The Hardware Design and Implementation of a Key Exchange Protocol for Low-cost IoT Devices

Dennis Agyemanh Nana Gookyi, Kwangki Ryoo

Abstract: ***The General Data Protection Regulation (GDPR) which was enforced in May 2018 clearly stated that the protection of data by organizations is a mandatory task. Protecting or securing data on data collecting and sensing devices used in the Internet-of-Things (IoT) platform is a challenge for the fact that the devices are resource-constrained in terms of operation frequency, hardware area, computational complexity, and power consumption. The first step to securing data on low-cost IoT devices is to generate keys for subsequent encryption and authentication. This paper, therefore, proposes and implements a lightweight key exchange protocol with the capability of authenticating the generated key without the need for public-key cryptography. The protocol is meant to be simple and make use of minimal hardware resources. It uses components such as the pseudorandom number and bit generators, dot product, XOR gates, shift registers and basic logic gates making it very resource-efficient. The hardware architecture of the protocol was implemented using Verilog Hardware Description Language (HDL) and synthesized using Xilinx ISE 14.7 software which includes XPower Analyzer for power estimation. The protocol was tested on a Field Programmable Gate Array (FPGA) board with a synthesizable Reduced Instruction Set Computer Five (RISC-V) processor core. The synthesis and simulation results which include area, maximum frequency, latency, and power consumption show that the protocol is suitable for IoT low-cost devices as compared to standard public-key primitives.***

Keywords: Hardware Design, IoT, Key Exchange Protocol, Low-cost Devices, RISC-V, Synthesizable Processor.

I. INTRODUCTION

The General Data Protection Regulation (GDPR) [1] is an enforced regulation in European Union Law that was proposed in April 2016 and subsequently implemented and enforced in May 2018. This regulation is a binding law that ensures that organizations, agencies, and individuals that collect personal data of citizens of any European country are liable when the data are breached. To summarize, the regulation enforces data collectors to transforms personal data in such a way that the visible data cannot be traced to an individual without additional data set. This, in simple terms, means that data collected must be encrypted and decryption should be rendered impossible without the keys used for encryption. The regulation goes further to state that encryption/decryption of personal data must be done locally and the keys used must remain only in the hands of the data owner.

With the enforcement of the GDPR, it is important for Internet-of-Things (IoT) application developers to comply and secure data stored on their devices. The lowest abstraction of IoT platforms consists mainly of data collecting and sensing devices [2]. These devices are basically constrained in terms of operational frequency, computational complexity, hardware area, and power consumption. These constrains make it challenging when implementing security protocols for low-cost devices used at the end nodes of the IoT platform. The first step in securing data on devices is implementing a key exchange protocol to be used for subsequent encryption/decryption of data and authentication. A lot of key exchange protocols involve the use of asymmetric algorithms like the Rivest-Shamir-Adleman (RSA) algorithm [3] and the Elliptic Curve Diffie Hellman (ECDH) algorithm [4]. The execution of many public key algorithms involves the use of complex mathematical operations like modular exponentiation, division, multiplication, and inversion which consume a lot of hardware resources and are therefore not suitable for low-cost devices.

Recently, lightweight key exchange protocols have been proposed for constrained devices like RFID tags. SPAKE [5] is a lightweight protocol that generates keys for contactless applications. The protocol is based on a version of the RSA [6] public-key algorithm and a block cipher like the Advanced Encryption Standard (AES) [7]. Using a public key and a block cipher for key exchange is infeasible for ultra-constrained devices like the RFID tag. IwAKE [8] is an authenticated key exchange protocol for Device-to-Device communication. This protocol makes use of SipHash algorithm [9] as a keyed hash function. The SipHash algorithm requires 3700 Gate Equivalent (GE) for compact implementation which might not fit into a low area device. Session-HB [10] is a protocol that improves the HB+ [11] protocol by generating initial session keys. The protocol involves two stages with the first stage using random number generators and a MixBits [12] function to generate the session key. The second stage uses an algorithm that involves comparators, addition, multiplication, and division for authentication.

Revised Manuscript Received on July 22, 2019.

Dennis Agyemanh Nana Gookyi, Department of Information and Communication Engineering, Hanbat National University, Daejeon, South Korea. Email: dennisgookyi@gmail.com

Kwangki Ryoo*, Department of Information and Communication Engineering, Hanbat National University, Daejeon, South Korea. Email: kkryoo@gmail.com

The use of multiplication and division could be a deal-breaker for implementation on resource-constrained devices.

The purpose of this work is to design and implement an efficient lightweight key exchange protocol that meets the requirement of low-cost hardware devices used at the end nodes of IoT platforms. The protocol is based on the security and simplicity of the HB family of protocols. The protocol has the ability to generate a shared key and authenticate the shared parameters used for the generation of the key. Components such as pseudorandom number/bits generators, dot product, shift registers, and basic logic gates are used in the key exchange protocol making it extremely lightweight.

The contributions of this paper are as follows: Section II discusses the specific requirements for low-cost IoT devices, Section III delves into the proposed key exchange protocol and its security, Section IV illustrates the hardware architecture and simulation results of the proposed key exchange protocol, Section V tests the hardware architecture on a Field Programmable Gate Array (FPGA) board with a synthesizable Reduced Instruction Set Computer Five (RISC-V) processor core [13] and lastly the conclusion and future directions are discussed in Section VI.

II. SPECIFIC REQUIREMENTS FOR LOW-COST DEVICES

IoT end-to-end devices are usually ultra-constrained to cut down the cost of mass implementation. Nonetheless, there are specific requirements to meet when designing key establishment protocols for IoT devices. As stated by [14], the four requirements for a key exchange protocol include: the algorithm should be bidirectional, the devices may or may not require pre-shared parameters, the algorithm should be scalable and efficient.

Expanding on the efficiency of the algorithm, [15, 16] gave a clear hardware resource requirement of low-cost IoT devices. Some of the requirements include: the hardware gate count should be less than 2000 gates, latency should be less than 5 milliseconds and power consumption should be less than 10 microwatts. Recent hardware designs of RSA and ECDH by [17, 18, 19] show that the gate count is usually above 50000 gates and the latency is above 50 milliseconds for both algorithms as shown in Table I. This makes standard public-key protocols not suitable for low-cost devices.

Table- I: Summary of Hardware Resource Requirements

Metric	RSA	ECDH	Low-cost Devices Specification
Operations	Modular exponentiation	Modular multiplication, division, inversion	Basic logic gates
Gates	>200,000	>50,000	<2000
Time (ms)	>800	>50	<5
Power (mW)	>100	>100	<10

III. PROPOSED LIGHTWEIGHT KEY EXCHANGE PROTOCOL

It is evident that a lightweight key exchange protocol is very much needed for generating shared keys among devices. The proposed lightweight key exchange scheme is based on an authentication set of algorithms known as the HB family of authentication protocols [20]. These protocols are used for authenticating ultra-lightweight devices that require some form of security. The authentication algorithms exploit the Learning Parity in the Presence of Noise (LPN) problem [21] to enable successful authentication. Due to the difficulty of solving the LPN problem, the proposed lightweight key exchange protocol relies on the LPN problem for its security.

A simple description of the LPN problem is as follows: let A be a random challenge with the size of a qxk matrix and let v be a random noise bit of size q-bit with the probability of generating a 1 (η) in the set of (0, 0.5). The magnitude of v is given in (1) and let x be a key-value with the size of k-bit. When given A, η, and z (calculated in (2)), find a k-bit value x' that makes (3) true.

$$|v| = \eta q \tag{1}$$

$$z = (A \cdot x) \oplus v \tag{2}$$

$$|(A \cdot x') \oplus v| = \eta q \tag{3}$$

A practical example of the LPN problem is illustrated in Fig. 1. From the figure, side A and side B share a common key value x and noise parameter η. Side A generates a random number A and sends it to side B. Side B calculates the value z using (2) and sends it to side A. Side A confirms that side B share the same x value using (3). When an adversary captures A and z, the challenge is to find x' that makes (3) true and from the example, the probability is exactly 0.5 which makes it difficult to guess the actual x value.

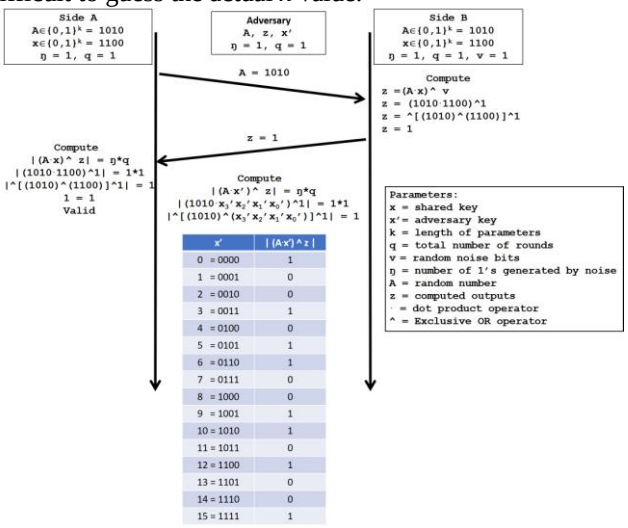


Fig 1. A Practical Example of the LPN Problem

A. Proposed Key Exchange Protocol

The proposed lightweight key exchange protocol is based on the HB family of authentication protocols whose security is dependent on the LPN problem. To generate a shared key between two sides, both of them should have pre-shared parameters x (a k-bit key), y (a k-bit key), η (noise parameter) and a counter (cnt) value.

The key generation takes two simple steps.

The first step is as follows: Side A starts by generating a 1-bit random noise value v_a and a k-bit random number value a. Side A then computes z_a using (4) and sends a together with z_a to side B. Side B computes v_a using (5) and increments the *cnt* value if v_a is equivalent to 1. Side B then generates a 1-bit noise value v_b, a k-bit random number value b and computes z_b using (6). Side B then sends z_b along with b to side A. Side A computes v_b using (7) and if it is equal to 1, the *cnt* value is incremented. Both side A and side B compute the shared key value v_{ab} using (8). This process is repeated for a number of rounds (n).

$$z_a = (x \cdot a) \oplus v_a \tag{4}$$

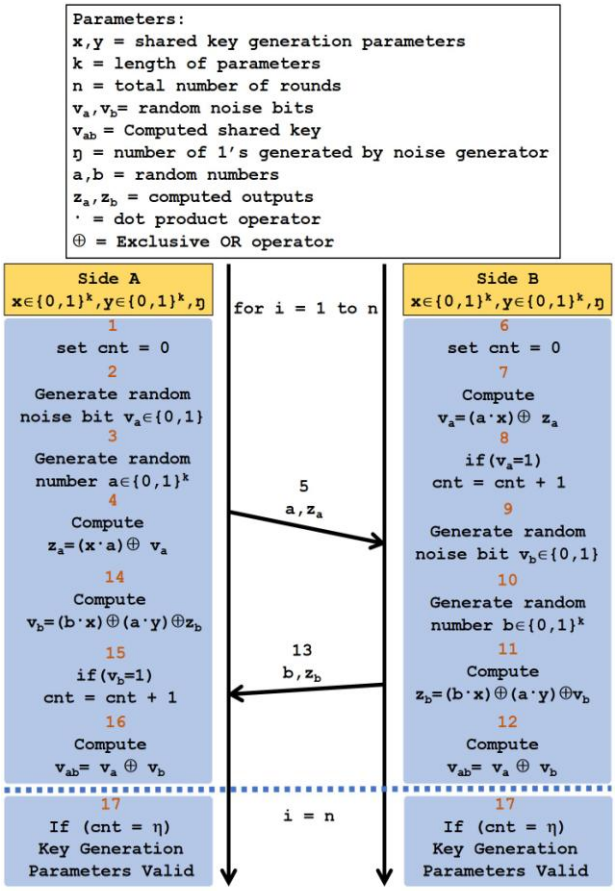
$$v_a = (a \cdot x) \oplus z_a \tag{5}$$

$$z_b = (b \cdot x) \oplus (a \cdot y) \oplus v_b \tag{6}$$

$$v_b = (b \cdot x) \oplus (a \cdot y) \oplus z_b \tag{7}$$

$$v_a = v_a \oplus v_b \tag{8}$$

The second step is to ensure that both sides have the same shared parameters x and y. This is done by simply comparing the *cnt* value to the noise parameter η. If they are equivalent, then both sides are authenticated and the key generated is valid. The top-level flow of the proposed key exchange protocol is shown in Fig. 2.



Flow Diagram of Proposed Key Exchange Protocol

IV. KEY EXCHANGE PROTOCOL HARDWARE ARCHITECTURE

Recently, a majority of lightweight security proposals do not come with real-world hardware architecture implementations. This makes it difficult to ascertain its feasibility when it comes to low-cost devices. This section describes the hardware implementation of the proposed lightweight key exchange protocol together with some synthesis and simulation results.

A. Hardware Architecture

The hardware architecture of the proposed lightweight key exchange protocol is shown in Fig. 3. The same hardware architecture can be implemented by all devices wishing to generate shared keys. The architecture consists of both a Datapath and a Control unit. The inputs to the Datapath include a 128-bit seed value, a 128-bit key_x, a 128-bit key_y, computed value zb from the communicating partner, 7-bit noise_cnt and noise_parameter from the Control unit. The outputs include a 128-bit random number a, computed value za, a 128-bit generated shared key (shared_key), and a valid signal. The modules in the Datapath include a PRNG unit, a PRBG unit, COMPUTE1 unit, COMPUTE2 unit, VERIFICATION unit, XOR gate, and a SHIFT REGISTER unit.

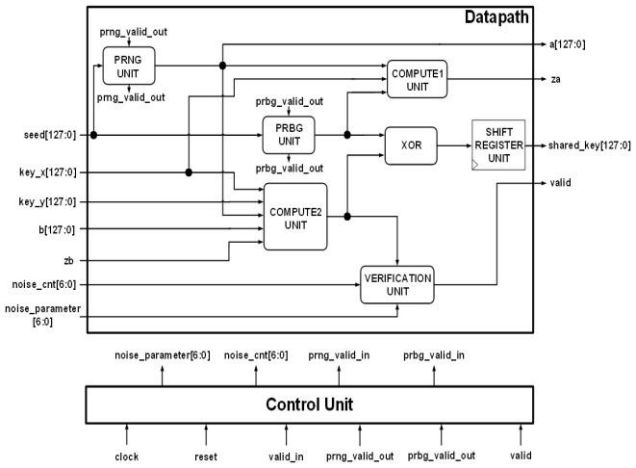


Fig. 3. Proposed Hardware Architecture of the Key Exchange Protocol

The PRNG unit and the PRBG unit use Linear Feedback Shift Register (LFSR) method to generate a 128-bit random number and a random bit respectively. The PRNG unit used here generates a 128-bit random number using 128-bit seed from the seed register while the PRBG unit generates a random bit using 4-bit seed from the seed register.

The COMPUTE1 unit computes the output za by using as its inputs 128-bit value key_x, 128-bit random number a, and 1-bit random bit va. The key_x and a values are passed through a DOT PRODUCT unit. The DOT PRODUCT unit is implemented using the architecture designed by [16]. The output from the DOT PRODUCT unit is XORed with the random noise bit (va) to produce the output za. The output value za is sent to the communicating partner and it is used for the computation of the random bit generated. The hardware architecture of the COMPUTE1 unit is illustrated in Fig. 4.

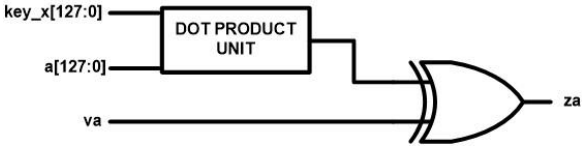


Fig. 4. Hardware Architecture of COMPUTE1 Unit

The COMPUTE2 unit estimates the random bit (vb) generated by the communicating partner by using as its inputs 128-bit value key_x, 128-bit value key_y, 128-bit random number a, 128-bit random number from the partner (b) and 1-bit computed value from the partner (zb). The key_x and b values are passed through a DOT PRODUCT unit while key_y and a are also passed through a second DOT PRODUCT unit. The outputs from the DOT PRODUCT units are XORed with the zb value which results in the computation of the vb value. The architecture of the COMPUTE2 unit is shown in Fig. 5.

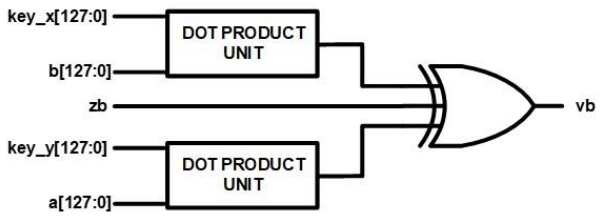


Fig. 5. Hardware Architecture of COMPUTE2 Unit

The VERIFICATION unit is a simple logic that indicates whether the shared key generated is valid or not after the whole key generation process. It takes as its inputs two 7-bit values noise_cnt and noise_parameter from the Control unit. It also takes in vb from the COMPUTE2 unit. The noise_cnt and noise_parameter values are passed through a comparator and if they are equivalent, the shared key is deemed valid, otherwise, it is deemed invalid. The comparator is designed using only AND gates. The hardware architecture of the VERIFICATION unit is shown in Fig. 6.

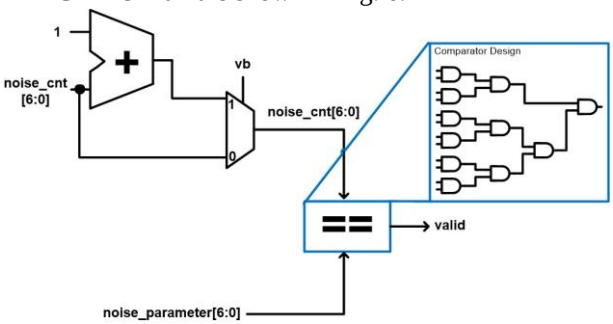


Fig. 6. Hardware Architecture of VERIFICATION Unit

B. Synthesis Results of Hardware Architecture

The hardware architecture of the proposed lightweight key exchange protocols was designed using Verilog HDL and synthesized using Xilinx ISE 14.7 software. The ISE software is embedded with XPower Analyzer which was used to estimate the power consumption of the design. The design was implemented on a Virtex4 FPGA device.

Since most lightweight key exchange protocols do not come with a real-world hardware implementation, the proposed design was compared with the compact version of the RSA [22] and ECDH [23] public key protocols. The metrics for comparison includes the hardware area, maximum operating frequency, latency, and power consumption. The results of the comparison are shown in Table II.

From the table, the proposed design used 312 slices, achieve a maximum frequency of 432 MHz, generated a shared key in 76 microseconds and consumed the power of 33 mW. With regards to hardware area measured in slices, the proposed design achieves an area-saving of 93% and 95% as compared to ECDH and RSA respectively. In terms of maximum achievable frequency measured in Megahertz, the proposed design is 4 times and 3 times higher than that of RSA and ECDH respectively. With regards to latency, it takes 420 times and 3 times more time to generate a shared key using RSA and ECDH as compared to the proposed protocol. Finally, ECDH consumes 12 mW more power than that of the proposed design while the power consumption of RSA was not recorded.

Table- II: Synthesis Results and Comparisons

Design	Area (Slices)	Frequency (MHz)	Latency (us)	Power (mW)
[22]	12881	100	31930	-
[23]	9670	147	283	45
Proposed	312	432	76	33

C. Simulation Results and Test Vectors

The functional simulation of the proposed key exchange protocol architecture was done using Xilinx ISE ISim which enables the functional and timing simulation of designs written in HDL. The design is simulated using a clock frequency of 100MHz. The simulation waveform is shown in Fig. 7.

From the Figure, label A illustrates the operations in the first round of the key generation while label B is for the last round. All values are in hexadecimal format. Side A and side B share the same 128-bit key_x (ABCDEFADFFDADCDA8452695648587540) and key_y (8452695648587542ABCDEFADFFDADCDA) parameters. The 128-bit seed at side A for generating the random number and bit is given as adeacde54295641254ad8122de7543ad while the seed at side B is given as 1254856954adeaca4259a4d5c8921536. In the first round, side A generates a random bit (va) and the random number (a) before computing the output (za). The random number a and za are sent through port a_z to side B as shown in label 1. Side B computes va, generates vb, generates b and computes zb. b together with zb are sent to side A through port b_z.

In the last round, at both side A and side B, the done signal is asserted because noise_cnt and auth_cnt are equivalent. The 128-bit shared key generated is given as 98155B328749326D07638DC327653F75. This is shown in both label 3 and label 4.

To investigate the possibility of generating the same or a similar shared key when both sides do not share the same parameters, the key_x value at side B was varied by just 1 bit and the shared key generated is observed.

SIDE A					SIDE B				
key_x	key_y	seed	shared_key	Valid	key_x	key_y	seed	shared_key	Valid
ABCDEFADFFDADCDA 8452695648587540	8452695648587542ABC DEFADFFDADCDA	adeacde5429564125 4ad8122de7543ad	98155B328749326D0 7638DC327653F75	✓	ABCDEFADFFDADCDA8 452695648587540	8452695648587542A BCDEFADFFDADCDA	1254856954adeaca42 59a4d5c8921536	98155B328749326D07 638DC327653F75	✓

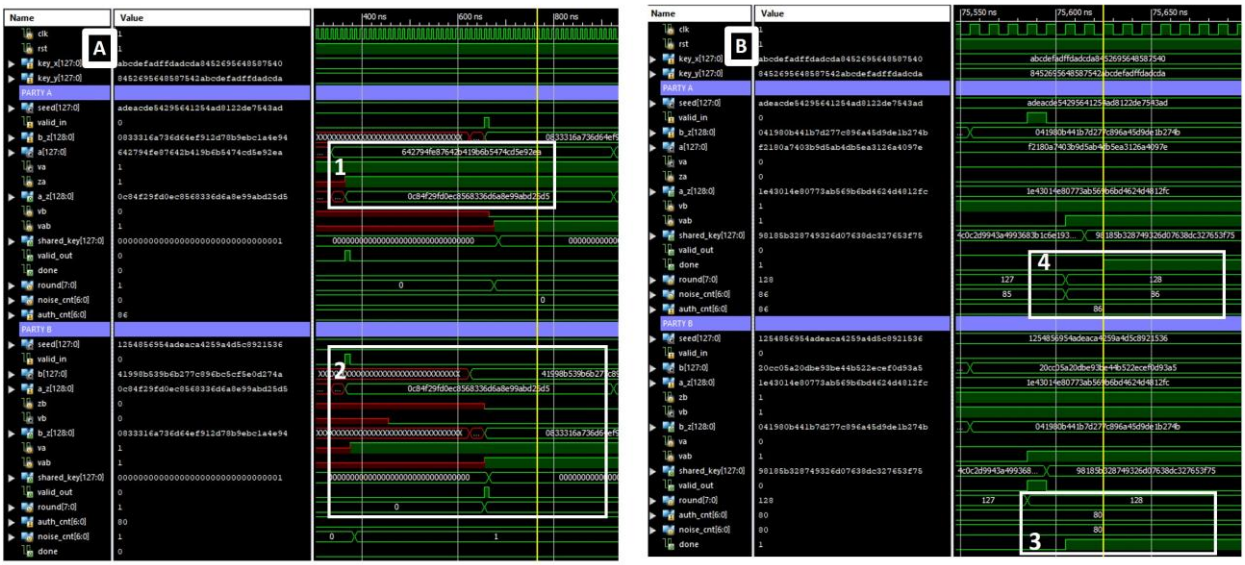


Fig 7. Simulation Waveform of the Proposed Key Exchange Protocol

SIDE A (seed = adeacde54295641254ad8122de7543ad)					SIDE B (seed = 1254856954adeaca4259a4d5c8921536)				
key_x	key_y	shared_key	Valid	Hamming Distance	key_x	key_y	shared_key	Valid	Hamming Distance
ABCDEFADFFDADCD A8452695648587540	8452695648587542A BCDEFADFFDADCDA	98155B328749326D 07638DC327653F75	✓	0	ABCDEFADFFDADCDA8 452695648587540	8452695648587542A BCDEFADFFDADCDA	98155B328749326D 07638DC327653F75	✓	0
ABCDEFADFFDADCD A8452695648587540	8452695648587542A BCDEFADFFDADCDA	BBCCE6AE5CF7A5C5 3336458995529E62	×	67	ABCDEFADFFDADCDA8 452695648587541	8452695648587542A BCDEFADFFDADCDA	B338F2E8E26A0E1A DDD400FE2EF11569	×	65
ABCDEFADFFDADCD A8452695648587540	8452695648587542A BCDEFADFFDADCDA	279606D14B9697B8 1B1932EFDFFD3B61	×	69	ABCDEFADFFDADCDA8 452695648587542	8452695648587542A BCDEFADFFDADCDA	499C8D0A006B193C 6B2F10AF18E7C6FC	×	61
ABCDEFADFFDADCD A8452695648587540	8452695648587542A BCDEFADFFDADCDA	0442BB4D90280010 2F4CFAA56DCA9A76	×	67	ABCDEFADFFDADCDA8 452695648587543	8452695648587542A BCDEFADFFDADCDA	62BC24D06548254B B1989D921173ECE0	×	65
ABCDEFADFFDADCD A8452695648587540	8452695648587542A BCDEFADFFDADCDA	E9BB76EEC0260E898 F0E895CFAAAE9E0	×	71	ABCDEFADFFDADCDA8 452695648587544	8452695648587542A BCDEFADFFDADCDA	B4CEB2FB716B92AF 3052988783ECAF36	×	55
ABCDEFADFFDADCD A8452695648587540	8452695648587542A BCDEFADFFDADCDA	CA6FCB721B98992E BB5B4116489D48F7	×	63	ABCDEFADFFDADCDA8 452695648587545	8452695648587542A BCDEFADFFDADCDA	9FEE1B211448AED8 EAE515BA8A78852A	×	65

Fig. 8. Test Vectors for the Proposed Key Exchange Protocol

The results of this experiment are illustrated in Fig. 8.

From the figure, it can be observed that when both sides do not share the same parameters, the generated shared key is different at both ends the key is deemed invalid. The Hamming distance (calculated by computing the difference in bits position between the valid generated shared key and the invalid generated shared key) indicates that the invalid keys are completely different from the valid generated shared keys.

V. FPGA BOARD TEST

A real-world test of the proposed lightweight key exchange protocol was done on HBE-SOC-IPD board which is equipped with a Virtex4 XC4VLX80 FPGA device. The hardware architecture for the test is shown in Fig. 9. From the figure, two modules of the key exchange protocol were implemented to depict communication between two entities (side A and side B). PICORV32 [24] is a synthesizable processor core that is used for controlling, scheduling and configuring the protocols. The processor core implements the RISC-V open-source Instruction Set Architecture (ISA). Other components such as LEDs and segment display are used for monitoring control signals.

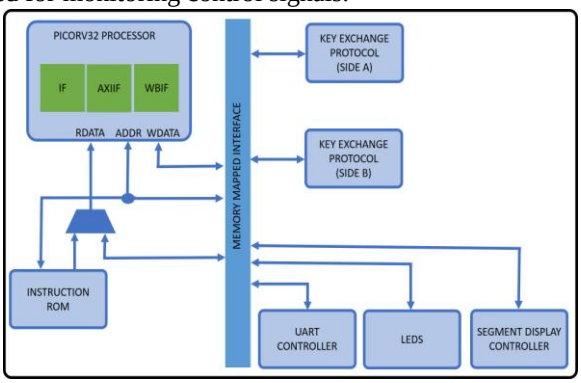


Fig. 9. Hardware Architecture for Testing the Proposed Key Exchange Protocol

A. Board Test Setup

Fig. 10 shows the setup used to test the proposed lightweight key exchange protocol. The setup consists of a test board that is equipped with an FPGA device that implements the design, Universal Asynchronous Receiver Transmitter (UART) for transfer of data to and from the test board, LEDs and segment display for signal monitoring. The setup also consists of a computer with a desktop application that serves as an interface to the FPGA test board. The desktop application Graphical User Interface (GUI) was designed using Microsoft Visual Studio software.

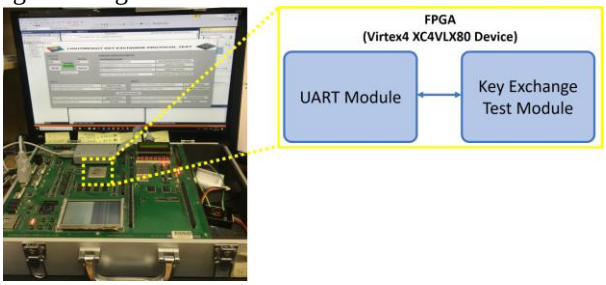


Fig. 10. FPGA Board Test Setup

B. Board Test Results

Fig. 11 shows the GUI of the desktop application and the procedure involved in testing the proposed lightweight key exchange protocol. To generate shared keys between two sides, first, the UART communication port must be selected and connected to enable communication between the desktop application and the FPGA test board. After a successful connection, the user can display the SEED and the NOISE parameter used at both sides for the key generation. The user then enters the shared parameter Key_X and Key_Y which are then sent to the key generation modules. The key generation is then started. The generated shared keys at both sides together with the validity of the keys are finally displayed.

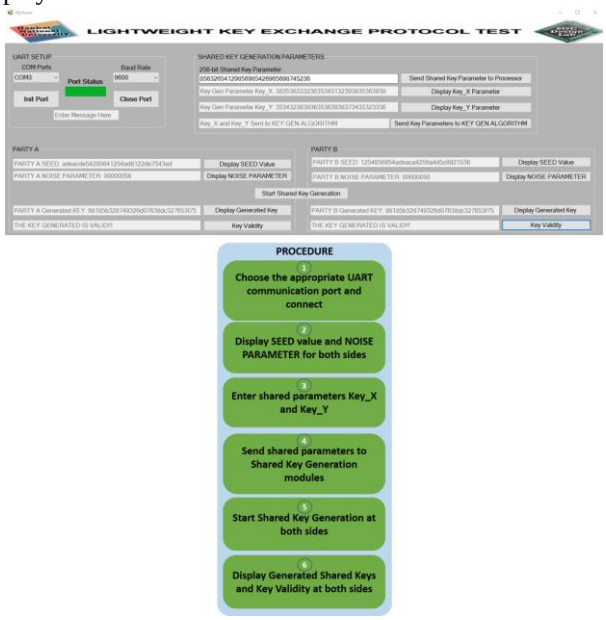


Fig. 11. FPGA Board Test Results

VI. CONCLUSION AND FUTURE DIRECTION

This paper illustrates the design and implementation of a lightweight key exchange protocol for low-cost devices used at the end nodes of IoT platforms. The protocol has the capability of generating shared keys for two entities and also authenticating the parameters used for generating the shared keys. The security of the proposed protocol is based on the LPN problem which is known to be difficult to solve. The protocol makes use of pseudorandom number generator, pseudorandom bit generator and basic logic gates for implementation in hardware making it very efficient. A hardware architecture is proposed for the protocol which was synthesized using Xilinx Virtex4 FPGA device. The hardware architecture consumed 312 slices at 434 MHz maximum clock frequency making it convenient for constrained devices. The hardware design was tested on an FPGA board using a synthesizable RISC-V processor core for controlling, scheduling and configuring the protocols. In the future, the algorithm will be embedded in a lightweight cryptographic module that is capable of key generation, encryption/decryption, and authentication.

REFERENCES

1. The European Parliament and the Council of the European Union, “Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation),” Official Journal of the European Union, vol. 59, Apr. 2016, pp. 1–88.

2. J. Guth, U. Breitenbucher, M. Falkenthal, P. Fremantle, O. Kopp, F. Leymann, and L. Reinfurt, A Detailed Analysis of IoT Platform Architectures: Concepts, Similarities, and Differences. Internet of Everything (Technology, Communication, and Computing), Singapore: Springer, 2018, ch. 4.
3. L. Adleman, A. Shamir, and R. Rivest, "A Method for Obtaining Digital Signatures and Public-key Cryptosystems," Communication of the ACM, vol. 21, Feb. 1978, pp. 120–126.
4. W. Diffie and M. Hellman, "New Directions in Cryptography," Communication of the ACM, vol. 22, Nov. 1976, pp. 642–643.
5. J. S. Coron, A. Gouget, P. Paillier, and K. Villegas, A Single-Party Public-Key Authenticated Key Exchange Protocol for Contact-Less Applications. Financial Cryptography and Data Security, Berlin: Springer, 2010, ch. 11.
6. A. Shamir, "RSA for Paranoids," CryptoBytes (The Technical Newsletter of RSA Laboratories), vol. 1, Aug. 1995, pp. 1–16.
7. National Institute of Standards and Technology, "Advanced Encryption Standard (AES)," Federal Information Processing Standards, vol. 197, Nov. 2001, pp. 1–51.
8. J. E. Juan, L. Jon, L. Janire, and R. D. G. Jonathan, "A Lightweight Authenticated Key Exchange for Class 0 Devices," International Journal of Distributed Sensor Networks, vol. 12, May. 2016, pp. 1–5.
9. J. P. Aumasson and D. J. Bernstein, "A Fast Short-Input PRF," Lecture Notes in Computer Science, vol. 7668, May. 2012, pp. 489–508.
10. K. K. Ahmad, Session-HB: Improving the Security of HB+ with a Session Key Exchange. Innovation and Interdisciplinary Solutions for Underserved Areas, International: Springer, 2018, ch. 1.
11. J. Ari and A. W. Stephen, "Authenticating Pervasive Devices with Human Protocols," Lecture Notes in Computer Science, vol. 3621, May. 2005, pp. 293–308.
12. P. L. Pedro, C. H. C. Julio, M. E. T. Juan, and R. Arturo, "Advances in Ultralightweight Cryptography for Low-cost RFID Tags," Lecture Notes in Computer Science, vol. 5379, May. 2008, pp. 56–68.
13. A. Waterman, Y. Lee, D. A. Patterson, and K. Asanovic, "The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Version 2.1," University of California at Berkeley/Department of Electrical Engineering and Computer Science Technical Reports, vol. 2016, Sep. 2016, pp. 1–133.
14. B. S. Yosra, O. Alexis, Z. Djamal, and L. Maryline, "Lightweight Collaborative Key Establishment Scheme for the Internet of Things," Computer Networks, vol. 64, Feb. 2014, pp. 273–295.
15. J. H. Nicholas and B. Manuel, "Secure Human Identification Protocols" Lecture Notes in Computer Science," Lecture Notes in Computer Science, vol. 2248, May. 2001, pp. 52–66.
16. D. A. N. Gookyi and K. K. Ryoo, "Hardware Design and Analysis of HB Type Lightweight Authentication Protocol for Pervasive Devices," Journal of Pure and Applied Mathematics, vol. 118, Feb. 2018, pp. 1927–1946.
17. A. Rafael, C. G. Candido, S. Juan, and Z. Antonio, "Algorithms for Lightweight Key Exchange," Sensors, vol. 17, Feb. 2017, pp. 1–14.
18. R. N. Boateng and K. K. Ryoo, "ASIC Design of Low Area RSA Crypto-core Based on Montgomery Multiplier," International Journal of Engineering and Technology, vol. 7, Mar. 2018, pp. 278–283.
19. G. P. Byung and W. S. Kyung, "A Hardware Implementation of Public-key Cryptographic Processor for 233-bit Elliptic Curve Over Binary Field," IDEC Journal of Integrated Circuits and Systems, vol. 4, Jul. 2018, pp. 1–6.
20. F. Armknecht, M. Hamann, and V. Mikhalev, "Lightweight Authentication Protocols on Ultra-Constrained RFIDs - Myths and Facts," Lecture Notes in Computer Science, vol. 8651, May. 2015, pp. 1–18.
21. A. Blum, A. Kalai, and H. Wasserman, "Noise-tolerant Learning, the Parity Problem, and the Statistical Query Model," Journal of the ACM, vol. 50, Nov. 2003, pp. 506–519.
22. K. Mohamed, W. Hau, and P. Arul, "Design and Implementation of a Private and Public Key Crypto Processor for Next-Generation IT Security Applications," Malaysian Journal of Computer Science, vol. 19, Nov. 2006, pp. 26–45.
23. S. Anissa, Z. Medien, and M. Chiraz, "Design and Implementation of Low Area/Power Elliptic Curve Digital Signature Hardware Core," Electronics, vol. 19, Mar. 2017, pp. 1–23.
24. C. Wolf. (2019, Aug 05). PicoRV32 – A Size-Optimized RISC-V CPU [Online]. Available: https://github.com/cliffordwolf/picorv32.

AUTHORS PROFILE

Dennis Agyemanh Nana Gookyi received his 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 was awarded his BSc., MSc, 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 Korea Military Academy (KMA) in South Korea. He later worked as a Senior Researcher at the Electronics and Telecommunications Research Institute (ETRI), Korea, from 2000 to 2002. He was a Visiting Scholar at the University of Texas in Dallas from 2010 to 2011. 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.