Hardware Low Power implementation of Attribute-Based Encryption

Mohamed M. Abdel-Aziz, Amr T. Abdel-Hamid
Electronics Department, Information Engineering and Technology,
German University in Cairo (GUC), Cairo, Egypt
mohammed.mamdoh@student.guc.edu.eg , amr.talaat@guc.edu.eg

Abstract—**Attribute-Based Encryption (ABE) is a novel privacy-preservation encryption approach that changes its key according to user attributes and a policy between these attributes. These attributes can be the user's e-mail, IP address, etc. This paper proposes the usage of ABE for securing Internet of Things (IoT) networks. In this paper, we implemented the first hardware Ciphertext policy Attribute-Based Encryption (CP-ABE) on Field Programmable Gate Array (FPGA), with the aim of providing a low power, low area secure hardware-based privacy-preservation encryption to be used in different Internet of Things (IoT) applications.**

Keywords— Attribute-Based Encryption (ABE); Internet of Thing (IoT); Hardware implementation

I. INTRODUCTION

With the ever increasing number of connected devices and the overabundance of data generated by these devices, data privacy has become a critical concern in the Internet of Things (IoT). One promising privacy-preservation approach is Attribute-Based Encryption (ABE), a public key encryption scheme that enables fine-grained access control, scalable key management, and flexible data distribution.

Attribute-Based Encryption (ABE) [1] which was introduced by Sahai and Waters in 2005 is a type of asymmetric Encryption. An ABE system usually consists of a key authority, senders, and recipients. The key authority authenticates senders and recipients, generates public/private keys, and issues the keys to publishers and subscribers [2]. A policy is generated by the system between the attributes of the users which adds privacy to the authorized users in the system. Fig.1 shows the Ciphertext Policy-Attribute-Based Encryption (CP-ABE) scheme which describes how the encryption works with (PK_A and MSK_A), (PK_B and MSK_B), (PK_C and MSK_C) are the public key and the master secret key of the attributes A, B, C respectively. (SK_{Alice} and SK_{Bob}) are the secret keys of the users Alice and Bob respectively.

In this paper, we proposed the first hardware ABE implementation based on quaternion ring operations proposed in [3]. The quaternion ring is a type of noncommutative ring that is represented by a 4-dimensional vector of complex numbers (1,i,j,k) and does operations on this vector. To achieve our goals, we had to model the whole encryption and decryption processes as well as the

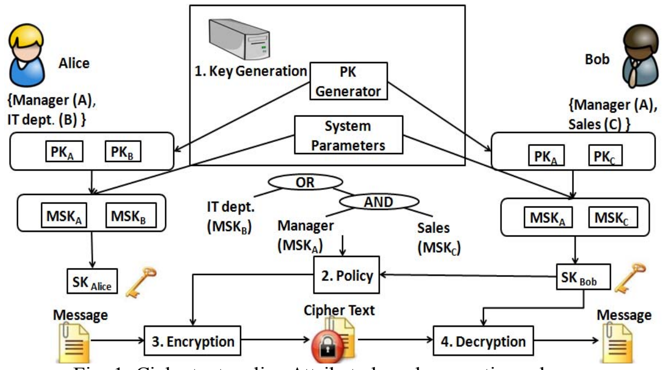

Fig. 1. Ciphertext policy Attribute based encryption scheme

quaternion ring operations using MATLAB. This process was done to ensure the proper operation of the algorithm as well as to identify all quaternion operations needed to be implemented. The identified operations were analyzed to basic binary operations. All these basic operations were built and their architecture was chosen to ensure the lowest Delay/Area product to ensure the lowest power and the highest throughput possible. Quaternion operations were implemented and integrated to provide a complete ABE Encryption/Decryption scheme and the whole system was verified against the MATLAB model to ensure its proper operations. Fig.2 shows our methodology taken to reach our objectives.

II. ATTRIBUTE BASED ENCRYPTION

ABE is a type of encryption that provides the IoT network with privacy and security through a policy between the attributes of the users in the system. ABE consists of two types, Key Policy ABE (KP-ABE) and Ciphertext- Policy ABE (CP-ABE).

In CP-ABE, the sender's data access policy is embedded in the ciphertext, and a recipient's attributes are associated with its private keys. A sender can decrypt the ciphertext only if the attributes associated with its private key satisfy the access policy embedded in the encrypted data [2].

In this paper, we will discuss the CP-ABE as the sender of the message could control who can access the message through a policy including all the attributes in the system. This paper is based on [3] in its implementation of ABE but implements it in hardware. The design of the algorithms is

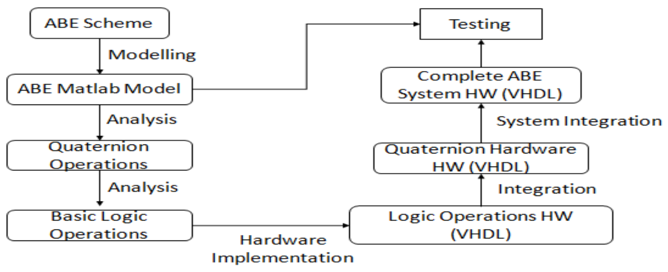

Fig. 2. ABE system methodology

based on the quaternion ring operations that depend on operations over *(Fq = {0, 1, 2, 3… q-1})* where (*Fq*) is a finite field of binary numbers where q is a prime number.
The algorithm generates as random the list of quaternions (*LQ*) defined in (2) which are composed of (*n*) quaternion vectors *Q(i)*s defined in (1), the prime number (*q*), the public keys (*PK*) of the users' attributes.

$$Q(i) = \{qi1, qi2, qi3, qi4\} \quad (1)$$

Where $(i = 1,2,..,n)$ $qik \in (Fq)$ defined in (1), $(k = 1,2,3,4)$
And norm of $Q(i) \bmod q \neq 0$

$$LQ = \{Q(1), Q(2), Q(3) \dots Q(n)\} \quad (2)$$

Points to consider:

1. The system parameters SP = [*q, c, r, n*] are selected where *q* is a prime number, *c* is the class category which is the category of the users, *r* is the number of the rank which defines the security level of the message, *n* is the number of *Q(i)* in *LQ*.
2. The quaternion *Q(i)* should satisfy that the norm of *Q(i) mod q* should be ≠ 0.
3. The *PK* of each attribute of the users can be represented by the 4-dimensional vector *V(h,o)* defined in (3).
   $$V(h,o) = \{vho(1), vho(2), vho(3), vho(4)\} \quad (3)$$
   Where ($h=a,b,c,d,\dots c;\ o=1,2..,r$), $vhok \in (Fq^* = \{1, 2, 3 \dots q-1\})$, *k* is the position of *vho* in the vector *V(h,o)* i.e. $k = (1,2,3,4)$.
4. The *MSK* of each attribute of the users whose attributes are class *h* and rank *o* can be denoted by the quatrenion *E(h,o)* defined in (4)
   $$E(h,o) = \{eho(1), eho(2), eho(3), eho(4)\} \quad (4)$$
   Where ($h=a,b,c,d,\dots c;\ o=1,2..,r$), $ehok \in (Fq)$, *k* is the position of *eho* in the quaternion *E(h,o)* i.e. *k* = (1,2,3,4).
5. The FPGA classifies the users according to their attributes' *MSK*s i.e. class and rank. The attributes of the users are embedded in their *MSK*.
6. *E(h,o)* is calculated from *LQ* and *V(h,o)* as defined in (5).
   $$E(h,o) = Q(vho(1)) * Q(vho(2)) * Q(vho(3)) * Q(vho(4)) \bmod q \quad (5)$$

A. Key Generation

The algorithm takes as input {*PK*, *q*, and *LQ*} and outputs {*MSK*}. For an example if the *PK* of a user's attribute class *A* and rank 1 *V(a,1)=(1,2,3,l)* and *PK* of a user's attribute class *B* and rank 1*V(b,1) = (3,3,2,1)* so from the *LQ*, the *MSK* of the user's attribute *(a,1)* is

$$E(a,1) = Q(1) * Q(2) * Q(3) * Q(1) \bmod q$$

and the *MSK* of the user's attribute *(b,1)* is

$$E(b,1) = Q(3) * Q(3) * Q(2) * Q(1) \bmod q.$$

B. Policy

It is used to set a certain policy or access structure between the *MSK* of user's attributes which control the access of the message. The algorithm takes as input {*q*, *MSK*, and the unknown secret key (*X*) of the user who wants to decrypt the message *M*} and it outputs the vector (*K(X)*) which represents the (*MSK*) of the user's attribute that fulfills the policy condition. The attributes of the users are embedded in their (*MSK*). An example of the policy is {*A OR B*} where A and B are some attributes. The policy is written as {*E(a,1) OR E(b,1)*}. *E(a,1)* is *MSK(A)*, *E(b,1)* is *MSK(B)*. *K(X)* is calculated from (6) by letting *A = E(a,1)* and *B = E(b,1)*.

$$K(X) = OR[A,B,X] = B[(1,0,0,0) - A * B^{-1}]^{-1} * [(1,0,0,0) - A * X] + A[(1,0,0,0) - B * (A^{-1})]^{-1} * [(1,0,0,0) - B * X] \bmod q \quad (6)$$

C. Encryption

It is used to encrypt the message with a ciphertext. The algorithm takes as an input the following: {*K(x)* from the previous algorithm, the prime number *q*, the message *M*} and it outputs: {the ciphertext *C(x)*}. Let the conjugate of *K(X)* be denoted by *conj (K(X))*. The ciphertext is calculated according to (7).

$$C(X) = (K(X) * M * conj\,(K(X)) \bmod q \quad (7)$$

D. Decryption

It generates the message (*M*) if the user decrypting the message is an authorized user. It takes as an input {the *SK* of the user, the prime number q, the MSK and the ciphertext *C(SK)*}. It outputs {*M*}. When the message is available the policy algorithm is run with *X = SK* to calculate *K(SK)* then the encryption algorithm is run to calculate the *C(SK)* which is taken an input to the decryption algorithm. Let the norm of *MSK* be denoted by *norm (MSK)*. Let the *MSK* of the user *(a,1)* who has the attribute *A* to be *E(a,1)*. The message is decrypted according to (8) by letting $A = E(a,1)$, $B = E(a,1)^{-1}$ which is his *SK*, $C(X) = C(E(a,1)^{-1})$ which is the ciphertext, *D = E(a,1)* which is his attribute's *MSK*.

$$M = \frac{1}{norm(A)} * B * C(X) * D \bmod q \quad (8)$$

III. Implementation

A. Logic implementation

1) Binary operations

Binary operations are the fundamental building blocks in the hardware implementation of the quaternion ring operations that are used to implement the ABE.

a) *Adder/ Subtractor*

The adder is the main building block of the other binary operations. We implemented a 16-bit Brent-Kung parallel prefix adder [4] [5] as they are among the best adders, with respect to the area and time and power consumption (cost: performance ratio). Then all bits of the sum will begin the process concurrently. Table I shows a comparison between Brent-kung adder and other adders. The subtractor is implemented inside the adder with a select input to select whether to add or subtract.

b) *Multiplier*

There are many types of multipliers such as Array multipliers, Wallace tree multipliers. We choose to implement 8x8 bit Vedic multiplier based on [6] because of its low power consumption compared to other multipliers. The following Table II shows a comparison between 8-bit Vedic multiplier and other multipliers.

c) *Modulus*

The modulus is needed for the quaternion operations to be done in the finite field Fq. We implemented 16-bit dividend by 16-bit divisor. The successive adding or subtracting between the dividend and the divisor is designed using a clock input.

d) *Division*

We implemented 8 by 4 bit division to get 4 bit quotient. It is done by storing both the numerator and the denominator in registers and then shifting the registers of the numerator with each other and then loop four times with the help of an external clock through the registers of the numerator until the 4 loops are done then outputs the quotient.

2) Quaternion Ring

The operations are implemented using the binary operations. Quaternions are represented as 16 bits where each component in (3) is represented by 4 bits then they are concatenated to each other starting from left to right that's to say from most significant 4 bits to the lowest significant 4 bits. An external clock is used in order to finish the operations that need too many cycles.

a) *Multiplication*

$$A*B \bmod q = (a1b1 - a2b2 - a3b3 - a4b4 \bmod q,\; a1b2 + a2b1 + a3b4 - a4b3 \bmod q,\; a1b3 - a2b4 + a3b1 + a4b2 \bmod q,\; a1b4 + a2b3 - a3b2 + a4b1 \bmod q) \quad (9)$$

Where $A*B \neq B*A$

b) *Addition*

$$A+B \bmod q = (a1+b1 \bmod q,\; a2+b2 \bmod q,\; a3+b3 \bmod q,\; a4+b4 \bmod q) \quad (10)$$

c) *Subtraction*

$$A\text{-}B \bmod q = (a1\text{-}b1 \bmod q,\; a2\text{-}b2 \bmod q,\; a3\text{-}b3 \bmod q,\; a4\text{-}b4 \bmod q) \quad (11)$$

d) *Norm*

$$norm(A) = (a1^2 + a2^2 + a3^2 + a4^2) \bmod q \quad (12)$$

With condition that norm(A) mod q $\neq 0$

TABLE I. COMPARISON OF DELAY, POWER, & AREA FOR ADDERS [5]

	Adder Name (16 bits)	Delay (in ns)	Power (in watts)	Area (LUTs,IOBs)
1	Kogge stone adder	6.700	1.211	71,50
2	Sparse Kogge stone adder	8.015	1.186	28,50
3	Spanning tree adder	6.667	1.186	30,49
4	Brent Kung adder	8.094	1.186	27,49

e) *Conjugate*

$$conj(A) = (a1, -a2, -a3, -a4) \bmod q \quad (13)$$

f) *Inverse*

$$A^{-1} = \frac{1}{norm(A)} * conj(A) \quad (14)$$

B. ABE system implementation

The encryption is done by implementing all the four algorithms of CP-ABE in the same module because this will save area and power. The quaternion operations that are used to implement the algorithms equations are called once and a control unit is done to control what operation should work at a rising clock edge in order to save area and power. Fig. 3 shows the CP-ABE block diagram that shows how the four algorithms are connected together. The encryption is designed for an infinite number of users. The attributes that are used in the system are (a, b and c). These attributes can be replaced with real attributes for a real life application.

1) Key Generation

The *LQ* in (2) and the *PK*s for each attribute in (3) are generated randomly beforehand with the following SP; [n = 5, r = 1, c = (a, b, c), q = 13]. The *LQ* is *{Q(1), Q(2), Q(3), Q(4), Q(5)}*, The *PK*s for each attribute are *V(a,1)* for *a*, *V(b,1)* for *b*, *V(c,1)* for *c*. The *MSK*s that are generated for each attribute are *E(a,1)* for *a*, *E(b,1)* for *b* and *E(c,1)* for *c*.

2) Policy

The policy that is used in the implementation is *((a AND b) OR c)*. This policy is done with the *MSK*s of each attribute with the attributes embedded in them so the policy is *k(x) = OR [E(a,1) AND E(b,1), E(c,1), x]*. So by substituting in (6) and in (15) we get the result of *k(x)*.

$$K(x) = AND[A,B] = A*B \bmod q \quad (15)$$

If $x = (E(a,1) * E(b,1))^{-1}$ then *k(x) = E(a,1) * E(b,1)* this *SK* is given to the users having the attributes (*a*,*b* or *a*,*b*,*c*), and if $x = E(c,1)^{-1}$ then *k(x) = E(c,1)* this *SK* is given to the users with attributes (*a*,*c* or *b*,*c* or *c*).

3) Encryption

The ciphertext *C(x)* is calculated by substituting in (7) with the value of the *k(x)* calculated from the policy and with the message (*M*) received.

4) Decryption

The decrypted message is calculated by substituting in (8). If $x = (E(a,1) * E(b,1))^{-1}$ then *A = E(b,1)*E(a,1)*, *B* = $E(b,1)^{-1} * E(a,1)^{-1}$, *C(x)* from the encryption algorithm, and *D = E(a,1)*E(b,1)* and if $x = E(c,1)^{-1}$ then *A = E(c,1)*, *B* = $E(c,1)^{-1}$, *C(x)* from the encryption algorithm, and *D = E(c,1)*.

TABLE II. COMPARISON OF DELAY, POWER, AND AREA FOR MULTIPLIERS [7]

	Parameters	[8] design	Booth algorithm	Vedic multiplier
1	Delay (ns)	37.668	46.74	27.14865
2	Power (mw)	29.34	151.34	0.1692
3	No. of transistors	4299	7296	14382

IV. IMPLEMENTATION RESULTS

This section presents the simulation results of the new hardware implementation of the Ciphertext-Policy Attribute-Based Encryption (CP-ABE) algorithm using 16 bits key size data path. The authors used a small key size is for testing purpose. All the results in this section were computed with MATLAB beforehand in order to check the validation of the values. FPGA Xilinx technology was used to simulate and synthesize the design using Xilinx ISE 13.1 to verify the encryption values. We targeted the Xilinx Spartan 3E FPGA hardware board for testing the encryption. The clock frequency of the board is 50 MHZ. Table III shows the results of the simulation. As shown it has 11.418 ns delay and dynamic power of about 18 mW. The area usage is also shown in the number of slices (2004/ 4656), the 4 input look up tables (LUTs) (3301/ 9312) and the input output banks (IOBs) (135/232). The total memory usage is about 312340 kilo bytes.

V. CONCLUSION AND FUTURE WORK

In this paper, the first hardware implementation of CP-ABE is proposed based on quaternion ring mathematical operations over finite fields. We discussed the design of the CP-ABE algorithm as well as the binary operations that we used to implement the quaternion ring operations. FPGA Xilinx technology was used to simulate and synthesise the design using Xilinx ISE 13.1 (Xilinx software development kit, Xilinx platform studio). We intend to implement ABE with larger key size and test this implementation with many IoT applications to calculate power usage.

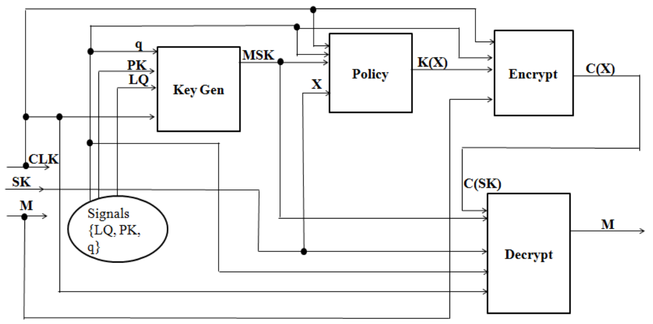

Fig. 3. CP-ABE block diagram

TABLE III. CP-ABE SYNTHESIS RESULTS

Parameters	CP-ABE
Delay (ns)	11.418
Slices (4656)	2004
4 input LUTs (9312)	3301
IOBs (232)	135
Dynamic power (mW)	18
Total memory usage (Kilo Bytes)	312340

VI. REFERENCES

[1] A. Sahai and B. Waters, "Fuzzy Identity-Based Encryption," *Lecture Notes in Computer Science Advances in Cryptology – EUROCRYPT* 2005, pp. 457–473.

[2] X. Wang, J. Zhang, E. Schooler, and M. Ion, "Performance evaluation of Attribute-Based Encryption: Toward data privacy in the IoT," *IEEE International Conference on Communications (ICC)*, pp. 725–730, 2014.

[3] M. Yagisawa, "Key distribution system and attribute-based encryption on non-commutative ring," *Cryptology ePrint Archive, Report 2012/24*, 2012.

[4] A. Zainal, Z. Othman, and M. Haron., " 4-bit Brent-Kung Parallel Prefix Adder Simulation Study Using Silvaco EDA tools," *International Journal of Simulation, Systems, Science and Technology*, vol. 13, pp. 51-59, 2009.

[5] S. Yezerla and B. Naik, "Design and estimation of delay, power and area for Parallel prefix adders," *Recent Advances in Engineering and Computational Sciences (RAECS)*, vol. 2, pp. 1–6, 2014.

[6] V. Agrawal, "FPGA Implementation of Low Power and High Speed Vedic Multiplier using Vedic Mathematics.," *IOSR Journal of VLSI and Signal Processing (IOSR-JVSP*, vol. 2, no. 5, p. 51–57, 2013.

[7] R. bathija, R. Meena, S. Sarkar, and R. Sahu, "Low Power High Speed 16x16 bit Multiplier using Vedic Mathematics," *International Journal of Computer Applications IJCA*, vol. 59, no. 6, p. 41–44, 2012.

[8] K. Gurumurthy, M. Prahalad, "Fast and power efficient 16x16 Array of Array multiplier using Vedic Multiplication," *2010 5th International Microsystems Packaging Assembly and Circuits Technology Conference*, pp. 1–4, 2009.