International Journal of Engineering & Technology, 7 (2.6) (2018) 231-235

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research Paper

Effectiveness of the NIZKP Protocol for Authentication in IoT Environment

Teyi Yann Cedric Lawson [1]*, Senthilnathan T [2]

[1]PG Scholar, Department of Computer Science, CHRIST (Deemed to be University), Bangalore, India
[2]Associate Professor, Department of Computer Science, CHRIST (Deemed to be University), Bangalore, India
**Corresponding author E-mail: lawson.cedric@cs.christuniversity.in*

Abstract

Elliptic Curves when compared to other encryptions scheme such as RSA etc., provides an equivalent security, smaller key sizes, less power consumption, faster calculations, less bandwidth used and is more suitable for Internet of Things devices. In addition of encrypting the data, the devices in the network should also be able to authenticate themselves, which can be achieved with the implementation of "Non-Interactive Zero Knowledge protocol" (NIZKP). This protocol involves two parties: The prover and the Verifier. Prover party should prove to the Verifier that they have the knowledge of something, without revealing what is it. In this paper, a study of Schnorr protocol or ∑- protocol over Elliptic Curves is done and the protocol is implemented in Python using the Python Cryptography Toolkit PyCrypto which is a collection of cryptographic modules implementing various algorithms and protocols. Finally, the results were compared with Elliptic Curve Diffie-Hellmann(ECDH) and present a performance evaluation of the protocols on the Raspberry Pi 3B model, a credit-card sized computer used for the development of IoT devices hence the perfect platforms to test the protocol.

Keywords: *NIZKP, ECC, Internet of Things, Raspberry Pi, Elliptic Curve Diffie-Hellman.*

1. Introduction

The Internet of Things is a technology that aims to monitor and connect billions of information devices that contain sensors, actuators, microprocessors, communication interfaces and power sources. The need for security is because there is no uniform infrastructure from one device to another and that these devices communicate wirelessly; all this adding up makes them prone to security attacks such as eavesdrop, Man-In-The-Middle and so; since most of the IoT objects have constraints resources in terms of power, memory and processing capability, it follows that lightweight algorithms are necessary to obtain an efficient end-to-end communication because they use minimal power consumption and less memory than the traditional algorithms such as RSA. [1] Authentication, Integrity, and confidentiality are the pillars of Network Security. In order to be properly secured, each device in the IoT network should authenticate itself to the rest of the devices each time that communication is initiated. A suitable protocol for this is the Zero Knowledge proof, where one entity called the Verifier seeks to verify that another entity, the Prover, can prove that he knows something without having to reveal what he knows. In that way, no information is leaked and the user who is proving a statement is authenticated if the verifier asses that the proof is valid. This assessment can be done in an interactive manner through a series of challenges or in a non-interactive manner through a one-time challenge. There exists a various type of zero-knowledge protocol such as graph isomorphism, discrete logarithms, fair coin flips etc. Example of some Internet applications that use the zero-knowledge proofs is e-Voting, e-Commerce, access authorization etc. The proposed approach is considering the non -interactive way, since it will use less computation and memory, for this end the Sigma protocol and zero-knowledge proof are combined with the ECC algorithm to provide authentication and data protection for the Internet of Things.

2. Related Work

In the past years, researchers have proposed many ECC based and also Zero Knowledge-Proof(ZKP) based security protocols for resources-constrained devices to overcome the security and privacy challenges present in the IoT. Francisco Martín-Fernández, Pino Caballero-Gil and Cándido Caballero-Gil [3] have proposed a method for authenticated exchange of confidential data in an insecure channel based on the concept of a non-interactive ZKP which verify the legitimacy of the sender in a single communication. Ioannis Chatzigiannakis, Apostolos Pyrgelis, Paul G. Spirakis, Yannis C. Stamatio [2], claim to be the first to use a well-established Zero Knowledge Interactive Protocol based on the discrete logarithm problem and optimized by implementing ECC settings with regards to resources constrained devices. Authors I.-H. et al. [4] have implemented a Multi-Graph Zero-knowledge-based authentication. A. P. Haripriya and K. Kulothungan [5] proposed an ECC based authentication that implement Zero Knowledge proof in the context of Internet of Things. T. Yalçin [6], proposed a secure lightweight ECDSA for the IoT. Pádraig Flood, Michael Schukat [7], have proposed a method combining ZKP and key exchange mechanism to provide secure and authenticated communication in M2M networks.

3. Importance of security and its challenge in the internet of things

There are serious technical reasons why security in IoT is not trivial. The basic problem is that the proven technologies used to date to secure traditional interactions with the Internet will not work properly with the Internet of Things. For example, to use a public key infrastructure (PKI), each terminal must be able to store digital keys and execute encryption and decryption algorithms, conduct sophisticated handshakes to establish secure SSL connections, etc. Many nodes such as passive RFID tags simply do not have the electrical power, storage, or processing power to perform even the simplest of PKI.
Second, much of the Internet of Things currently relies on machine-to-machine (M2M) technologies. In other words, IoT sensors talk to each other instead of talking to a centralized server. If your smart thermostat tells your dishwasher when to start, that communication goes over your Wi-Fi or Bluetooth network, even without going over the Internet, you're taking great risks. It goes without saying that the Wi-Fi and Bluetooth protocols are easily hackable, but how do the two communication nodes know that the information coming from the other is allowed? Any type of M2M interaction requires a certain level of trust, only we have no way to predict that confidence a priori, or to be able to revoke it if an incident occurs. How can your dishwasher know someone has hacked your thermostat?
A significant amount of sensitive data is shared among the IoT devices, (medical data recorded by health monitors, location needed to provide a spot in a smart parking application, etc.) and if those data where to be breached it could cause some serious problem to the user or compromise the IoT network. Security and privacy of IoT are hence of prime importance. The author in [1] resumes the challenges faced by the IoTs as follow:
- Passive or non-existent human intervention might lead to physical and logical attacks.
- Communication done through a wireless channel are prone to attacks such as man-in-middle, DoS (Denial of Service), eavesdropping etc.
- Unauthorized access may easily be granted due to the interconnection ability of these devices
- Resources constrained devices can't support intricate security solutions.
- Power limitation
- Heterogeneous platforms
- Network scalability, bandwidth etc.

A secure IoT device should have the following abilities [1]:

- Confidentiality: Data should only be accessible to the sender or receiver whether it is at rest or in transit.
- Integrity: No intruders should be able to modify the original contents of the data while it is in transit.
- Authentication: The identity of the sender should be verified so that the receiver can judge the validity of the data.
- Authorization: Only authorized users should be able to access and maintain the resources of the IoT.

Most common attacks to which the IoTs are exposed are [8]:

- Interruption: The aim of the attacker is to affect the availability of the system (example shutting it down) which usually results in exhaustion of the resources.
- Eavesdropping: The attacker is spying on the communication between the devices, compromising the confidentiality of the data.
- Alteration: Attacker may alter the data being forwarded between the sender and the receiver misleading the communication and threatening the integrity of the data.
- Message replay: Attacker intercepts and resends the data after modifying it, confusing the targeted node in the network.
- Man-in-the-middle: Attacker secretly eavesdrops and possibly altering the data, inducing the two parties that they are directly communicating with each other.

4. Asymmetric cryptography

Cryptography is a cryptology discipline that focuses on protecting messages and ensuring confidentiality, authentication, and integrity by using secrets or keys. Symmetric key cryptography has long been used for the encryption of confidential messages. Its use has been progressively reduced since the advent of public key cryptography (asymmetric cryptography) even though both techniques are still sometimes used together. In symmetric key or secret key encryption, it is the same key that is used both to encrypt and decrypt a message. It's exactly the same principle as a door key: it's the same thing used to open and close a lock.
Asymmetric cryptography (also known as public-key cryptography) is a method used to transmit and exchange messages securely by ensuring that the following principles are respected:

- Issuer Authentication
- Integrity guarantee
- Confidentiality guarantee

This technique is based on the principle of "key pair" (or two-key) consisting of a so-called "private key" kept completely secret and must not be communicated to anyone and a key called "public" which, like its name may be transmitted to all without any restriction. The so-called asymmetric keys are encryption keys. Encryption is the general name given to mathematical coding or decoding techniques.
The general principles of public key cryptography are:

- A message encoded with a private key can only be decoded by the associated public key.
- A message encoded with a public key can only be decoded by the associated private key.
- A given public key can only be associated with one private key.
- Several different private keys cannot have the same public key as a complementary key.
- A given private key can only be associated with one public key.
- Several different public keys cannot have the same private key as a complementary key.

Symmetric key uses less number of keys and less key size but it doesn't provide authentication. Popular symmetric key algorithms are AES, DES, 3DES, BLOWFISH, RC5, PRESENT etc.
Asymmetric key meets all the security requirements but is not suitable for resources-constrained devices due to the large size of the key generated. RSA, DIFFIE-HELLMAN KEY EXCHANGE, ECC are popular Asymmetric Key Algorithms.
Due to the reasons cited above lightweight algorithms are henceforth more fit to implement security and privacy in IoT.

5. Elliptic curve Diffie-Hellman

Elliptic Curve Diffie-Hellman (ECDH) is an exchange of keys based on the Diffie-Hellman algorithm. Two parties let say Yann and Cedric want to securely exchange information in such a way that even if a third-party intercept them, he won't be able to decode them.

Parameters of the domain
This algorithm work in a cyclic subgroup of an elliptic curve over a finite field. The parameters of the algorithm are:

- A prime **p** that specifies the size of the finite field.
- The coefficients **a** and **b** for the elliptic curve equation
- The **base point G** from which the subgroup is generated
- The **order n** of the subgroup
- The cofactor **h** of the subgroup

Step 1: The two parties generate their own private and public keys.

- A random number **d** picked from {1, …, n-1} constitute the private key
- The point **H = d.G** constitute the public key.

Step 2: Public keys exchange

- Yann and Cedric proceed to the exchange of their public keys $\mathbf{H_Y}$ and $\mathbf{H_C}$ through a channel which is insecure.

Step 3: Calculate the shared secret key using their own private key and the received public key.

- Yann calculates $\mathbf{S = d_Y H_C}$
- Cedric calculates $\mathbf{S = d_C H_B}$

The resulting key $S = \mathbf{d_Y H_C} = \mathbf{d_Y(d_C G) = d_C(d_Y G) = d_C H_Y}$.
The fact that the private key **d** is randomly generated makes it hard for a given third-party performing a Man-In-The-Middle attack to find **d** even if he knows **H** and **G** because, in order to do so, he or she would have to solve the discrete logarithm problem. At the end even if he won't be able to discover the Shared Secret Key **S** without knowing either $\mathbf{d_Y}$ or $\mathbf{d_C}$.

6. An overview of zero-knowledge proof

A zero-knowledge proof is a method by which one party, the prover can prove to another party the verifier that a given statement is true without conveying any information apart from the fact that the statement is true. We do that in case we have a secret that we don't want to tell anyone but we want to prove that we know the secret.

Zero-Knowledge requirements

- Completeness: If Prover is honest and does have a secret that he wants to prove to the verifier who is also honest then everything will work as long as both prover and verifier follow the protocol, Zero-Knowledge Proof will follow.
- Soundness: The prover cannot convince the verifier that he/she know the secret when they actually don't.
- Zero-Knowledge: Verifier knows that Prover has a secret without knowing the actual secret. Anyone spying on the communication between the Prover and Verifier has no way of knowing if it is scripted or genuine, Prover can prove that he/she knows the secret to Verifier and only to him.

Classical Example for Zero Knowledge Proof

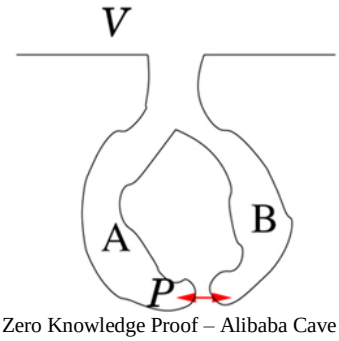

Fig. 1: Zero Knowledge Proof – Alibaba Cave Problem

As depicted in the Fig.1, the Verifier and the Prover enter a cave and there is a door at one end of the cave, which opens only when told the proper secret. So, the Prover wants to prove that she/he knows that secret. Protocol starts, Verifier wait outside the cave and Prover enters it choosing randomly entry A or B. Then Verifier enter the cave and calls out which exit he wants Prover to appear at. Prover appears at the correct exit and Verifier is convinced. But if this is performed only once the Verifier should not be convinced. Let say that Verifier calls out the B exit, and that initially Prover chooses this exit and got stuck at the door because he/she doesn't know the secret to open it, and got lucky when Verifier called her to appear at exit B. It should only be convincing to the Verifier only if this process is repeated several number of times, thus reducing the chances of Peggy lying about knowing the secret to open the door.
So, in this example, Completeness is assured by Prover and Verifier following the protocol. Soundness is assured by repetition of the protocol and Zero-Knowledge is assured because Verifier never learn the secret to open the door, and nobody observing the process outside of Verifier and Prover have no way of knowing whether it was orchestrated or not.

The other application of ZKP is discussed here with Sudoku problem. The objective of the sudoku is to fill the 9x9 grid so that each row, column and block contain exactly 1 of each digit (1-9) as shown in Fig.2. So, let says that Prover **P** has the solution to a sudoku problem but wants to prove it to the Verifier **V**. In order to do so the Prover upload the solution to a computer program and this program is verifiably honest since it is open source and anyone can look at the code and verify that it encrypts the given solution by using a simple substitution cipher key where each transmutation has an equal chance of appearing, in other word digit 1 has as much chance to be transmuted into a 3 or an 8 etc.

Next the program displays the following options to **V**:

- Reveal a row
- Reveal a column
- Reveal a block
- Reveal original problem
- Every time **V** makes a new choice, the program re-encodes the solution with a brand-new key. In the above picture, the first image shows an unmasked row, there is only one of the digit in the row, likewise with the column and the block, and it can be easily mapped to his encoded original puzzle, thus knowing that it is a solution to its puzzle.
- The computer program is verifiably honest and **P** and **V** are following the protocol so Completeness is assured.
- The chances of **P,** cheating **V** are 27/28 if the protocol is performed only once. But if performed multiple times with random choices let say 100 times, the chances are less than 0 .05%.
- Since every permutation of the cell rows column or block are equally likely, **V** learns nothing from the solution other than it is valid. And any outside observer wouldn't be able to separate legitimate transcript from any false one.

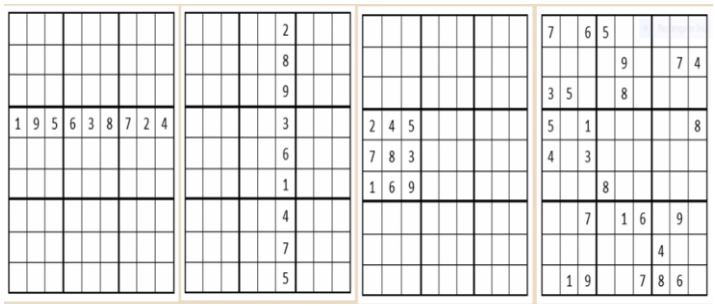

Fig. 2: Zero Knowledge Proof Sudoku Problem

7. Schnorr zero-knowledge protocol over elliptic curves

The first introduction of this scheme was made in 1989 by C.P Schnorr [9], he created a new scheme and a matching authentication all based on the discrete logarithms in a subgroup $\mathbf{Z_p}$ with p

being a large prime. The protocol implemented in this paper, the Non-Interactive Zero-Knowledge Proof (NIZKP) version of the Schnorr Protocol over Elliptic Curve is a modified version of the one introduced in [9] and include the usage of Elliptic Curves.

Protocol steps:

- The Prover and the verifier both agree on an elliptic curve.
- Prover chooses a private key from a = r where r such that r ∈ {0, ..., Q – 1}
- Prover calculates the public key v = −a.G(modP) (a point in the elliptic curve) and sends it to the verifier.
- Prover generates a number R to use in x=r.G(modP) to calculate a point on the elliptic curve and sends it to the Verifier.
- Verifier choses a random number e such that e ∈ {1,2, …, 2^t} and sends it to prover.
- Prover verifies that the value [e] is in the appropriate interval and calculate y = a*e + r and sends it to receiver.
- Verifier computes:
  z = y.G(modP)
  v = e.G(modP)
  z = z+v
  And checks that z.x = x.x and z.y = x.y. if true he accepts else he rejects it.

The non-interactive aspect of this protocol makes it more suitable for the resource constrained devices as it only needs one round of execution. Given that the two parties follow the protocol, completeness is assured. The prover may be able to cheat the verifier if he can guess the challenge **e** sent by the prover. He can then prepare a message accordingly, and if the sender indeed sends the guessed **e** as challenge, the prover would have the correct response to the challenge. In this scenario the prover has a 1 / 2 probability to succeeds, however it is applicable only if the interval of the challenge **e** is restricted to a small space. That scenario fails when **e** is selected from a broader interval. In our protocol the interval is defined at:
{1,2, …, 2^t}, which ensure the soundness of it. A third party observing the transaction would be able to assert that z.x = x.x and z.y = x.y but will not be able to learn anything apart from that hence Zero Knowledge is proved.

8. Evaluation of the protocol

8.1. Pycrypto

The implementation of the algorithms is done on Python using Pycrypto which is a bundle of various encryption algorithms such as AES, DES, RSA, ECC etc., as well as hash function such as SHA256, SHA512, MD5 etc., it provides secure administrative tools. Clients and servers can encrypt the data being exchanged and also authenticate each other. With the help of its arbitrary-length integers, public key algorithms can easily be implemented.

8.2. Hardware

The hardware used for this implementation is the Raspberry Pi 3 B which is a credit sized computer, on which IoT project can easily be set up on. It consists of a Broadcom BCM2837, a CPU running at 1.2 GHZ quad-core ARM Cortex A53(ARMv8 Instruction Set), GPU (Broadcom VideoCore IV @ 400 MHz, a memory of 1GB LPDDR2-900 SDRAM, 4 USP ports and is equipped with a 10/100 MBPS Ethernet, 802.11n Wireless LAN, Bluetooth 4.0. It runs on Raspbian Jessi a Debian-based operating system which has been specially conceived for the Raspberry Pi.

8.3. Results

All operations of the Elliptic Curve Cryptography have been done on Python using PyCrypto. The implementation of the ECDH uses the curve secp256k1 (which is also used by Bitcoin in digital signatures.) with the equation:

$$y^2=x^3+ax^2+b. \quad (1)$$

The parameters of the curve are:

-Field characteristic:
p=0xfffffffffffffffffffffffffffffffffffffffffffffffffffffffefffffc2f
- Curve coefficients:
a=0, b=7
- Base Point G with:
X=0x79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798,
Y=0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8
- Subgroup order:

n=0xfffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141,
- Subgroup cofactor:
h=1

For the Schnorr protocol over Elliptic Curves, Brainpool curve over a 192-bit prime field has been used. The parameters of the curve are:
P=C302F41D932A36CDA7A3463093D18DB78FCE476DE1A86297
a=6A91174076B1E0E19C39C031FE8685C1CAE040E5C69A28EF
b=469A28EF7C28CCA3DC721D044F4496BCCA7EF4146FBF25C9
Base point G with coordinates:
x=C0A0647EAAB6A48753B033C56CB0F0900A2F5C4853375FD6
y=14B690866ABD5BB88B5F4828C1490002E6773FA2FA299B8F
n=C302F41D932A36CDA7A3462F9E9E916B5BE8F1029AC4ACC1
h = 1
Both the protocols are implemented between client and server using Secure Shell and socket programming.
In Table 1 we can observe the execution time taken to perform the fundamentals Elliptic Curves operations by both the algorithms on the Raspberry Pi.

Table 1: Execution Time in second of Fundamentals ECC Operations in NIZKP Schnorr Protocol over Elliptic curves and Elliptic Curves Diffie-Hellman on Raspberry Pi 3 B.

Operations	Execution Time in seconds	
	NIZKP Schnorr Protocol over Elliptic curves	Elliptic Curves Diffie-Hellman
The point addition on the curve	0.344830036163	0.975810050964
The scalar multiplication	0.457688808441	0.976742982864
The keys generation	0.03047490119934	0.423557043076

The interpretation of the above table shows that the NIZKP Schnorr Protocol over Elliptic curves performs faster than the Elliptic Curves Diffie-Hellman algorithm. For both the algorithms, the scalar multiplication is the most exhaustive operation, but in the NIZKP Schnorr Protocol it takes almost half the time needed by the ECDH to perform. When under heavy CPU loads the current consumption of the Raspberry Pi 3 B model when none of the

USB ports is used is: 400mA and input Voltage is 5V [10]. Taking into consideration the following formula:

$$\mathbf{E = V \cdot I \cdot t}. \quad (1)$$

A Corresponding Theoretical Energy Consumption for the two algorithms is also proposed in Table 2.

Table 2: Energy consumption in mJ of Fundamentals Elliptic Curve Operations Raspberry Pi 3 B.

Operations	Execution Time in seconds	
	NIZKP Schnorr Protocol over Elliptic curves	Elliptic Curves Diffie-Hellman
The point addition on the curve	689.660072326	1951.620101928
The scalar multiplication	915.377616882	1953.485965728
The keys generation	60.94980239868	847.114086152

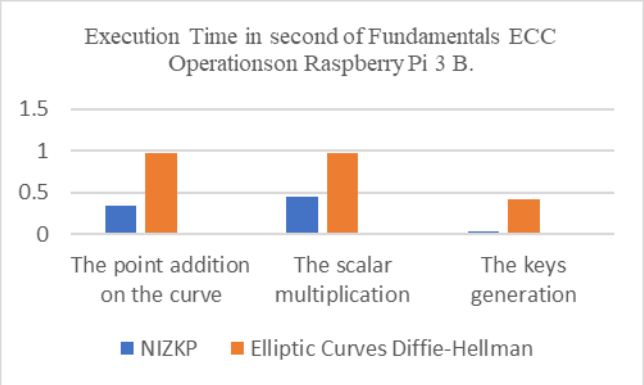

Fig. 3: Execution Time in second of Fundamentals Elliptic Curve Operations on Raspberry Pi 3 B.

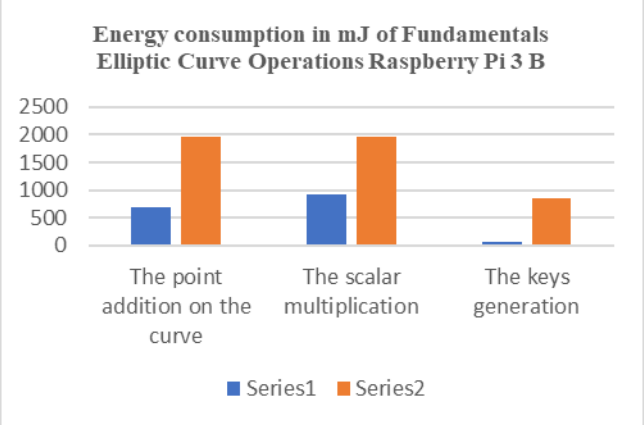

Fig. 4: Energy consumption in mJ of Fundamentals Elliptic Curve Operations Raspberry Pi 3 B.

On the basis of the comparison made and from observing Fig. 3 and Fig. 4 it is safe to say that the NIZKP consumes less in term of power and takes less execution time, hence fitting well into the context of resource-constrained devices.

9. Conclusion

In this paper, A non-interactive Zero Knowledge Proof, more specifically, the Schnorr Protocol Over Elliptic Curves on Raspberry Pi was implemented. Then the results were contrasted with another ECC based algorithm the ECDH and analyzed. It is found that the NIZKP performs faster and have a lesser energy consumption than the ECDH. Using this protocol, devices can authenticate themselves without leaking any important information which reduces the chances for any third-party to have access to them. This is a lightweight protocol which can be incorporated comfortably into any resource-constrained device.

Future Work

As future work, the authors are planning on improving the NIZKP protocol on Autonomous vehicle as a mean of authentication.

Acknowledgement

The authors gratefully acknowledge the financial support from the Department of Computer Science, Christ University, Bangalore, India.

References

[1] Isha and A. Luhach, "Analysis of Lightweight Cryptographic Solutions for Internet of Things", *Indian Journal of Science and Technology*, vol. 9, no. 28, p. 7, 2016.

[2] Chatzigiannakis, A. Pyrgelis, P. G. Spirakis, and Y. C. Stamatiou, "Elliptic Curve Based Zero Knowledge Proofs and their Applicability on Resource Constrained Devices," *2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems*, pp. 715–720, Jul. 2011.

[3] F. Martín-Fernández, P. Caballero-Gil, and C. Caballero-Gil, "Authentication Based on Non-Interactive Zero-Knowledge Proofs for the Internet of Things," *Sensors*, vol. 16, no. 1, p. 75, Jul. 2016.

[4] I.-H. Chuang, B.-J. Guo, J.-S. Tsai, and Y.-H. Kuo, "Multi-graph Zero-knowledge-based authentication system in Internet of Things," *2017 IEEE International Conference on Communications (ICC)*, May 2017.

[5] A. P. Haripriya and K. Kulothungan, "ECC based self-certified key management scheme for mutual authentication in Internet of Things," *2016 International Conference on Emerging Technological Trends (ICETT)*, Kollam, 2016, pp. 1-6.

[6] T. Yalçin, "Compact ECDSA engine for IoT applications," in *Electronics Letters*, vol. 52, no. 15, pp. 1310-1312, 7 21 2016.

[7] P. Flood and M. Schukat, "Peer to peer authentication for small embedded systems: A zero-knowledge-based approach to security for the Internet of Things," *The 10th International Conference on Digital Technologies 2014*, Zilina, 2014, pp. 68-72.

[8] M. Nawir, A. Amir, N. Yaakob and O. B. Lynn, "Internet of Things (IoT): Taxonomy of security attacks" *2016 3rd International Conference on Electronic Design (ICED)*, Phuket, 2016, pp. 321-326.

[9] Schnorr, C.P., "Efficient signature generation by smart cards", *Journal of cryptology*, vol. *4*, *no.* 3,1991, pp.161-174.

[10] Raspberry Pi. (2017). *Raspberry Pi FAQs – Frequently Asked Questions.*[online] Available at: https://www.raspberrypi.org/help/faqs/