International Journal of Control and Automation
Vol. 11, No. 7 (2018), pp.97-108
http//dx.doi.org/10.14257/ijca.2018.11.7.08

The Design and Implementation of a Hardware Crypto Core for Securing Pervasive Devices

Dennis Agyemanh Nana Gookyi[1] and Kwangki Ryoo[*]

Department of Information and Communication Engineering, Hanbat National University, Daejeon, 34158, South Korea
[1]dennisgookyi@gmail.com, [*]kkryoo@gmail.com

Abstract

There is generally a high demand for lightweight cryptography for the fact that low-cost sensing devices used on the Internet-of-Things (IoT) platforms are required to be secured. This paper implements a lightweight crypto core that unifies a variety of algorithms while meeting the hardware area requirement of constrained devices. The cryptographic core consists of an Elliptic Curve Diffie-Hellman (ECDH) protocol for key exchange, a unified architecture of two encryption/decryption lightweight ciphers (PRESENT and a newly proposed algorithm) and a unified architecture of four Hopper-Blum (HB type) authentication protocols (HB, HB+, HBMP, and HBMP+). All the unified hardware architectures share resources in other to minimize the total hardware area of the core. The cryptographic core was designed using Verilog Hardware Description Language (HDL) and implemented on Virtex4 Field Programmable Gate Array (FPGA) which synthesized to 8644 slices at 164 MHz maximum clock frequency.

Keywords: *Lightweight Cryptography, IoT, Crypto Core, Hardware Architecture.*

1. Introduction

The end devices used for collecting data for platforms such as the IoT are physically available to adversaries and therefore have to be cryptographically secure. Though these devices such as sensors have small hardware footprint and low power, they are used in applications that require high throughput and low latency [1]. For the fact that most users are not willing to pay for security in their end devices [2] and also for the constrained nature of the devices, implementing standard cryptographic algorithms in IoT end devices is discouraged. Lightweight cryptography has therefore taken center stage in IoT constrained devices. Lightweight ciphers generally trade security for hardware area by using reduced key sizes and simple algorithms.

The objective of this paper is to design a lightweight hardware cryptographic core that implements a key sharing algorithm, encryption/decryption algorithms, and authentication algorithms. The key sharing algorithm is a 163-bit ECDH [3] used for generating a shared key for parties involved in communication. The key is authenticated using one of the four HB type lightweight authentication algorithms which include HB [4], HB+ [5], HBMP [6] and HBMP+ [7]. This is to provide proof that the key is authentic [8]. The authenticated key is used to encrypt or decrypt data using two lightweight encryption/decryption algorithms which include the PRESENT algorithm [9] and a recently proposed algorithm [10]. The lightweight encryption/decryption algorithms are unified to share resources such as a substitution box (sbox), permutation box (pbox) and a key scheduling logic. The authentication algorithms are also unified to share a pseudorandom number/bit (prng/prbg) generator, a dot product unit, a rotation unit, a key generation unit, and various logic gates. The resource sharing help to minimize the hardware area for the core

Received (April 8, 2018), Review Result (June 2, 2018), Accepted (June 4, 2018)

to fit in constrained devices. This paper is an expanded and elaborate version of our previous work [11].

The rest of this paper is organized as follows: Section 2 summarizes all the cryptographic algorithms, Section 3 describes the hardware architecture, Section 4 gives the hardware results and the paper concludes in Section 5.

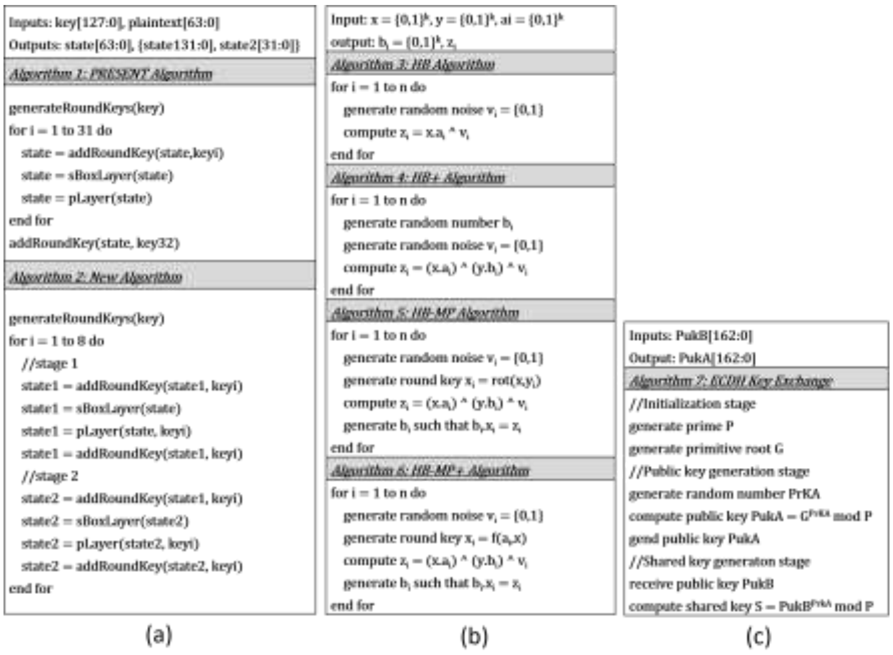
2. Cryptographic Algorithms Summary

The cryptographic core consists of three main types of algorithms: Lightweight encryption/decryption algorithms, lightweight authentication algorithms, and a lightweight key sharing protocol.

The top-level pseudocode for the encryption algorithms (PRESENT and a New algorithm) is shown in Figure 1 (a). PRESENT is a Substitution-Permutation (SP) network structure cipher that takes 31 rounds to encrypt a 64-bit block data using an 80/128 bit key while the new algorithm is a Feistel structure cipher that takes 8 rounds to encrypt a block of 64-bit data using a 128-bit key. Both ciphers use the same 4-bit input/output sbox while the pbox for the new algorithm is a one stage omega permutation network and that of PRESENT is a simple bit permutation network.

The HB type authentication protocols which include HB, HB+, HBMP and HBMP+ were proposed to be as simple as possible. The strength of HB type authentication protocols is generally based on the concept of Learning Parity with Noise (LPN) problem [12]. The inputs to the protocols include a key and a random number from the authenticator while the output is a computed value. All the protocols involve the generation of random numbers and random bits, dot product (.) calculation and XORing (^) as shown in Figure 1 (b).

The ECDH protocol computes a shared key by using the communicating partner's public key and a generated private key. The protocol also generates a public key which is sent to any communicating party. The pseudocode for the protocol is shown in Figure 1 (c).



```
Inputs: key[127:0], plaintext[63:0]
Outputs: state[63:0], {state131:0], state2[31:0]}
Algorithm 1: PRESENT Algorithm
generateRoundKeys(key)
for i = 1 to 31 do
   state = addRoundKey(state,keyi)
   state = sBoxLayer(state)
   state = pLayer(state)
end for
addRoundKey(state, key32)
Algorithm 2: New Algorithm
generateRoundKeys(key)
for i = 1 to 8 do
   //stage 1
   state1 = addRoundKey(state1, keyi)
   state1 = sBoxLayer(state)
   state1 = pLayer(state, keyi)
   state1 = addRoundKey(state1, keyi)
   //stage 2
   state2 = addRoundKey(state1, keyi)
   state2 = sBoxLayer(state2)
   state2 = pLayer(state2, keyi)
   state2 = addRoundKey(state2, keyi)
end for
```

(a)

```
Input: $x = \{0,1\}^k$, $y = \{0,1\}^k$, $ai = \{0,1\}^k$
output: $b_i = \{0,1\}^k$, $z_i$
Algorithm 3: HB Algorithm
for i = 1 to n do
   generate random noise $v_i = \{0,1\}$
   compute $z_i = x.a_i$ ^ $v_i$
end for
Algorithm 4: HB+ Algorithm
for i = 1 to n do
   generate random number $b_i$
   generate random noise $v_i = \{0,1\}$
   compute $z_i = (x.a_i)$ ^ $(y.b_i)$ ^ $v_i$
end for
Algorithm 5: HB-MP Algorithm
for i = 1 to n do
   generate random noise $v_i = \{0,1\}$
   generate round key $x_i = rot(x,y_i)$
   compute $z_i = (x.a_i)$ ^ $(y.b_i)$ ^ $v_i$
   generate $b_i$ such that $b_i.x_i = z_i$
end for
Algorithm 6: HB-MP+ Algorithm
for i = 1 to n do
   generate random noise $v_i = \{0,1\}$
   generate round key $x_i = f(a_i,x)$
   compute $z_i = (x.a_i)$ ^ $(y.b_i)$ ^ $v_i$
   generate $b_i$ such that $b_i.x_i = z_i$
end for
```

(b)

```
Inputs: PukB[162:0]
Output: PukA[162:0]
Algorithm 7: ECDH Key Exchange
//Initialization stage
generate prime P
generate primitive root G
//Public key generation stage
generate random number PrKA
compute public key PukA = $G^{PrKA}$ mod P
gend public key PukA
//Shared key generaton stage
receive public key PukB
compute shared key S = $PukB^{PrKA}$ mod P
```

(c)

Figure 1. Crypto Core Algorithms (a) Encryption (b) Authentication (c) Key Exchange

3. Hardware Architectures of Crypto Core Algorithms

The hardware architectures consist of a unified design of the encryption algorithms, a unified design of the authentication algorithms and lastly the key sharing algorithm.

3.1. Hardware Architecture of a Unified Encryption/Decryption Algorithms

Figure 2 shows the datapath of encryption/decryption algorithm for PRESENT and the new cipher.

From Figure 2, The *INPUT DATA* and the *OUTPUT DATA* which are 64-bit could be either the plaintext or a ciphertext since the core has both encryption and decryption capabilities. *Reg1_1* and *Reg1_2* are two 32-bit registers used for storing intermediate computational data after which the data from the registers are concatenated and stored in 64-bit *Reg2*. Two 32-bit input/output sboxes (*SBOX1* and *SBOX2*) are designed by combining eight 4-bit input/output sboxes. The sboxes combines both substitution and inverse substitution used by the encryption and decryption algorithms respectively. The implementation of the two sboxes was done to reduce latency and increase throughput while hardware area is slightly increased. Four *XOR* gates are used with a number of multiplexers for routing the appropriate data based on select signals. From the figure, the green long dash lines indicate the path of the new algorithm, the red shot dash lines indicate the path of the PRESENT algorithm and the dark straight lines indicate the path common to both algorithms.

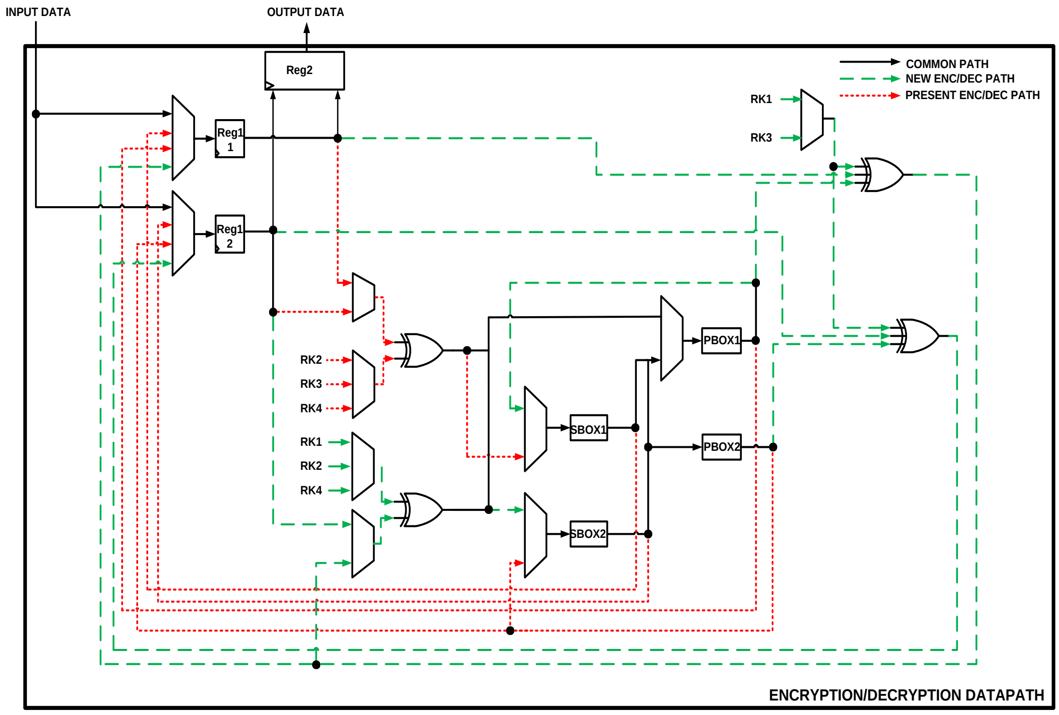
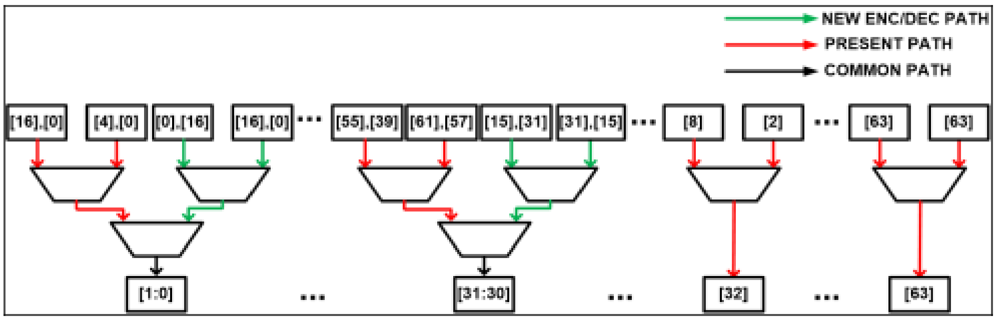
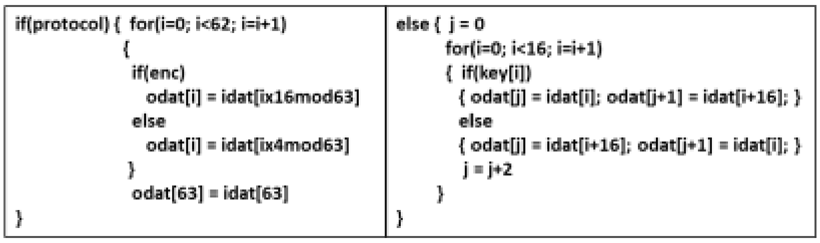


Figure 2. Hardware Architecture of Unified Encryption Algorithms

The pbox for PRESENT algorithm involves only bit permutation and therefore consume no hardware area while that of the newly proposed cipher uses a one stage omega permutation network structure that makes the output of the pbox less predictable without having access to the cipher key. The pbox of the two algorithms are unified and the datapath instantiates *PBOX1* and *PBOX2*. The hardware structure and pseudocode for the pbox are shown in Figure 3 (a) and (b) respectively. The input takes a 64-bit *idat* and produces a 64-bit *odat* as the output. The newly proposed algorithm uses only the first 32 bit of *odat* while PRESENT algorithm uses all the 64-bit *odat*.



(a)



```
if(protocol) {  for(i=0; i<62; i=i+1)
                {
                  if(enc)
                    odat[i] = idat[ix16mod63]
                  else
                    odat[i] = idat[ix4mod63]
                }
                  odat[63] = idat[63]
}
```

```
else {  j = 0
        for(i=0; i<16; i=i+1)
        {  if(key[i])
          { odat[j] = idat[i]; odat[j+1] = idat[i+16]; }
          else
          { odat[j] = idat[i+16]; odat[j+1] = idat[i]; }
           j = j+2
        }
}
```

(b)

Figure 3. (a) Pbox Hardware Structure (b) Pbox Pseudocode

The architecture of the key scheduling for both algorithms is shown in Figure 4. From the figure, the key schedule algorithm takes in a 128-bit *KEY_IN* and generates a 128-bit *KEY_OUT*. For PRESENT algorithm it involves a rotation of the intermediate key by 61-bits for encryption and 19 bits for decryption before part of it passes through a 4-bit *SBOX* and another part *XORed* with the *COUNTER*. The new algorithm uses the same key schedule for both encryption and decryption which involves only a 25-bit rotation.

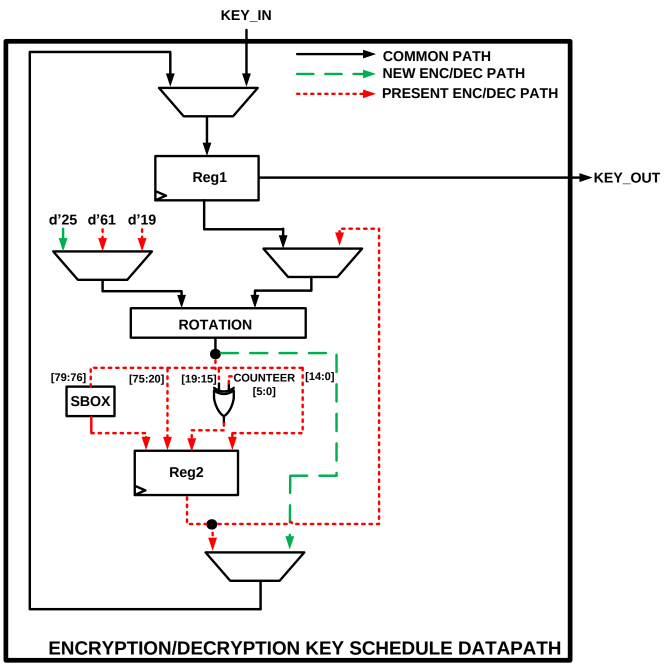


Figure 4. Hardware Architecture of Unified Key Schedule Algorithm

3.2. Hardware Architecture of Unified Authentication Algorithms

The hardware architecture of the unified authentication algorithms which include HB, HB+, HBMP, and HBMP+ is shown in Figure 5. The components are used by both the authenticator and the "authenticatee" that is the reader and tag respectively. The inputs consist of 64-bit *TAG_INPUT* which is a computed value from the tag only used by the reader, a 64-bit *KEY1*, a 64-bit *KEY2* and a 64-bit *RAN_NUM_IN*. The outputs consist of a 64-bit *RAN_NUM_OUT*, a 64-bit *TAG_OUTPUT* and a 1-bit *TAG_VALID* used by only the reader.

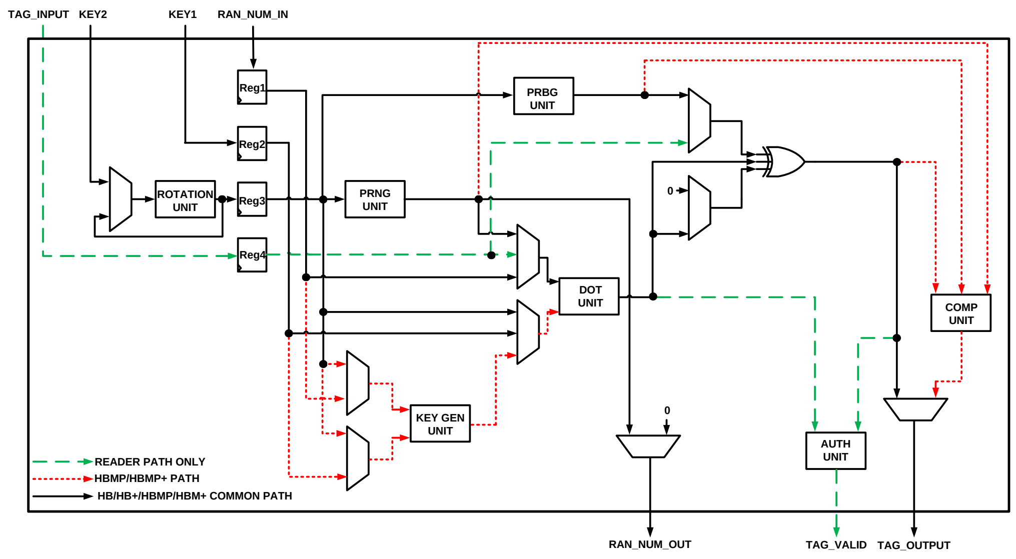
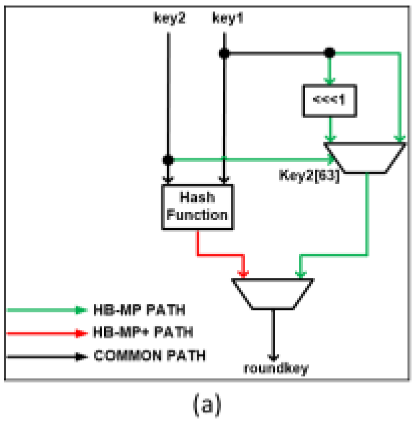
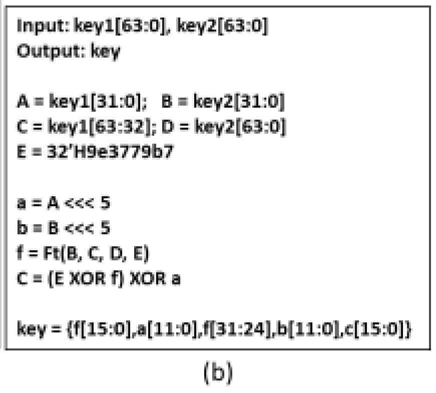


Figure 5. Hardware Architecture of Unified Authentication Algorithms

The internal modules consist of *PRNG* and *PRBG* units for generating pseudorandom numbers and bits using linear feedback shift register method. The *DOT* unit computes the dot product of the keys and the random numbers. The *KEN GEN* and *COMP* units are used by HBMP and HBMP+ only to generate round keys (using simple rotation for HBMP and a one-way hash function for HBMP+) and compute the output (using a comparator) respectively. The *AUTH* unit is used by the reader to authenticate the tag. The hardware architecture for the *KEN GEN* unit is shown in Figure 6 (a). The structure takes *key1* and *key2* as its input and produces *roundkey* as the output. The *Hash Function* [13] is only used by HBMP++ and its pseudocode is shown in Figure 6 (b).





```
Input: key1[63:0], key2[63:0]
Output: key

A = key1[31:0];  B = key2[31:0]
C = key1[63:32]; D = key2[63:0]
E = 32'H9e3779b7

a = A <<< 5
b = B <<< 5
f = Ft(B, C, D, E)
C = (E XOR f) XOR a

key = {f[15:0],a[11:0],f[31:24],b[11:0],c[15:0]}
```

(b)

Figure 6. (a) KEN GEN Hardware Structure (b) Pseudocode for Hash Function

3.3. Hardware Architecture of Point Multiplication for ECDH Algorithm

The main computation involved in ECDH algorithm is the Scaler Point Multiplication (SPM) which involves repeated point additions and point doublings. The Montgomery Ladder [14] is among the widely used methods for SPM because it involves fewer operations. This paper implements the Montgomery Ladder Method (MLM) over projective fields using one *MULTIPLY* unit, three *SQUARER* units and one *DIVIDE* units as shown in Figure 7. The datapath takes two 163-bit inputs *XP* and *YP* and generates two 163-bit outputs *YQ* and *XQ*. The *MULTIPLY* unit implements serial multiplication as described by [15] while the *DIVIDE* unit was proposed by [16]. The *SQUARER* unit is implemented using only XOR gates.

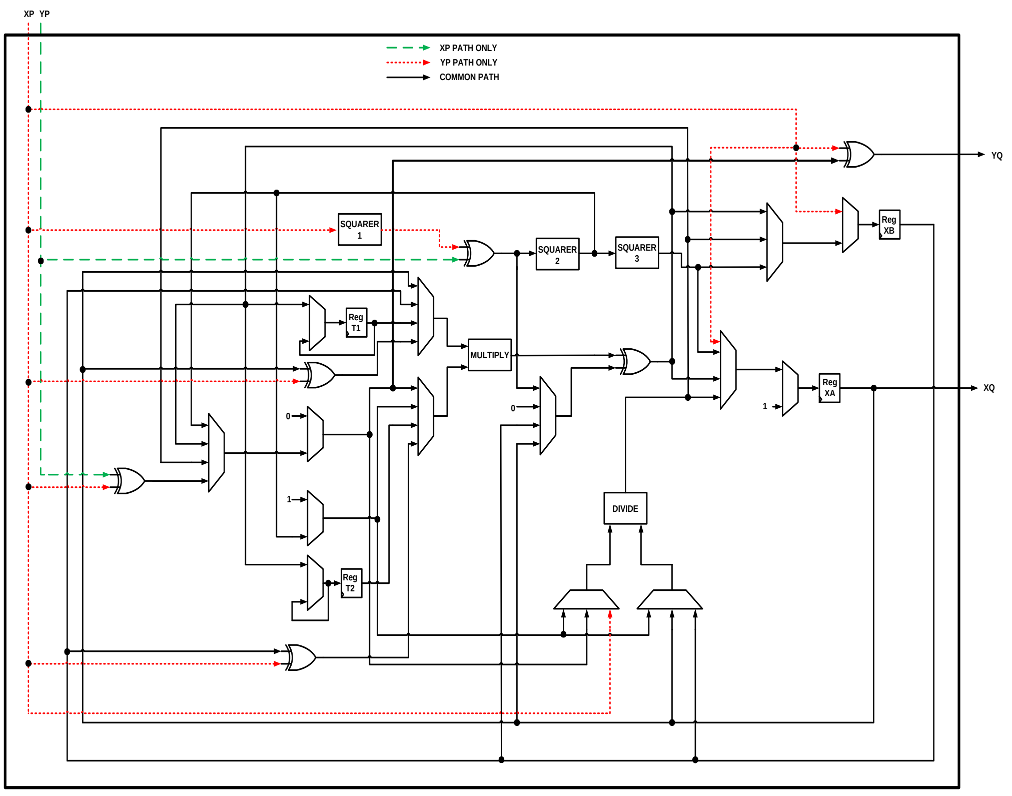


Figure 7. Hardware Architecture of Scaler Point Multiplication for ECDH

3.4. Hardware Architecture of the Crypto Core

The overall hardware architecture of the crypto core is shown in Figure 8. The core is capable of generating a shared key, authenticating the shared key and using the shared key to encrypt or decrypt a block of data. The *CONS* unit provides the seed variable for both *PRNG* and *PRBG* units. The seed variable is 64-bit for *PRNG* and 4-bit for *PRBG*. The *CONS* unit also generates constants such as the ECC curve equation, the Y-coordinate and the X-coordinate for the *ECDH* unit. The *PRBG* unit generates a pseudorandom bit using a seed obtained from the *key*. The random bit serves as noise and is used by the *AUTH* unit to mask data. The *PRNG* unit generates pseudorandom numbers using seed from the *key* for both the *AUTH* and *ECDH* units. The random number for the *AUTH* unit is a random challenge that is mixed with the key while the random number for the *ECDH* unit serves as the private key used in the computation of the public key and the shared key. The *ECDH* unit takes as its inputs, some constants, and the private key. The *ECDH* unit is responsible for generating a public key and also the shared key which is authenticated by the *AUTH* unit. The authenticated key is used by the *ENC_DEC* unit to encrypt or decrypt data. This is achieved by receiving a 128-bit authenticated key from the *ECDH* unit and a block of 64-bit data from the *DATA INTERFACE*. The *CONTROL SIGNAL INTERFACE* and the *DATA INTERFACE* generates control and data signals respectively.

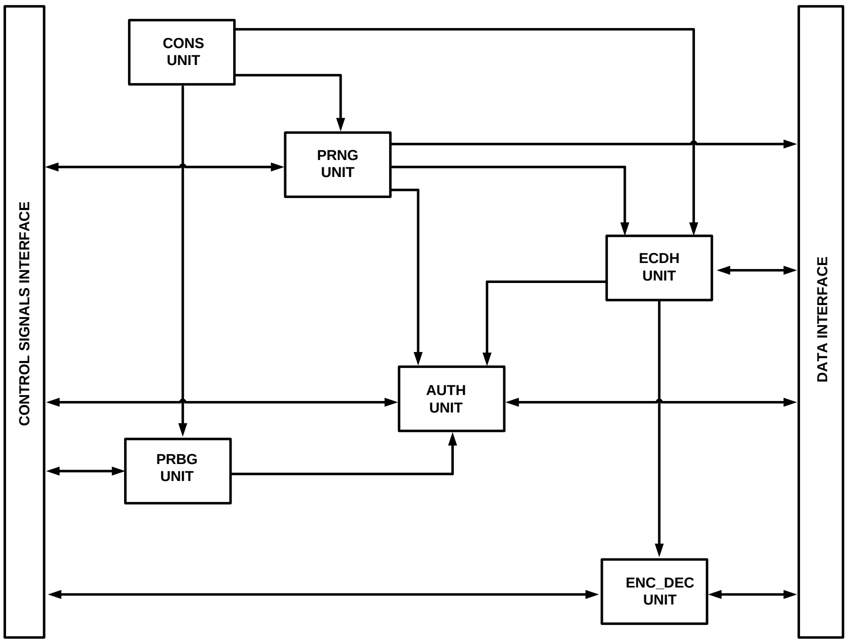


Figure 8. Hardware Architecture of Crypto Core

4. Hardware Experimental Results

The hardware architecture of the cryptographic core was designed using Verilog Hardware Description Language (HDL) [17]. The software used for the design is the Xilinx ISE 14.7. Modelsim SE-64 10.1c software was used for the simulation of the design. For the purpose of synthesis, Xilinx Virtex4 XC4VLX80 FPGA [18] device was used. Here we discuss the simulation and synthesis results.

4.1. Simulation Results

The simulation flowchart is shown in Figure 9. At the start, the core receives a 163-bit public key from the parties involved in the communication. The 163-bit private key which is a random number is then generated using a pseudorandom number generator. The ECDH receives the communicating partner's 163-bit public key and 163-bit private key which is used to generate the public key. The public key is then used to generate the shared key of which the communicating partner also generates using the public key of the sender. The public key is sent to the communicating parties. The key that is shared is then authenticated by choosing one of the authentication algorithms (HB, HB+, HBMP or HBMP+). Given that the authentication is not valid, the process is terminated. If the authentication process is valid then an encryption/decryption algorithm (PRESENT or the New algorithm) is selected and used to decrypt received ciphertext or encrypt plaintext. The encrypted data is then sent to the appropriate receiver and the process is terminated.

The simulation results waveform is shown in Figure 10. The clock frequency in the testbench is set to 100*MHz*. From the Figure, the shared key generation which uses the ECDH algorithm take about 2762*us* to generate the shared key. After the key generation, the key is authenticated by choosing HB authentication protocol which takes about 21*us* to complete. The authenticated key is then used to encrypt a 64-bit plaintext by choosing PRESENT encryption/decryption algorithm which takes approximately 1*us* to generate the 64-bit ciphertext. Overall, the whole key generation, authentication and encryption/decryption process takes about 2.7*ms* to complete. One of best known authenticated procedure for RFID tags is designed by [19] using AES algorithm. The paper used a challenge-response protocol that takes at least 18*ms* to authenticate each tag. With our overall simulation time of 2.7*ms*, this could be of interest in the RFID world.

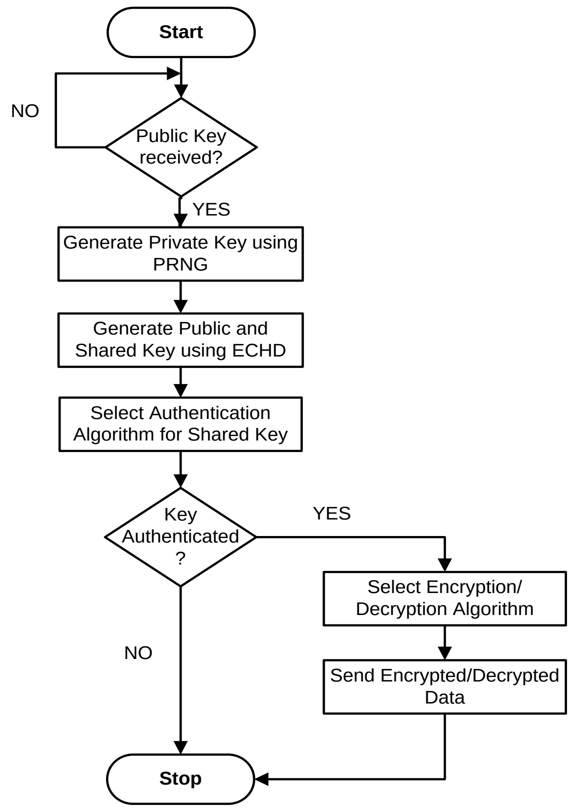

Start
NO
Public Key received?
YES
Generate Private Key using PRNG
Generate Public and Shared Key using ECHD
Select Authentication Algorithm for Shared Key
Key Authenticated ?
YES
Select Encryption/ Decryption Algorithm
NO
Send Encrypted/Decrypted Data
Stop

Figure 9. Simulation Flow Chart Diagram

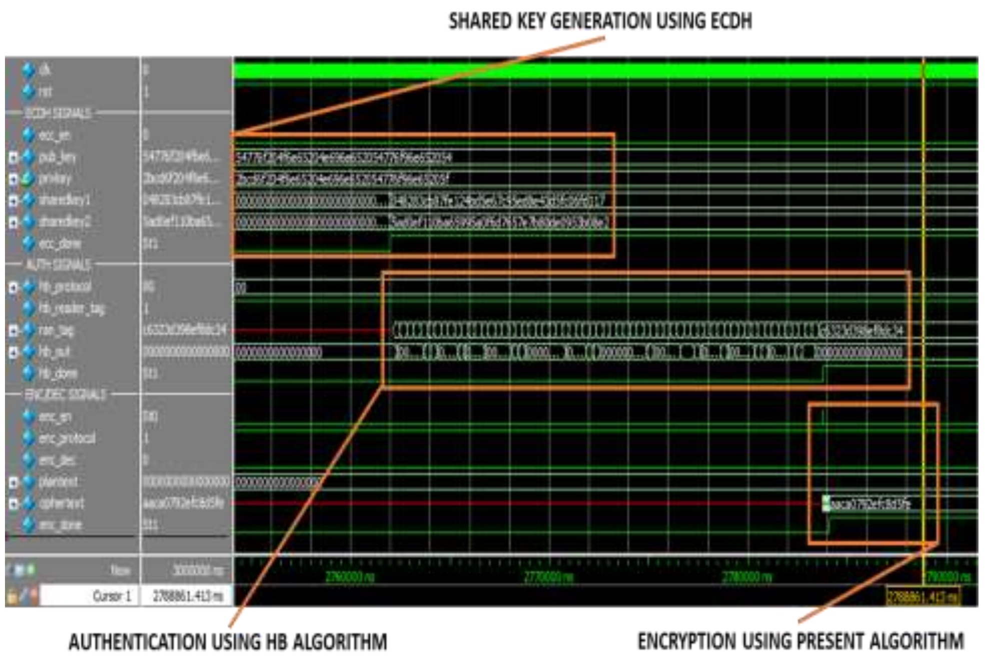

SHARED KEY GENERATION USING ECDH
AUTHENTICATION USING HB ALGORITHM
ENCRYPTION USING PRESENT ALGORITHM

Figure 10. Simulation Waveform Diagram

4.1. Synthesis Results

The hardware synthesis result is shown in Table 1. The tabulated results are given in terms of total hardware area measured in slices, total clock cycles, maximum frequency measured in *MHz*, throughput measured in *Mbps* and efficiency of the overall core which is measured in *Mbps/slice*. Since it is difficult to find a cryptographic core that implements the same algorithms as our design, we decided to compare our core to [20] and [21] in terms of hardware area, frequency, throughput (obtained by multiplying the frequency and number of bits and dividing by the number of cycles) and efficiency (obtained by taking the throughput and dividing by the hardware area). Our proposed core is about 2 and 5 times smaller in terms of hardware area (slices) as compared to the proposed cores by [20] and [21] respectively. With regards to the maximum frequency and overall efficiency, our proposed core outperforms both [20] and [21].

Table 1. Hardware Synthesis Results Comparison

Ref.	Algorithms	Area (Slices)	Clock Cycles	Frequency (MHz)	Throughput (Mbps)	Efficiency (Mbps/slice)
[20]	SHA	18504	32	73	584	0.037
	ECC		380		31	
	GRAIN		128		73	
[21]	AES	43598	86	100	148	0.013
	RSA		3193000		0.032	
	ECC		36846		0.442	
	SHA		120		426.66	
Design	PRESENT	8644	31	164	338.5	0.193
	NEW		8		1312	
	HB		1920		5.47	
	HB+		2952		3.56	
	HB-MP		4214		2.49	
	HB-MP+		4444		2.36	
	ECDH		49580		0.54	

6. Conclusion

This paper implemented a hardware cryptographic core that is capable of generating a shared key (using ECDH key sharing protocol), authenticating the key (using either HB, HB+, HBMP or HBMP+ lightweight authentication algorithms) and using the shared key for encryption/decryption (using either PRESENT or a New encryption/decryption algorithm). The authentication algorithms and the encryption/decryption algorithm are unified into two separate modules. The unified structures resort to the sharing of resources such as pseudorandom number and bit generators, rotation units, key schedule algorithms, dot product unit and other logic gates in other to meet the maximum area requirement of a lightweight cryptographic core. The core synthesized to 8644 slices at 164 MHz maximum clock frequency using a Virtex4 FPGA device. In the simulation, the overall procedure of generating a shared key, authenticating the shared key and encryption/decryption of plaintexts or ciphertexts take approximately 2.7*ms*. The proposed core is suitable for systems that require low hardware resources, medium security level and high throughput such as RFID applications.

In the future, we will be looking at implementing the cryptographic core in a wireless System-on-Chip (SoC) application environment using a lightweight synthesizable processor that implements the RISC V Instruction Set Architecture (ISA).

Acknowledgments

This paper is a revised and expanded version of a paper entitled “A Hardware Cryptographic Core for Securing IoT Constrained Devices” presented at The Fourth Asia Workshop on IT Convergence KIICE 2018 (AWITC 2018) held at Busan, South Korea on February 9, 2018.

This research was supported by the MSI (Ministry of Science, ICT and Future Planning), Korea, under the Global IT Talent support program (IITP-2017-0-01681) supervised by the IITP (Institute for Information and Communication Technology Promotion).

References

[1] A. Biryukov and L. Perrin, “State of the Art in Lightweight Symmetric Cryptography”, Cryptology ePrint Archive Report 2017/511, **(2017)**.

[2] B. Collin, “Cryptography in the Cloud: Advances and Challenges”, Journal of Information and Communication Convergence Engineering, vol. 11, no. 1, **(2013)**, pp. 17-23.

[3] M. Smid and J. D. E. Barker, “Recommendation for Pairwise Key Establishment Schemes Using Discrete Logarithm Cryptography”, Federal Information Processing Standards Special Publication 800-56A., **(2007)**.

[4] N. J. Hopper and M. Blum, “Secure Human Identification Protocols”, Lecture Notes in Computer Science, vol. 2248, **(2001)**, pp. 52-66.

[5] A. Juels and S. A. Weis, “Authenticating Pervasive Devices with Human Protocols”, Lecture Notes in Computer Science, vol. 3621, **(2005)**, pp. 293-308.

[6] J. Munilla and A. Peinado, “HB-MP: A Further Step in the HB-Family of Lightweight Authentication Protocols”, Computer Networks, vol. 51, no. 9, **(2009)**, pp. 2262-2267.

[7] X. Leng, K. Mayes and K. Markantonakis, “HB-MP+ Protocol: An Improvement on the HB-MP Protocol”, IEEE International Conference on RFID, Las Vegas, USA, **(2008)** April 16-17, pp. 118-124.

[8] A. Vegendla, H. Seo, D. Lee and H. Kim, “Implementation of an RFID Key Management System for DASH7”, Journal of Information and Communication Convergence Engineering, vol. 12, no. 1, **(2014)**, pp. 19-25.

[9] A. Bogdanov, C. Paar and A. Poschmann, “PRESENT: An Ultra-Lightweight Block Cipher”, Lecture Notes in Computer Science, vol. 4727, **(2013)**, pp. 450-466.

[10] D. A. N. Gookyi, S. Park and K. Ryoo, “The Efficient Hardware Design of a New Lightweight Block Cipher”, International Journal of Control and Automation., vol. 10, no. 1, **(2017)**, pp. 431-440.

[11] D. A. N. Gookyi and K. Ryoo, “A Hardware Cryptographic Core for Securing IoT Constrained Devices”, Proceedings of the Fourth Asia Workshop on IT Convergence of KIICE (AWITC 2018), Busan, South Korea, (2018), February 9.

[12] E. Levieil and P. A. Fouque, “An Improved LPN Algorithm”, Lecture Notes in Compute Science, vol. 4116, **(2006)**, pp. 348-359.

[13] D. A. N. Gookyi and K. Ryoo, “Hardware Design and Analysis of HB Type Lightweight Authentication Protocols for Pervasive Devices”, International Journal of Pure and Applied Mathematics, vol. 118, no. 19, (2018), pp. 1927-1946.

[14] P. L. Montgomery, “Speeding the Pollard and Elliptic Curve Methods of Factorization”, Mathematics of Computation, vol. 48, no. 177, **(1987)**, pp. 243-264.

[15] G. D. Sutter, J. L. Imaa and J. P. Deschamps, “Hardware Implementation of Finite Field Arithmetic”, McGraw-Hill, New York, **(2009)**.

[16] C. P. Hong and C. H. Kim, “High Speed Division Architecture for GF(2m)”, Electronics Letters., vol. 38, no. 15, **(2002)**, pp. 835-836.

[17] M. B. Lin, Editor, “Digital System Designs and Practices: Using Verilog HDL and FPGAs”, John Wiley & Sons (Asia) Pte Ltd, Singapore, **(2008)**.

[18] Xilinx Inc, “Virtex-4 Family Overview”, Product Specification DS112 V3.1 (www.xilinx.com), (2010), pp. 1-9.

[19] M. Feldhofer, S. Dominikus and J. Wolkerstorfer, “Strong Authentication for RFID Systems using AES Algorithm”, Lecture Notes in Computer Science, vol. 3156, **(2004)**, pp. 357-270.

[20] M. Machout, C. Massoud, M. Zeighid and A. Sghaier, “Design and Implementation of Low Area/Power Elliptic Curve Digital Signature Hardware Core”, Multidisciplinary Digital Publishing Institute - Electronics, vol. 6, no. 2, **(2017)**, pp. 1-23.

[21] M. K. Hani, H. Y. Wen and A. Paniandi, “Design and Implementation of a Private and Public Key Crypto Processor for Next-Generation IT Security Applications”, Malaysian Journal of Computer Science, vol. 19, no. 1, **(2006)**, pp. 29-45.

Authors

Dennis Agyemanh Nana Gookyi received a BSC Degree in Computer Engineering from Kwame Nkrumah University of Science and Technology, Ghana, in 2013 and MENG Degree in Information and communication engineering from Hanbat National University, South Korea in 2017 where he is currently a Ph.D. candidate. His research interests include SoC Design and Verification Platforms, Lightweight Cryptography and SoC Design for Security.

Kwangki Ryoo received BS, MS and Ph.D. Degrees in Electronic Engineering from Hanyang University, Korea in 1986, 1988 and 2000 respectively. From 1991 to 1994, he was an Assistant Professor at the KMA (Korea Military Academy) in South Korea. From 2000 to 2002, he worked as a Senior Researcher at ETRI (Electronics and Telecommunications Research Institutes), Korea. From 2010 to 2011, he was a Visiting Scholar at the University of Texas in Dallas. Since 2003, he has been a Professor at Hanbat National University, Daejeon Korea. His research interests include Engineering Education, SoC Platform Design and Verification, Image Signal Processing and Multimedia Codec Design, and SoC Design for Security.