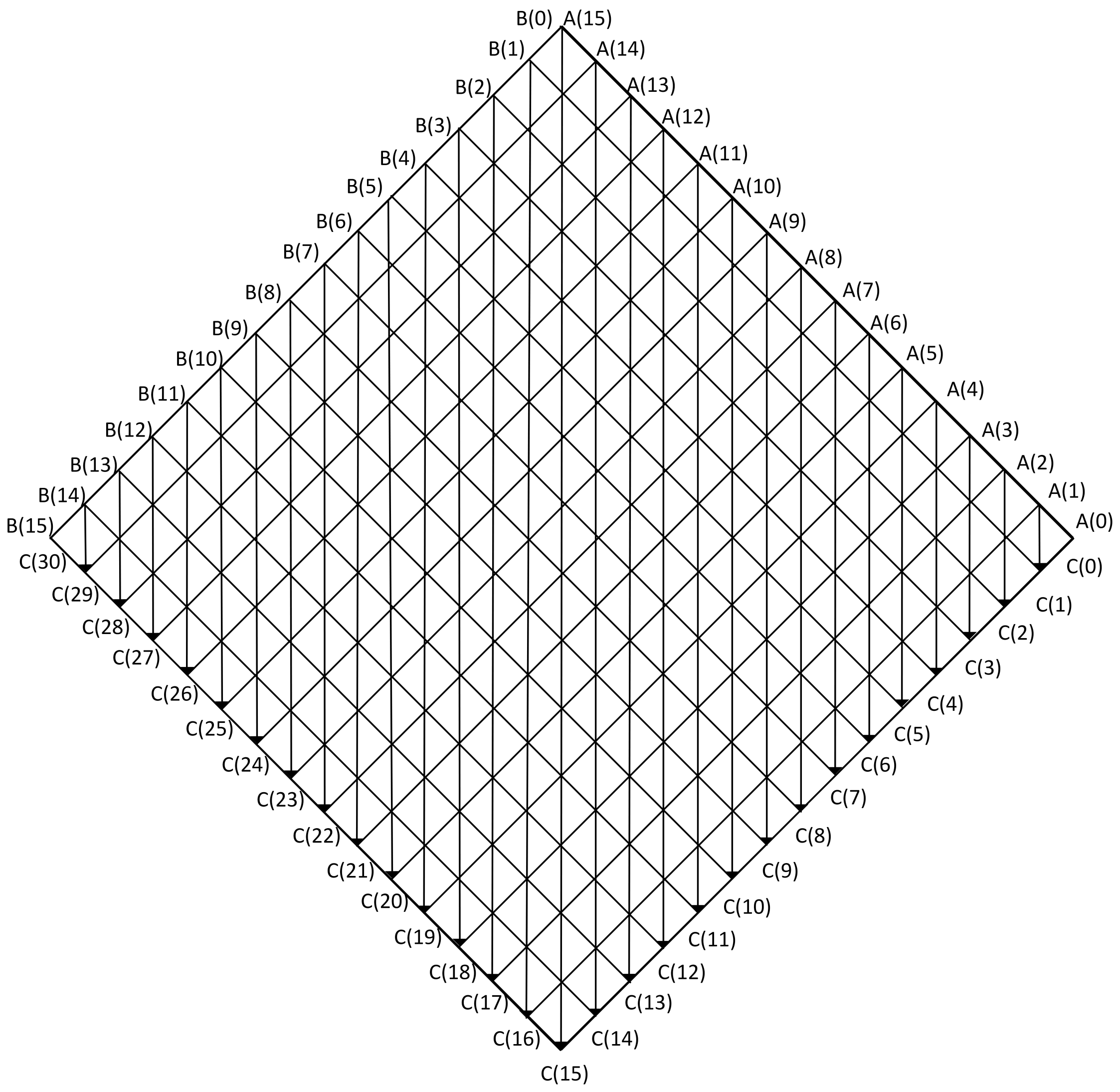


Figure 4.1: Operand Scanning for Multiplication of the two 256-bit Operands A and B on a 16-bit processor

b_{init}. The initialization block computes the products that were not included in the rows' computations.

On MSP430 there are only 16 registers and only 12 of these are available for general purpose use. We used five registers for each array with the help of the flash memory to save the addresses of the two multiplied arrays and the result array, using the absolute and indexed addressing mode we can load and restore information to the arrays. Absolute mode to load the address of the array into a register and indexed mode to load information from the array. The Algorithm starts with calculating the initialization block that starts from the top to the bottom, from the lower row to the higher one, and from the right to the left. We used the multiply and accumulate operands on the MSP430. The algorithm is illustrated more in the figure 4.2 .

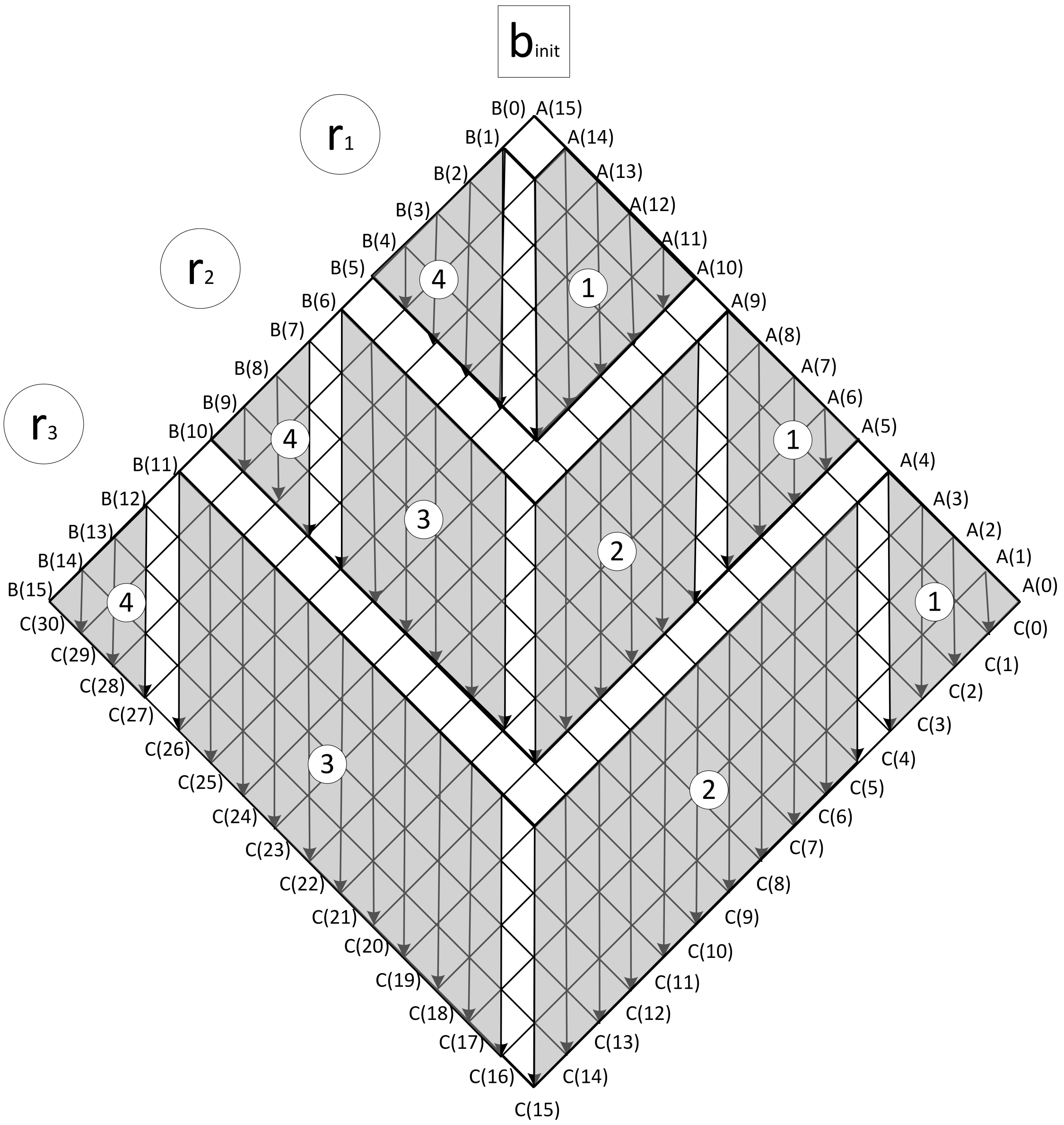


Figure 4.2: Operand Caching for Multiplication of the two 256-bit Operands A and B on a 16-bit processor

4.1.3 Karatsuba Multiplication

The Karatsuba algorithm was invented by Karatsuba and Ofman in 1962 [31]. It is a divide and conquer algorithm. Suppose we have two $2N$-bit integers as follows :

$$x = 2^N x_1 + x_0$$

$$y = 2^N y_1 + y_0$$

where x_0, x_1, y_0 and y_1 are N-bit unsigned integers. In Karatsuba Multiplication we perform one long multiplication which is $X \cdot Y$ with three half-length multiplications, four half-length additions and half-length one subtraction as follows :

$$Z = X \cdot Y = (2^N x_1 + x_0) \cdot (2^N y_1 + y_0)$$

$$Z = X \cdot Y = z_2 2^{2N} + z_1 2^N + z_0$$

$$z_2 = x_1 \cdot y_1$$

$$z_0 = x_0 \cdot y_0$$

$$z_1 = y_1 \cdot x_0 + y_0 \cdot x_1 = (y_1 + y_0)(x_1 + x_0) - z_0 - z_2 = (x_0 + x_1)(y_0 + y_1) - (z_0 + z_2)$$

The Karatsuba algorithm can be applied recursively for each half-length multiplication for better performance. In our work, we applied the above Karatsuba algorithm for one, two and three levels as illustrated in figures 4.3, 4.4 and 4.5 respectively .

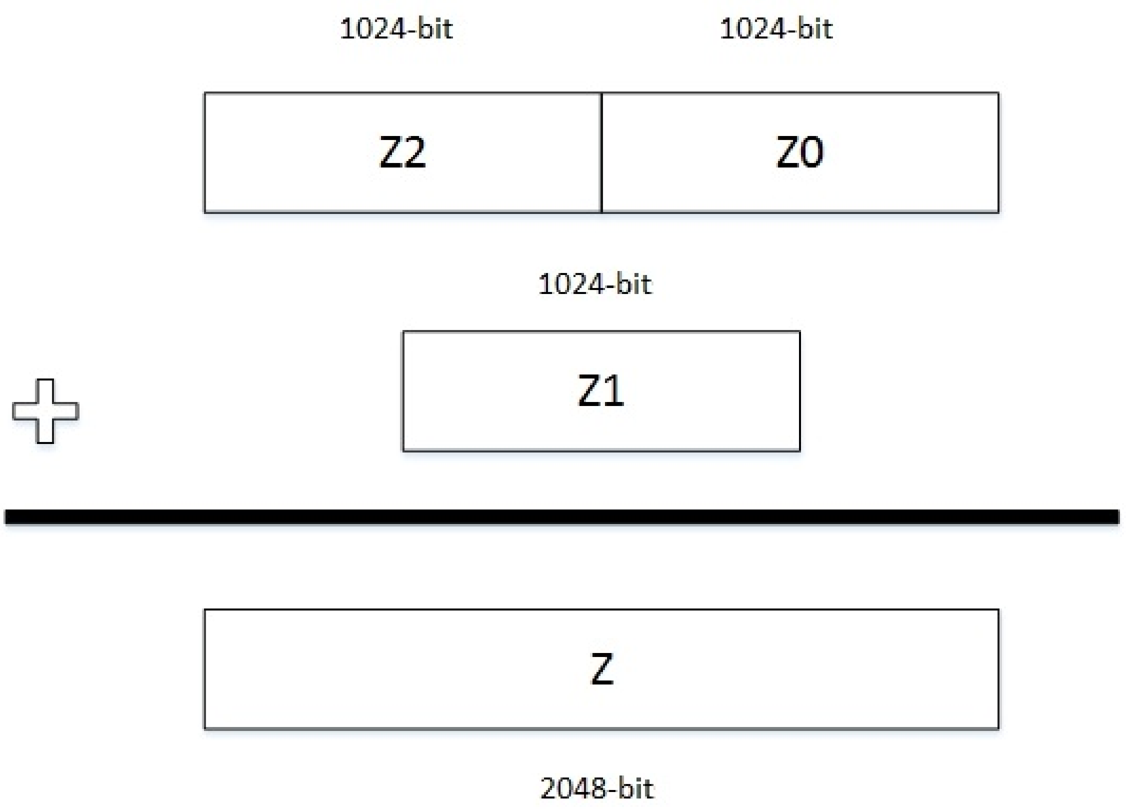


Figure 4.3: One-Level Karatsuba for 2048-bit Multiplication

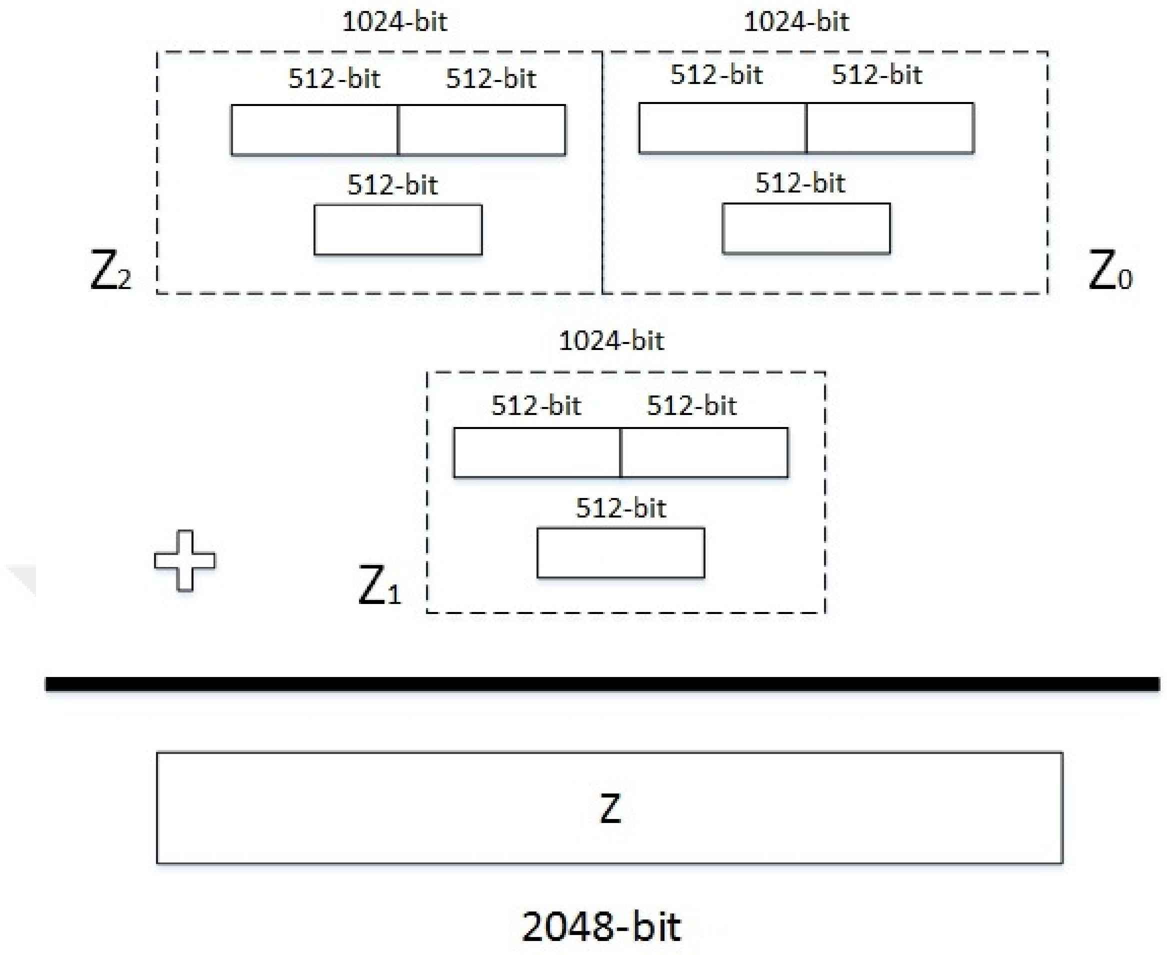


Figure 4.4: Two-Level Karatsubafor 2048-bit Multiplication

As we can see in the figure 4.4 each block of the above level of Karatsuba contains one-level beneath it.

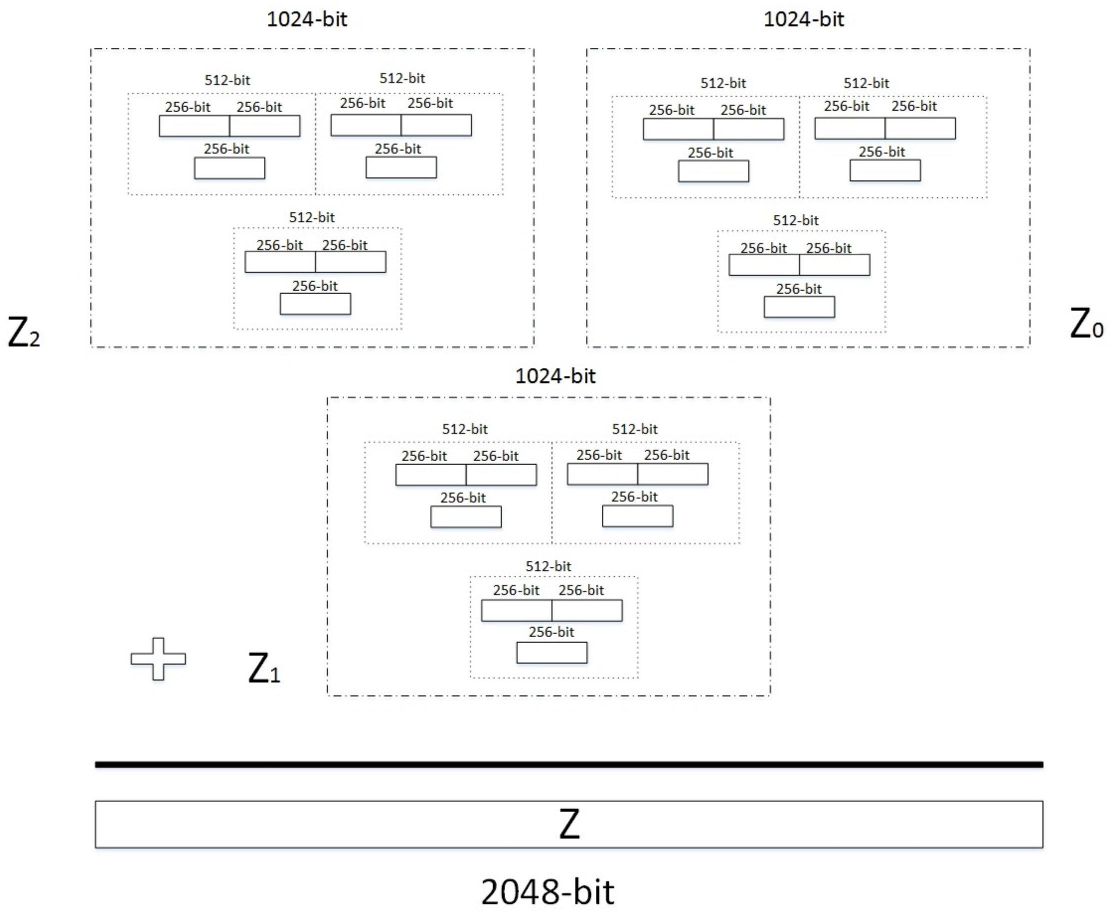


Figure 4.5: Three-Level Karatsuba for 2048-bit Multiplication

Also here in the figure each block of the highest level of Karatsuba contains one-level Karatsuba which in turn contains another level of Karatsuba.

A new variation of Karatsuba algorithm was proposed by Hutter et al. [32] which they called ***Subtractive Karatsuba***. This method depends on computing the two *absolute differences* $|x_0 - x_1|$ and $|y_0 - y_1|$, instead of computing two additions, which eliminates the load of taking care of the extra carry bit. This helps make the code more constant time. Suppose we have two $2N$-bit integers as follows :

$$X = 2^N x_1 + x_0$$

$$Y = 2^N y_1 + y_0$$

where x_0, x_1, y_0 and y_1 are N-bit unsigned integers.

$$L = x_0 \cdot y_0 \ , \ H = x_1 \cdot y_1$$

$$M = |x_0 - x_1| \cdot |y_0 - y_1|$$

$$r = r_x \oplus r_y$$

$$r = 0 \; if \; M = (x_0 - x_1) \cdot (y_0 - y_1)$$

$$r = 1 \; otherwise$$

$$Z = XY = L + 2^N(L + H - (-1)^r M) + 2^{2N} H$$

To get the *absolute differences* of the numbers, we first clear two registers r_x for X and r_y for Y. Then, we subtract each array with the instruction *SUBC* which is subtract with carry then we subtract the carry from the register r_x with the instruction *SBC* for the first array $(x_0 - x_1)$. If the value of $x_0 > x_1$ then the r_x will equal to $0xFFFF$, $0x0$ otherwise. After that we xor the register r_x with the resulted array $(x_0 - x_1)$ to get the one's complement, this operation will be done regardless of the value of r_x register to make the implementation time constant to be prone against time attacks, as if the value $0x0$ it will not change anything. Then *and* the register with one and add it to the $(x_0 - x_1)$ array to get the two's complement and rebel the carry through the array. Same operations will be done to the second array to get the absolute value of $(y_0 - y_1)$. We compute the xor value of r_x with r_y then store the resulted value in register r. The value of r will equal to zero if the array M ,which is the array resulted from multiplication of the two absolute differences we got before, equal to $M = (x_0 - x_1) \cdot (y_0 - y_1)$. The value of register r will be one otherwise. To compute the conditional negation of M which is $M' = (-1)^r M$. First we xor the array M with register r, then and r with one and add it to the M and rebel the carry. We will do this negation regardless of the value of the register r to avoid time attacks as we mentioned before. In figure 4.6 five levels of Subtractive Karatsuba is explained.

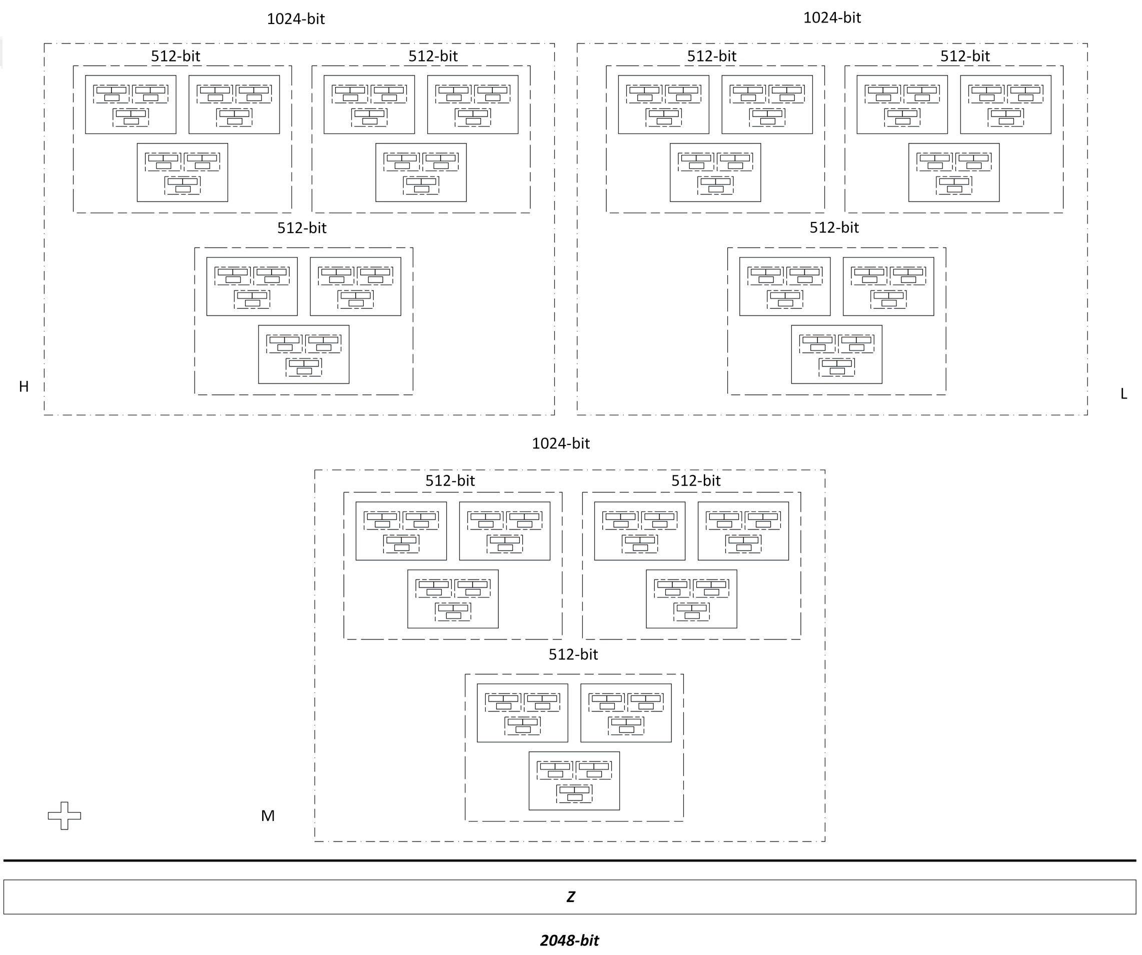


Figure 4.6: Five-Level of Subtractive Karatsuba for 2048-bit Multiplication

The first three levels in the figure 4.6 will be more explained in the figure 4.7.

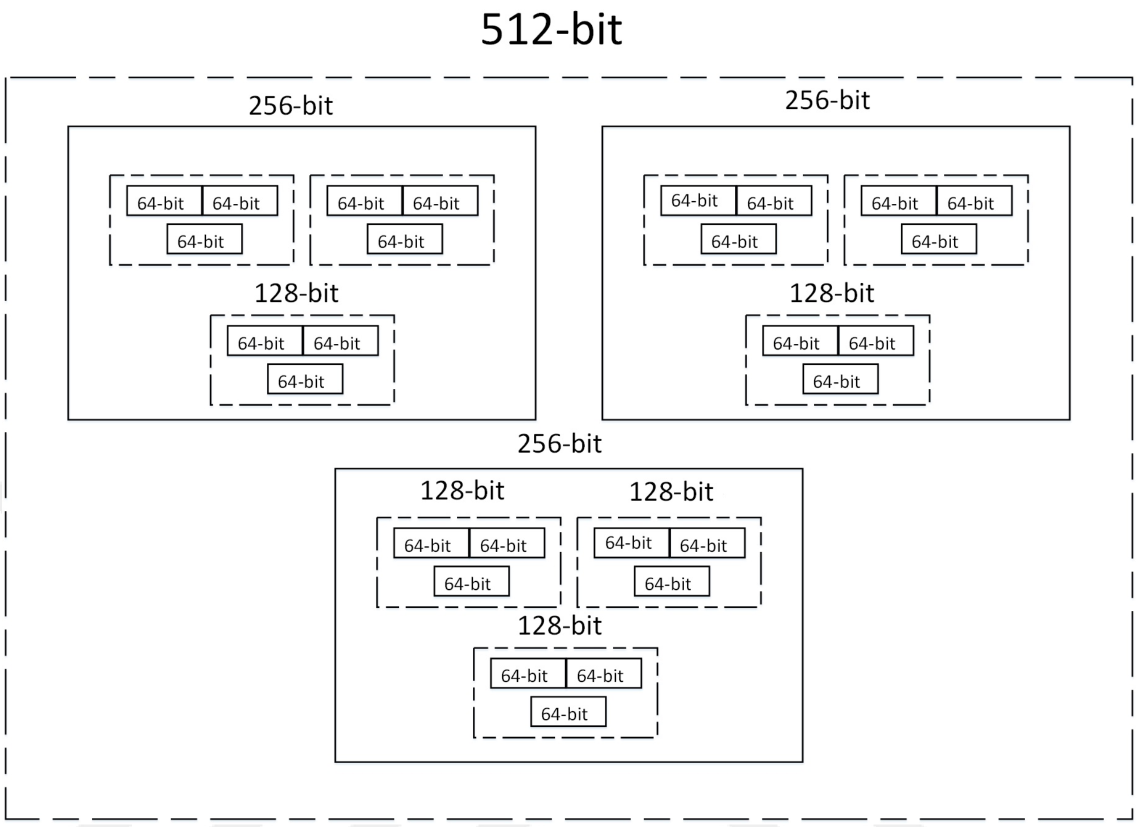


Figure 4.7: First Three-Level of Five-Level of Subtractive Karatsuba for 2048-bit Multiplication

4.2 Montgomery Modular Multiplication

Paillier encryption is achieved by 2 modular exponentiations as described in the equation. To provide 80-bit security, the bases G and R are 2048-bit integers and P and Q are 1024-bits. For realizing these exponentiations, a large number of 2048-bit modular multiplications should be computed. Optimizing the modular multiplication operation is crucially important in terms of speed, therefore we used the well-known Montgomery modular multiplication algorithm [33]. Here the 2048-bit integer multiplication operation is followed by a 2048-bit Montgomery reduction. In our work, we implemented 2048-bit Montgomery reduction by performing three 2048-bit integer multiplications, one 4096-bit addition and one 2048-bit subtraction in Algorithm 4.2. In Algorithm 4.2, the input operand is represented in Montgomery representation and the result is also in Montgomery residue representation.

Algorithm 4.2 Montgomery Reduction

```
Input: $T = A \cdot B$ where $A$, $B$, modulus $M$ are m-bits. $M' = -M^{-1} \bmod R$.
    Montgomery radix $R = 2^m$.
Output: $Z = T \cdot R^{-1} \bmod M$
1: $Q \leftarrow T \cdot M' \bmod 2^m$
2: $Z \leftarrow (T + Q \cdot M)/2^m$
3: if $Z \leq M$ then
4:     $Z \leftarrow Z - M$
5: Return $(Z)$
```

Reference: [$Handbook\ of\ Applied\ Cryptography$, chapter 14, page 601]

In Algorithm 4.2, 3 2048-bit integer multiplications are performed to obtain the 2048-bit Montgomery product. The value Z is obtained through the 4096-bit addition of T and $Q \cdot M$. Here, the least significant 2048 bits of Z will be 0 since M and M' are multiplicative inverses of each other in modulo 2^{2048}. Hence, division with 2^{2048} will be avoided. Lastly, Z must be checked to see if it is greater than M and if so, M will be subtracted from Z. After the Montgomery reduction operation, Z will be the output of the Montgomery modular multiplication which is 2048-bit long.

4.3 Exponentiation

In Paillier encryption we have to do exponentiations to get the ciphertext. We perform an exponentiation by doing repeated modular multiplication using Montgomery multiplication. The simplest way to do exponentiation is binary exponentiation by scanning the bits of the exponent as described in the Algorithm 4.3

Algorithm 4.3 Binary Exponentiation Method

```
Input: $g\ is\ a\ positive\ integer$ and $E = (e_{m-1}e_{m-2}......e_1e_0)_2$.
Output: $C = g^E \bmod N$
1: $A = g$
2: for from $i = m - 1$ to 0 do
3:     $g = g \cdot g \bmod N$
4:     if $e_i = 1$ then
5:         $g = g \cdot A \bmod N$
6: Return $C$
```

Reference: [$Handbook\ of\ Applied\ Cryptography$, chapter14, page 615]

4.3.1 K-ary Method

Algorithm 4.3 is slow since, we have to do m squarings and on average $m/2$ multiplication for performing an exponentiation with an m-bit exponent. There are other algorithms to perform this modular exponentiation operation. One of this algorithms is the K-ary method which requires some precomputations. Here, a desired window size is determined and precomputations are done. The algorithm is explained in details in Algorithm 4.4.

Algorithm 4.4 K-ary Method

```
Input: $g\ is\ a\ positive\ integer$ , $E = (e_{m-1}e_{m-2}......e_1e_0)_b$ where $b = 2^k$ for some
    $K \geq 1$
Output: $C = g^E \bmod N$
1: Precomputation
     • $g_0 = 1$
     • for $i\ from\ 1\ to\ (2^k - 1)$ do
   2:     $g_i = g \cdot g_{i-1} \bmod N$ Thus $g_i = g^i$
3: $C = 1$
4: for $i\ from\ t\ down\ to\ 0$ do
5:     $C = C^{2^k} \bmod N$
6:     $C = C \cdot g_{e_i} \bmod N$
7: Return ($C$)
```

Reference: [Handbook of Applied Cryptography, chapter 14, page 615]

5. OUR IMPLEMENTATION RESULTS AND CONCLUSION

We implemented 1024-bit Paillier Encryption on IAR Workbench by writing assembly subroutines and got timings using its cycle counter in debugging mode. To improve our timings, we used K-ary method with the window size of 4 bits. Also we used Montgomery multiplication and Karatsuba method for the 2048-bit integer multiplication operations. We used MSP430's on-chip hardware multiplier to perform word multiplications. Our first target microcontroller MSP430F149 has 16 x16-bit multiplier which achieve a word multiplication in 14 clock cycles. To speed up this operation, we utilized the 32x32-bit hardware multiplier that is supported by MSP430F5529, which is an advanced version of MSP430s. It achieved double length word multiplication in 28 clock cycles which is twice faster than doing the same operation using the 16x16-bit hardware multiplier.

MSP430F5529 has an extended instruction set and it is able to execute 20-bit operations. However, we used the same codes and instructions in our implementation with both microcontrollers except the hardware multiplier instructions. MSP430F5529 performs better with the help of its 32x32-bit hardware multiplier and renewed instruction set which requires less clock cycles for some instructions compared to MSP430F149. Also MSP430F5529 has a faster CPU which supports up to 25 MHz clock frequency while MSP430F149 runs at 8 MHz at the maximum.

5.1 Multiplication Results

For 16-bit Hardware Multiplier

We began by implementing multiplication using the Operand Scanning Method (Schoolbook method) for 2048-bit multiplication, then we tried the normal Karatsuba Multiplication for one, two, and three levels. Under three levels of Karatsuba we used 256-bit operand caching multiplication. After that we implemented the subtractive Karatsuba because it was faster. We implemented it with five level of Karatsuba based on 64-bit operand caching multiplication. Implementation results are included in the tables 5.1 and 5.2

Multiplication Method	Clock Cycles	Timings (ms)
Schoolbook	526,956	65.9
Karatsuba 2-level	297,835	37.2
Karatsuba 3-level	192,000	24
Karatsuba Sub. 3-level	136,719	17
Karatsuba Sub. 5-level	124,135	15.5

Table 5.1: Multiplication Methods using 16-HW Multiplier and 8 MHz speed

Multiplication Method	Clock Cycles	Timings (ms)
Schoolbook	526,956	21.08
Karatsuba 2-level	297,835	11.91
Karatsuba 3-level	192,000	7.68
Karatsuba Sub. 3-level	136,719	5.47
Karatsuba Sub. 5-level	124,135	4.97

Table 5.2: Multiplication Methods using 16-HW Multiplier and 25 MHz speed

For 32-bit Hardware Multiplier

We implemented our work using only three levels of Karatsuba using 256 bit operand caching multiplication as when we tried to go deeper to the fifth level of Karatsuba we found that it had become slower than three levels because of the addition and subtraction were more expansive than the multiplication. As the next table 5.3 shows a comparison of two levels Karatsuba and operand caching method for 256-bit multiplication. Implementation results are included in the tables 5.4 and 5.5

Multiplication Method	16-bit HW	32-bit HW
Karatsuba Sub. 2-level	3,525	2,702
Operand Caching	3,761	2,425

Table 5.3: Show the Comparison between 256-bit Multiplication Using Operand Caching and Two Levels Subtractive Karatsuba

Multiplication Method	Clock Cycles	Timings (ms)
Karatsuba 3-level	127,000	15.8
Karatsuba Sub. 3-level	102,069	12.8

Table 5.4: Multiplication Methods using 32-HW Multiplier and 8 MHz speed

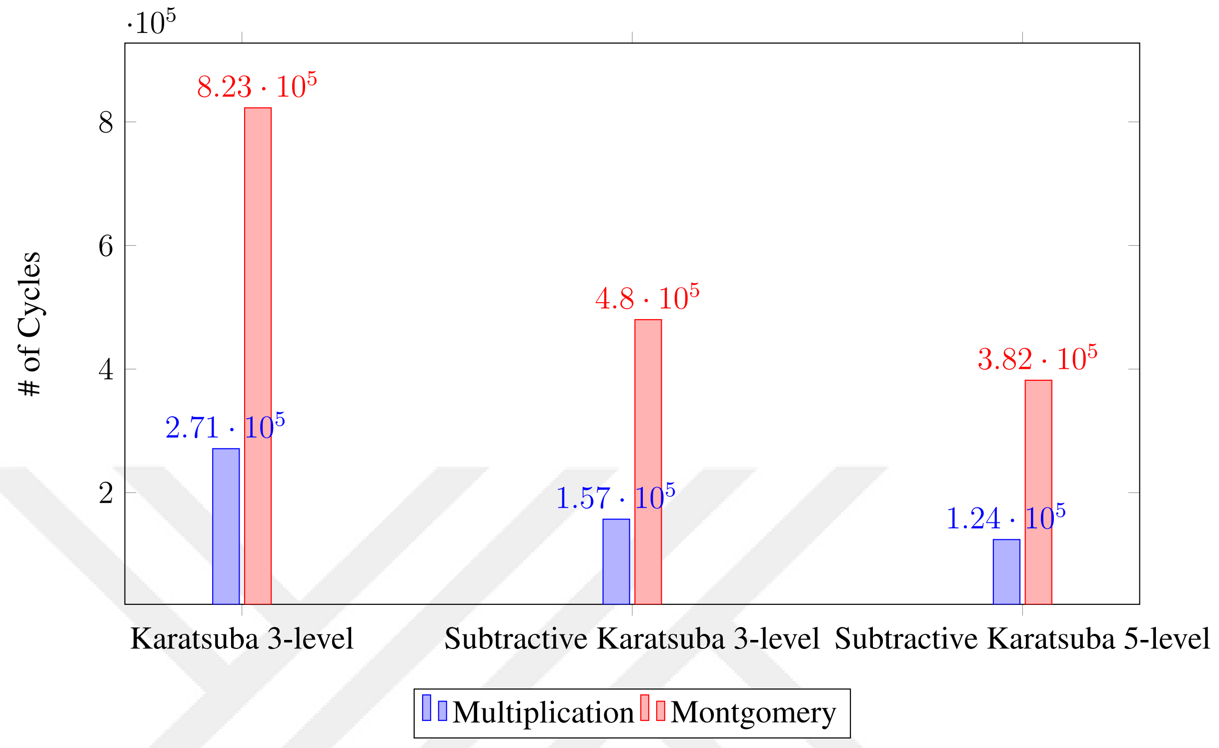
Multiplication Method	Clock Cycles	Timings(ms)
Karatsuba 3-level	127,000	5.08
Karatsuba Sub. 3-level	102,069	4.08

Table 5.5: Multiplication Methods using 32-HW Multiplier and 25 MHz speed

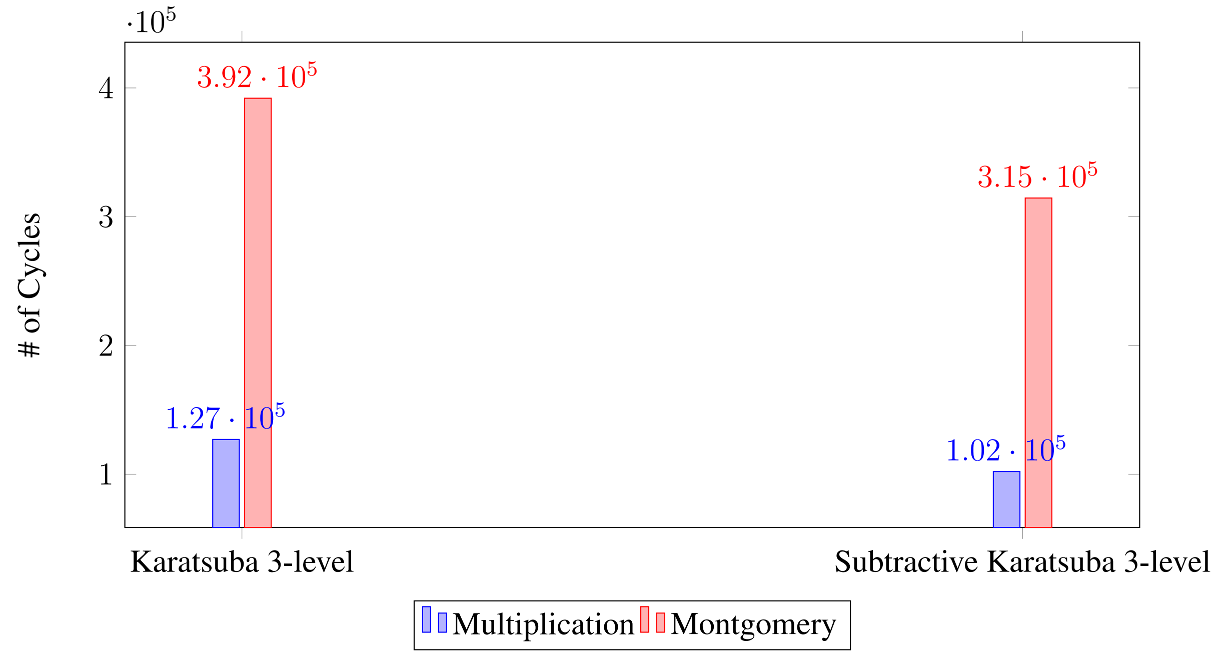
5.2 Montgomery Modular Reduction Results

Montgomery modular reduction is done by performing three 2048-bit integer multiplications, one 4096-bit addition and one 2048-bit subtraction as was explained in Algorithm 4.2. So, the speed of Montgomery modular reduction is dependent mainly on the integer multiplication method that is used. The comparison between the speed of Montgomery modular reduction that corresponding to different multiplication methods used in our implementation is explained in the next two bar chart one for the 16-bit hardware multiplier and the other for the 32-bit hardware multiplier.

5.2.1 For the 16-bit Hardware Multiplier



5.2.2 For the 32-bit Hardware Multiplier



5.3 K-ary Method

We used the window size of 4 in our implementations. The method needs 16 precomputations which are 0^{th} to 15^{th} powers of base integer, for the window size of 4-bit. The method scans the exponent 4 bits at a time and multiplies the corresponding precomputed value followed 4 square operations. Storing the precomputed values in the memory is a good trade of since only 4KB of memory space is needed and our target microcontrollers have more than 60KB flash memory available.

5.4 Generating The Random Integer R

The Paillier encryption scheme requires a large random integer R which is a 2048-bit integer in our case. We included 128-bit AES encryption code to our implementation in order to produce R. Since R is 2048 bits in length, we execute 128-bit AES encryption 16 times sequentially for generating R. Our AES encryption implementation takes two 128-bit initial vectors as the key and the initial plaintext and then uses the result of each encryption as the input to the next encryption operation. Thus, after 16 runs of AES a 2048-bit R value is produced to be used in Paillier encryption. The timing of 128-bit AES encryption is negligible compared to overall cost of Paillier encryption in our implementations.

5.5 Conclusion

Table 5.6 and Table 5.7 present core operation timings on MSP430 microcontrollers which we realized in our work. The main reason that MSP430F149 requires more clock cycles is because it has a smaller hardware multiplier. At 8 MHz clock frequency, MSP430F149 executes our 1024-bit Paillier encryption code in 61.8 sec. On the other hand, MSP430F5529 shows better performance with 32x32-bit hardware multiplier and its extended instruction set. At 25 MHz clock speed,MSP430F5529 executed 1024-bit Paillier encryption in only 32.6 sec nearly.

	Clock Cycles	Seconds
2048-bit AES Random Integer Generation	197,095	0.025
2048-bit Karatsuba Multiplication	124,135	0.016
2048-bit Montgomery Multiplication	381.932	0.047
1024-bit Paillier Encryption	988,477,160	61.8

Table 5.6: Timings of core operations for Paillier encryption on MSP430F149 running at 8 MHz

	Clock Cycles	Seconds
2048-bit AES Random Integer Generation	204,199	0.008
2048-bit Karatsuba Multiplication	102,069	0.004
2048-bit Montgomery Multiplication	314,500	0.012
1024-bit Paillier Encryption	817,124,938	32.68

Table 5.7: Timings of core operations for Paillier encryption on MSP430F5529 running at 25 MHz

6. REFERRENCES

[1] U.S. Department of Energy, National Energy Technology Laboratory, “A vision for the modern Grid,” March 2007.

[2] F. Rahimi and A. Ipakchi, “Demand Response as a Market Resource Under the Smart Grid Paradigm,” IEEE Trans. Smart Grid, vol.1, no.1, pp.82-88, June 2010.

[3] H. Gharavi and R. Ghafurian. Smart grid: The electric energy system of the future. Proc. IEEE, 99(6):917 – 921, 2011.

[4] B. Akyol, H. Kirkham, S. Clements, and M. Hadley. A survey of wireless communications for the electric power system. Prepared for the US Department of Energy. 2010

[5] T. Baumeister. Literature review on smart grid cyber security, Technical Report, http://csdl.ics.hawaii.edu/techreports/10-11/10-11.pdf. 2010.

[6] H. E. Brown and S. Suryanarayanan. A survey seeking a definition of a smart distribution system. North American Power Symposium’09, pages 1–7, 2009.

[7] T. M. Chen. Survey of cyber security issues in smart grids. CyberSecurity, Situation Management, and Impact Assessment II; and Visual Analytics for Homeland Defense and Security II (part of SPIE DSS 2010), pages 77090D–1–77090D–11, 2010.

[8] R. Yu, Y. Zhang, S. Gjessing, C. Yuen, S. Xie, M. Guizani, “Cognitive radio based hierarchical communications infrastructure for smart grid,” IEEE Network, vol.25, no.5, pp.6-14, September-October 2011.

[9] J. G. Cupp and M. E. Beehler, “Implementing Smart Grid Communications” TECH-Briefs 2008 No. 4, pp. 5-8.

[10] A. Kerckhoffs, “La cryptographie militaire (part i),” Journal des Sciences Militaires, vol. 9, no. 1, pp. 5–38, 1883.

[11] A. Kerckhoffs, “La cryptographie militaire (part ii),” Journal des Sciences Militaires, vol. 9, no. 2, pp. 161–191, 1883.

[12] J. Daemen and V. Rijmen, “The block cipher RIJNDAEL,” in (CARDIS ’98), vol. 1820 of Lecture Notes in Computer Science, pp. 247–256, Springer, New York, NY, USA, 2000.

[13] J. Daemen and V. Rijmen, “The design of Rijndael,” in AES— the Advanced Encryption Standard, Information Security and Cryptography, Springer, New York, NY, USA, 2002.

[14] G. Vernam, “Cipher printing telegraph systems for secret wire and radio telegraphic communications,” Journal of the American Institute of Electrical Engineers, vol. 45, pp. 109–115, 1926.

[15] R. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” Communications of the ACM, vol. 21, no. 2, pp. 120–126, 1978.

[16] T. ElGamal, “A prublic key cryptosystem and a signature scheme based on discrete logarithms,” in Advances in Cryptology CRYPTO ’84), vol. 196 of Lecture Notes in Computer Science, pp. 10–18, Springer, New York, NY, USA, 1985.

[17] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In Foundations of Secure Computation, pages 169–180, 1978.

[18] R. Cramer and I. Damgard, “Zero-knowledge for finite field arthmetic, or: can zero-knowledge be for free?” in Advances in Cryptology (CRYPTO ’98), vol. 1462 of Lecture Notes in Computer Science, pp. 424–441, Springer, New York, NY, USA, 1998.

[19] D. Rappe, Homomorphic cryptosystems and their applications, Ph.D. thesis, University of Dortmund, Dortmund, Germany, 2004, http://www.rappe.de/doerte/Diss.pdf.

[20] P.-A. Fouque, G. Poupard, and J. Stern, “Sharing decryption in the context of voting or lotteries,” in Proceedings of the 4th International Conference on Financial Cryptography, vol. 1962 of Lecture Notes in Computer Science, pp. 90–104, Anguilla, BritishWest Indies, 2000.

[21] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. Eurocrypt ’99, pp. 223–238.

[22] Gentry, C. (2009). Fully Homomorphic Encryption Using Ideal Lattices. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC’09), pp. 169-178, ACM Press, New York, NY, USA.

[23] I. Damgard and M. Jurik, “A generalization, a simplification and some applications of Paillier’s probabilistic public-key system, ” PKC 2001, LNCS 1992, pp.119-136, 2001.

[24] S. D. Galbraith, Elliptic curve Paillier schemes, Journal of Cryptology, Vol. 15, No. 2 (2002) 129–138. (available from http://www.isg.rhul.ac.uk/ sdg/)

[25] D. -H. Choi, S. Choi, and D. Won, “Improvement of probabilistic public key cryptosystem using discrete logarithm,” The 4th International Conference on Information Security and Cryptology, ICISC 2001, LNCS 2288, pp.72-80, 2002.

[26] Li, F., Luo, B. and Liu, P. (2011)‘Secure and privacy-preserving information aggregation for smart grids’, Int. J. Security and Networks, Vol. 6, No. 1, pp.28–39.

[27] Iraklis Leontiadis, Refik Molva, Melek Önen:Privacy Preserving Statistics in the Smart Grid. ICDCS Workshops 2014: 182-187

[28] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order-preserving encryption for numeric data. In SIGMOD Conference, pages 563–574, 2004.

[29] Handbook of applied cryptography Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone

[30] M. Hutter and E. Wenger. Fast multi-precision multiplication for public-key cryptography on embedded microprocessors. In Cryptographic Hardware and Embedded Systems HES 2011, pages 459:474, 2011. $https : //online.tugraz.at/tug_online/voe_main2.getvolltext?pCurrPk = 58138$

[31] Anatoly A. Karatsuba, Y. Ofman, “Multiplication of multi-digit numbers on automata”, Soviet Physics Doklady 7, 1963, pp. 595-596.

[32] M. Hutter and P. Schwabe. NaCl on 8-bit AVR microcontrollers. In A. Youssef, A. Nitaj, and A. E. Hassanien, editors, Progress in Cryptology AFRICACRYPT 2013, volume 7918 of Lecture Notes in Computer Science, pages 156:172. Springer, 2013. $http : //cryptojedi.org/papers/avrnacl - 20130220.pdf$

[33] Peter Montgomery. Modular Multiplication Without Trial Division, Math. Computation, vol. 44, pp. 519:521, 1985.

CURRICULUM VITAE

Name Surname : Abdelrahman Alkhodary

Date and Place of Birth : 18/03/1989 Saudi Arabia

M.S.: Bahçeşehir University, Electrical and Electronics Engineering

B.S. : Alexandria University ,Communication and Electronics Engineering

Work Experience :

- Research Assistant, Bahcesehir University, Istanbul, Ongoing since April 2016.