# Development of Authenticated Key Exchange Protocol for IoT Sensor Layer

Yoanes Bandung and Arvandy

School of Electrical Engineering and Informatics, Institut Teknologi Bandung
Bandung, Indonesia
bandung@stei.itb.ac.id, arvandy.vandy@gmail.com

*Abstract*: An authenticated key exchange for the Internet of Things (IoT) sensor layer is discussed in this paper. This paper presents an enhanced key exchange protocol to provide an authentication scheme and data confidentiality for IoT sensor layer. In our approach, we incorporate an identity-based authentication scheme into the existing key exchange protocol based on Elliptic Curve Diffie Hellman (ECDH). We utilize two communication channels for the process, main channel and auxiliary channel. The main channel is used to exchange key and sensor data and the auxiliary channel is used to exchange the identity information prior to the key exchange process. To provide the data confidentiality, AES encryption algorithm is implemented with a key derived from shared secret key to ensure the Perfect Forward Secrecy. For the evaluations, there are four parameters that are evaluated: the protocol resistance, formal verification of protocol, the protocol security, and performance testing. The protocol resistance was evaluated using security analysis against common security threats on IoT sensor layer. The formal verification of the proposed protocol was evaluated using Scyther, and the protocol security was evaluated using attack scenarios (i.e., authentication and sniffing attack) to prove the authentication and confidentiality. The performance testing was conducted to measure time complexity and memory complexity of the protocol. The experiment results show that the proposed protocol is able to provide an authentication mechanism, data confidentiality, and resilience against common security threats at IoT sensor layers.

*Keywords*: IoT Security, Sensor Layer, Authenticated Key Exchange, ECDH, AES.

## 1. Introduction

Internet of Things (IoT) refers to a system that consists of many devices and layers that are connected to the internet to provide many functionalities for end-users such as control, monitoring, and task automation. The IoT device usually is equipped with sensors and computational power to collect data and enable deployment in many different environments. There are many services that can be provided by the IoT system such as home monitoring, medical, smart shopping system, and many other services. These services offer huge benefits for people's daily life, but also offers concern regarding the security and privacy of the data collected by the sensors [1].

Sensor layer is the lowest layer on the IoT system where all the sensor devices either connect to each other or connected to the central device as the local gateway. One of the security issues on this layer lies in the security and privacy of the data collected by the sensor. The research on this layer mainly focuses on encryption algorithm and key management to overcome this issue [2]. The research on the encryption algorithm focuses on developing a fast and lightweight algorithm that has low computational cost to be used on the IoT devices. The research on key management focuses on developing a key exchange mechanism responsible for generating and distributing the encryption key. On the other hand, the authentication scheme also needed to ensure the devices that communicate in the sensor layer are the valid devices [3]. One Time Password (OTP) based authentication [4], identity-based with public-key certificate authentication [5], and token-based authentication [6] are some researches that have been done by previous researchers to provide authentication for sensor devices in IoT.

Received: March 24th, 2019. Accepted: July 23rd, 2020
DOI: 10.15676/ijeei.2020.12.3.11

Even though a lot of researches have been done to provide an authentication scheme in the sensor layer, the previous research more focuses on implementing authentication by adding a new device that acts as the authenticator. Shivraj et al. [4] implement OTP-based authentication using a central cloud that responsible for generating and distributing the OTP. Salman et al. [5] implement identity-based authentication using public key certificates that require a device to acts as the Certificate Authority (CA) server. Aman et al. [6] offer another solution for authentication scheme using token-based that also requires a device to acts as the authorization server. The addition of a new device means the requirement for extra resources is needed and the authenticator device that manages all the authentication scheme also has a risk of being a single point of failure (SPOF). In this case, if the device authenticator is compromised, the whole authentication scheme in the sensor layer will be compromised too.

In this research, we offer an alternative for the authentication scheme in IoT sensor layers that is applied on the key exchange process. We utilize two communication channels, the main and the auxiliary channel. The main channel is used for key exchange process and sensor data communication. The auxiliary channel is used to exchange identity information prior to the key exchange process. We utilize key exchange protocol based on Elliptic Curve Diffie Hellman (ECDH) that will be authenticated by identity information that are exchanged over the auxiliary channel. The identity information then will be used on the key exchange process as the authentication variables. The ECDH key exchange will generate the same shared secret key for the participants. This key can be used directly for encryption, but it does not provide Perfect Forward Secrecy (PFS), which is a security property that guarantees the safety of the secret keys in case the private key is compromised. To provide PFS in our authentication scheme, the secret key will be derived using a hash function to be later be used as the key on Advanced Encryption Standard (AES) algorithm. The usage of AES aims to secure the sensor's data that are transmitted over the network.

In this research, the Design Research Methodology (DRM) by Blessing et al. [7] was adopted to conduct the research. The DRM framework consists of four stages: research clarification, descriptive study I, prescriptive study, and descriptive study II. Research clarification intends to find the research goal, focus, and scope through the intensive literature study and analysis. After finding the goal and focus of the research, the descriptive study I aims to understand the existing solutions, determine which parts that can be improved and what is the researcher's desired outcome. The prescriptive study aims to realize the expected outcome into the proposed design. The last stage, descriptive study II focus on evaluation process by investigating the outcome of the researcher proposed design and its ability to matches the desired outcome. In this research, the evaluation of the proposed design consists of four aspects which are the resistance of protocol, the verification of the protocol, the security testing, and the performance testing.

The rest of this paper is written into a several parts. On the next part, we present the result of literature analysis from previous researches that are related to the providing authentication scheme for IoT sensor layers. Secondly, we present our proposed authenticated key exchange protocol. Thirdly, we present the experimental results to prove the outcome matches with our research goal which is to provide authentication mechanism and data confidentiality. The research's conclusion is written in the last part of this paper.

## 2. Related Works

This section describes some previous research works that have been done related to our research. The main topic of this research is the security of IoT, and the first stage of this research method is the research clarification. In this stage, we did literature analysis to find the research goal and focus derived from the research topic. The security of IoT become a popular research topic due to its rapid increasement of the IoT devices and applications. Yang et al. [2] did a study on IoT privacy and security issue in 2017. In their research, the security of IoT is analyzed on each layer that consists of perception or sensor layer, network layer, transport layer, and application layer. The research for security and privacy issues on the sensor layer are focuses on the detection of the faulty sensor node and the encryption algorithms and key management. The faulty sensor node can be caused by the physical attack on the devices or compromised over the networks. To keep the service quality, the system needs to have the ability to detect the faulty node and take the actions. Many researchers did studies on the faulty node detections [8]-[10] that focus on identifying faulty nodes, detecting intrusion, and deriving the probability of the intrusion.

Another security issue is finding a suitable encryption algorithm and key management that can be utilized on sensor layers. This is to ensure the confidentiality and privacy of data on the layer. Research on encryption algorithm focus on developing a lightweight, fast, and low computational cost algorithm. Goyal and Sahula [11] did research on providing a lightweight encryption algorithm for IoT devices using ECDH and AES encryption. Usman et al. [12] developed a new symmetric encryption algorithm that named Secure IoT (SIT). The proposed encryption algorithm is the merge of a Feistel and uniforms SPN with 64-bit cipher and key. Nandini and Vanitha [13] did a study on modern lightweight cryptography algorithms for IoT. In their study, the modern lightweight cryptography branch such as HISEC, PRINCE, OLBCA, PRESENT, PRINT, TWINE, and KLEIN are intensively analyzed using cryptanalysis. The analysis result shows the number of S-boxes on algorithm increase the security but also the cost.

Finding a key management scheme that acceptable for sensor devices have been researched continuously by previous researchers. Public key cryptography has been considered as one of the convenient solutions due to its scalability and simple key management. Gaubatz et al. [14] did research to find the most recent state of public key cryptography for low power devices. In their research, there are three most suitable algorithms that can be implemented on the low power devices: NtruEncrypt, Elliptic Curve Cryptography (ECC), and Rabin's scheme. The size of the key become the reason ECC is suited for small sensor devices in IoT. As shown in Table 1, the ECC requires much smaller key size compared to the Rivest-Shamir-Adleman (RSA) algorithm that provides the equal cryptographic strength [15]. The ECC's strength depends on the ability to solve the discrete logarithm.

Table 1. The key size comparisons of ECC and RSA [15].

| RSA (bit) | ECC (bit) |
|---|---|
| 15360 | 512+ |
| 7680 | 384-511 |
| 3072 | 256-383 |
| 2240 | 224-255 |
| 1024 | 160-223 |

ECC is an asymmetric encryption algorithm that covers cryptography mechanism such as key agreement and digital signatures [16]. ECC technique is built upon the elliptic curve that utilized to efficiently generates a small and fast cryptographic key [17]. The core function of the ECC operation is the scalar multiplication of *k.P.* Parameter *P* is a point on the ECC curve while *k* is a positive number. Computing the result of *k.P* into *Q* will be resulting in another point on the curve. Finding the *k* value with the knowledge of *Q* and *P* are proved to be difficult. Until now, there is no sub-exponential-time method that able to solve the discrete logarithm using properly selected parameters. ECDH is one of the ECC variants that provides key agreement protocol. Both parties that want to perform the key exchange and generating a shared secret key need to agree upon the curve type, field size, and some domain parameters as described in the Algorithm 1 [16].

```
Algorithm 1 ECDH
    1) $A$ and $B$ each choose a random number that less than $n$ to be their private key ($k_a$ and $k_b$), where $n$ is the domain parameter.
    2) $A$ and $B$ compute their public key, $Q_a = k_a * G$ and $Q_b = k_b * G$, where $G$ is the domain parameter
    3) $A$ and $B$ exchange their public keys.
    4) $A$ compute shared secret key, $K = k_a * Q_b$.
    5) $B$ compute shared secret key, $K = k_b * Q_a$.
```

The usage of ECC algorithms for key management scheme in the IoT system have been studied continuously. Dahshan [18] studied a key management mechanism based on ECC and the results show that the ECC has the high-security level of secrecy using small key size compared to other protocol such as the distributed key management protocols. Anggorojati et al. [19] did a study on cryptography key management that utilize ECC on the Machine to Machine (M2M) cloud platform. The results show the feasibility to implement the ECC key agreement despite the delay occurred when processing the certificate. Kodali and Nakoti [20] did a study to implements ECDH public key cryptography on the IoT sensor devices. In their study, the implementation uses NodeMCU ESP8266 and NIST P-192 curve to produce the same encryption key for the sensors. Based on the experimental results, the ECDH implementation requires 234,729 of 1,044,464 total storages on the ESP8266. From the experimental result, it can be concluded that the ECDH security model is feasible for sensor devices in the IoT system.

Even though the ECDH able to provide the key exchange management, it does not have an authentication scheme. Both parties that receiving the public key cannot verify if the received public key is originated from the trusted party or from the third impersonation party. This can be led to an attack known as Man-in-the-Middle (MiTM) as illustrated in Figure 1. This kind of attack happens when the attacker alters the original connection between two parties to make the communication flow through the attacker [21]. By redirecting the communication, the attacker can control and eavesdrop the entire communication that happens between both parties. To avoid this attack, the implementation of ECDH is usually being combined with other authentication algorithms such as digital signature or custom authentication scheme.

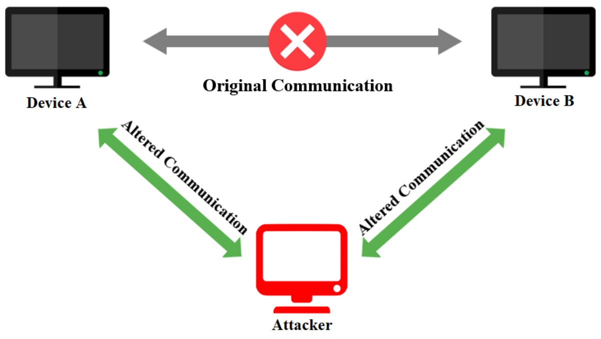


Figure 1. Man-in-the-Middle Attack [21].

Works on [22] studied on securing Voice over the Internet Protocol (VoIP) communication using ECDH as the key agreement protocol on the mobile phone [22]. The ECDH is combined with identity-based authentication to overcome the lack of authentication on the ECDH protocol. To achieve this, the proposed protocol utilizes Short Message Service (SMS) on the mobile phone to exchange the identity prior to the key exchange process. This identity information is sent manually by the user before starting the key exchange process. The results are validated using Scyther tools and proved to successfully overcome the authentication issue on the ECDH protocol despite the execution time that slightly increases.

Shivraj et al. [4] have done research to provide an authentication scheme for IoT sensor layer based on Elliptic Curve Cryptography. In their research, they combine the Lamport's One Time Password (OTP) with a lightweight Identity Based Elliptic Curve Cryptography scheme (IBE-ECC). The OTP will be used as the authenticator when the ECC key exchange occurs. In their authentication scheme, they implement the Private Key Generator (PKG) at IoT cloud platform that responsible for generating and distributing OTP. Sensor device will need to register itself first to PKG to obtain public and private keys. The authentication scheme happens when both devices receive the same OTP from the PKG that will be used as validation when receiving an authentication request. The result concludes that the proposed authentication based on OTP can be implemented in the real IoT system and can be an alternative for providing two-way authentication for IoT devices. The security of the proposed OTP generation scheme also proved to be as difficult as solving the computational problem on the Diffie-Hellman algorithm.

An alternative solution for authentication scheme in the IoT system was also offered by Salman et al. [5] that implementing public-key certificates consist of three devices: sensor, gateway, and controller. In their research, the authentication process has three phases. The first phase is started by the gateway requesting the public key certificate to the controller. This certificate will be used by the gateway when doing authentication to the controller. The next phase is sensor registration, the sensor will request the authentication to the gateway by sending its identity information. The gateway will perform the identity checks and after it confirmed, the request will be forwarded to the controller. The public key for the sensor will be generated by the controller along with the IPv6 address. The controller will be sent back the encrypted hash of the IPv6 address, the sensor's public key, and the gateway's public key to the gateway. The gateway will decrypt and save the hash of sensor identity *H(IPv6)* and the sensor public key. This information then will be sent to the sensors to be later used as the authentication parameters on the last phase. This authentication scheme has some challenges on the computation needed by the sensor. The sensor also needs more storages to store its private key, controller's public key, gateway's public key, and its identity. The proposed scheme was validated using the formal verification security protocol tool known as AVISPA and the results shown that the scheme has security against MiTM, replay attacks, and masquerade.

Aman et al. [6] did research to implements token-based security for IoT with the energy tradeoff that performed dynamically. The proposed scheme utilizes the authentication framework of OAuth 2.0 to provide authentication on large-scale networks. Authentication based on the OAuth 2.0 framework requires every sensor device to do authentication with the authorization server first before start accessing another resources or devices. To authenticate itself, the sensor device sends its ID and a random number to the authorization server. This ID will be checked by the authorization server, if the ID is not found in the memory, the request will be rejected. Otherwise, if the ID is found, the authorization server will generate an access token with predefine scope and time-limited session and sent it back to the sensor. This access token then will be used by sensor devices to authenticate among other devices. Based on the experimental results, the proposed security scheme is secure against different kind of attacks and able to reduce the power consumption up to 69% for authorization and authentication process, and up to 45% for data communication.

Most of the previous research that provides an authentication scheme for IoT sensor device more focus on utilizing a device authenticator. Even though the scheme has proved to be secure and feasible to be implemented, it requires more resources, in this case, a less constraint device that manages the authentication mechanism for all devices. Centralized authentication also has the disadvantages of being a single point of failure [23]. When the resources for the authentication device is unavailable for some reasons such as system failure or cyber attacks, then all the devices that depend on their authentication will not operate accordingly. Therefore, in this research, we propose an authentication scheme that occurred when cryptography key exchange process to provide authentication as well as confidentiality of the data without using a dedicated device authenticator.

## 3. Proposed Work

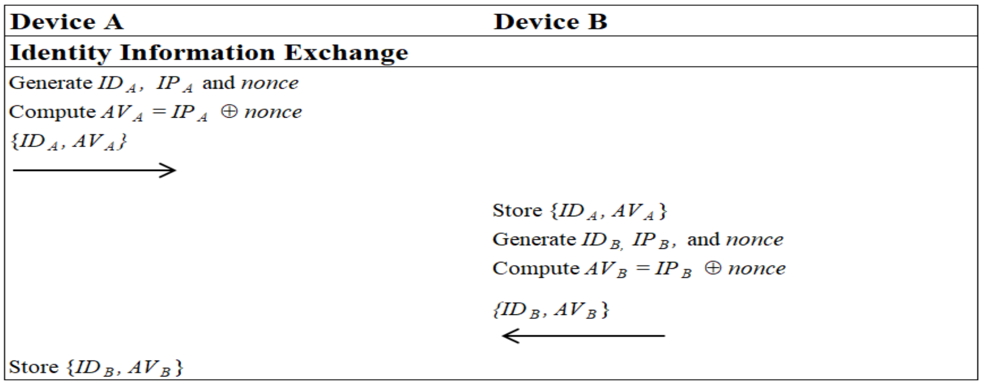
In this section, the proposed authentication scheme for key exchange protocol is described as part of stage three on our research methods which is the protocol design. The proposed scheme consists of three stages: identity's information exchange, authenticated key exchange, and secure data communication. In phase 1, the device's identity ($ID_A$, $ID_B$, $AV_A$, $AV_B$) are exchanged over the auxiliary channel. This identity information will be used later on the key exchange process to proves the device identity. The key exchange process will be resulting in the same secret key on both devices. This secret key then derived and used as AES's key for secure data communication. Some notations used in this paper are defined in Table 2.

Table 2. The Proposed Scheme's Notations.

| Notation | Description |
|---|---|
| $ID_A$ | Device A hostname |
| $ID_B$ | Device B hostname |
| $IP_A$ | Device A IP Address |
| $IP_B$ | Device B IP Address |
| *Nonce* | Pseudorandom numbers |
| $AV_A$ | Device A authentication variable |
| $AV_B$ | Device B authentication variable |
| $\alpha$, $Qa$ | Device A ECDH keypair |
| $\beta$, $Qb$ | Device B ECDH keypair |
| $C_A$, $C_B$ | The result of XOR operation between public key and authentication variable |
| $H_A$, $H_B$ | The hash of the authentication variable |
| $SK_A$, $SK_B$ | Shared Secret key |
| *AESkey* | AES-128 encryption key |
| $\oplus$ | XOR operation |
| $F_p$ | Prime finite field |
| $G$ | Base point |

### *A. Identity Information Exchange*

In this stage, the two devices will send each other identity's information using the auxiliary channel as shown in Figure 2. This is to ensure the identity information ($ID_A$, $ID_B$, $AV_A$, $AV_B$) are known to both devices. The *ID* is taken from the device hostname, and the *AV* is the result of exclusive OR operation between IP address and a nonce (random number). Device A will initiate the exchanges by sending its identity to the Device B. Device B will receive the identity and store it temporarily in the memory. Device B then sent back its own identity to the Device A. Both devices will keep this identity to be used in the authentication mechanism.


Device A
Device B
Identity Information Exchange
Generate $ID_A$, $IP_A$ and *nonce*
Compute $AV_A = IP_A \oplus nonce$
$\{ID_A, AV_A\}$
Store $\{ID_A, AV_A\}$
Generate $ID_B$, $IP_B$, and *nonce*
Compute $AV_B = IP_B \oplus nonce$
$\{ID_B, AV_B\}$
Store $\{ID_B, AV_B\}$
Figure 2. Identity Information Exchange.

*B. Authenticated Key Exchange*

In this stage, the proposed authenticated key exchange scheme is developed from the ECDH key exchange protocol combined with the identity-based authentication. The device's identity consists of identifier and authentication variable (*AV*) that have been exchanged in the previous stage. The proposed key exchange scheme contains several steps as shown in Figure 3 below.

| **Device A** | **Device B** |
|---|---|
| **Authenticated Key Exchange** | |
| Choose random integer $\alpha$, $\alpha\ \Sigma\ F_p$ | |
| Compute $Qa = \alpha * G$ | |
| Compute $C_A = (Qa \oplus AV_A)$ | |
| Compute $H_A = Hash(AV_A)$ | |
| $\{ID_A, C_A, H_A\}$ | |
| $\longrightarrow$ | |
| | Check if $H_A = Hash(AV_A)$ |
| | If so, extract $Qa = (C_A \oplus AV_A)$ |
| | Choose random integer $\beta$, $\beta\ \Sigma\ F_p$ |
| | Compute $Qb = \beta * G$ |
| | Compute $SK_B = \beta * Qa$ |
| | Compute $C_B = (Qb \oplus AV_B)$ |
| | Compute $H_B = Hash(ID_A \oplus SK_B)$ |
| | $\{ID_B, C_B, H_B\}$ |
| | $\longleftarrow$ |
| Extract $Qb = (C_B \oplus AV_B)$ | |
| Compute $SK_A = \alpha * Qb$ | |
| Compute $H_{B'} = Hash(ID_A \oplus SK_A)$ | |
| Check if $H_B = H_{B'}$ | |
| If so, $SK_A = SK_B$ | |
| $\{ID_A, H_{B'}\}$ | |
| $\longrightarrow$ | |
| | Check if $H_{B'} = H_B$ |
| | If so, $SK_A = SK_B$ |
| Compute $AESkey = Hash(SK_A)$ | Compute $AESkey = Hash(SK_B)$ |

Figure 3. Authenticated Key Exchange.

| | |
|---|---|
| **Known Variable:** | $Ep$, $\alpha$, $\beta$, $G$, $AV_A$, $AV_B$ |
| **Step 1:** | Device A generates its ECDH key pair ($\alpha$ and $Qa$). |
| **Step 2:** | Device A computes additional variable $C_A$ and $H_A$. $C_A$ value is computed from exclusive OR operation between the $Qa$ and Device A authentication variable. The $H_A$ value is computed from the hash function of the Device A authentication variable. |
| **Step 3:** | Device A sends its *ID*, $C_A$ and $H_A$ to Device B. |
| **Step 4:** | Device B receives and verify if the $H_A$ value is correct or not by also calculating its own $H_A$ and did the value comparison. |
| **Step 5:** | If verified, Device B extracts the Device A $Qa$ by doing exclusive OR operation of $C_A$ and $AV_A$. |
| **Step 6:** | Device B generates its ECDH key pair ($\beta$ and $Qb$). Device B also compute secret key $SK_B$ from the multiplication of $\beta$ with Device A $Qa$. |
| **Step 7:** | Device B computes additional variable $C_B$ and $H_B$. The $\mathrm{C_B}$ value is computed from the exclusive OR operation of $Qb$ with the Device B authentication variable. The $H_B$ value is computed from the hash of exclusive OR operation between the ID of Device A and the secret key of Device B. |
| **Step 8:** | Device B sends its *ID*, $C_B$, and $H_B$ to Device A. |
| **Step 9:** | Device A receive and extract Device B $Qb$ from exclusive OR operation of $C_B$ and $AV_B$. Device A will also compute its secret key $SK_A$ from the multiplication of $\alpha$ with Device B $Qb$. |
| **Step 10:** | At this point, Device A and Device B has the same shared secret key, $SK_A = SK_B$. To verify it, Device A computes $H_{B'}$, which the result of $hash(ID_A \oplus SK_A)$ and compare it with the received $H_B$. In this step, Device A also sends its *id* and $H_{B'}$ to the Device B so it can be verified if they share the same secret keys. To provide FPS, this secret key will be hashed before using it as an encryption on the AES. |

The usage of XOR operation aims to secure the $Qa$, $Qb$ and $AV_A$, $AV_B$ value. The variables that known to public are $C_A$ and $C_B$. $C_A$ value is computed from $Qa \oplus AV_A$ operation. $C_B$ value is computed from $Qb \oplus AV_B$ operation. To be able to derive $Qa$ and $Qb$ value from $C_A$ and $C_B$, the attacker needs to know the $AV_A$ and $AV_B$ value, $Qa = C_A \oplus AV_A$, $Qb = C_B \oplus AV_B$. To be able to derive $AV_A$ and $AV_B$ value from $C_A$ and $C_B$, the attacker needs to know the $Qa$ and $Qb$ value, $AV_A = C_A \oplus Qa$, $AV_B = C_B \oplus Qb$. $Qa$, $Qb$ and $AV_A$, $AV_B$ values are only known to both parties that want to do the key exchange. The authentication of key that being exchanged is provided by the $Hash(AV_A)$ and $Hash(ID_A \oplus SK_A)$ operations. The authentication variable $AV_A$ only known by both parties. Even though the attacker can sniff the $H_A$ value sent in the public channel, the $AVA$ cannot be retrieved since the value has been secured by the hash function. When attacker try to perform the impersonation attack by sending its own public key to both of the devices, the attacker needs to calculate the correct $C_A$, $C_B$, $H_A$, and $H_B$ value that only can be known if the hash value which is one-way function can be reversed. The security of the SHA-256 hash function still strong and did not have collisions found yet [24]. In this phase, the secret key that is resulted from the key exchange process did not use directly for encryption. The secret key is derived first using the hash function to provide PFS, which is a property of security protocol that ensuring the secret key will not be able to derive from the compromised private key [25].

## *C. Secure Data Communication*

Secure data communication can be achieved after both parties have the same encryption key. In this stage, the sensor data retrieved by the Device A will be encrypted using symmetric cryptography algorithms AES. The selection of a symmetric algorithm for encryption is due to its faster performance compared to the public key encryption [15]. The selection of AES as the encryption algorithm is simply due to its acknowledgment as the world's standard encryption algorithm since 2001. The encryption process on this phase is shown in Figure 4. The AES encryption accepts input in the form of sensor's data, encryption key, and the Initialization Vector (IV) and produces the encrypted data. IV is a number generated randomly using a certain pseudorandom function that is available on the implemented software. The usage of AES encryption with IV aims to avoid statistical cryptoanalysis. IV is often utilized when the plaintext value tends to be constant and slightly different. IV must be unique to each encryption process to avoid the repetition during the data encryption process. IV can be kept public or private. IV will be mixed first with the plaintext before the encryption process occurs to ensure the randomness of the generated ciphertext. By implementing AES with IV, the ciphertext will be completely different even though the plaintext value is identic.

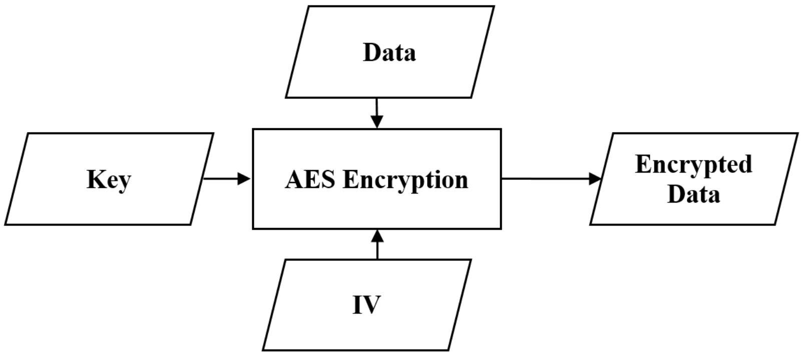


Figure 4. Secure Data Communication.

## 4. Result and Discussion

This section describes the evaluation process of the proposed work that part of stage four on our research methods. In this stage, the proposed protocol will be evaluated using four evaluations: protocol resistance, formal verification of the proposed protocol, security testing, and perfomance testing.

### *A. Protocol Resistance*

Protocol resistance is used to analyze the ability of protocol on preventing the potential attacks. In this testing, the protocol resistance is evaluated by doing security analysis against common security threats on the IoT sensor layers. The common security threats used on this testing is based on the research did by Burhan et al. [26] as follows.

a) Eavesdropping

Eavesdropping is one of the typical security threats in the wireless sensor networks where the unauthorized party intercept the communication between both authorized parties to steal the information. This threat takes advantage of unsecured communication to read communication data. In the proposed protocol, all the data collected by the sensor devices are encrypted using AES symmetric encryption with 128-bit key length *(ciphertext = AES128(data, IV, key))*. Even though the attacker can retrieve the encrypted data that is sent on the public channel, the attacker still needs the key to decrypt the data. If the attacker attempts to brute-force the keys, there are $2^{128}$ number of possible keys that make it impractical to achieved [15].

b) Node Capture

This kind of threat happens when the attacker gains physical access and taking full control over a certain node. The attacker then may try to retrieve the encryption key or other important information stored in the memory [27]. In the proposed protocol, the resilience against this kind of threats is provided by using the short-term key that did not hardcoded in the device's memory *(Hash(*$SK_A$*)* and *Hash(*$SK_B$*))*. This key is generated from the key exchange scheme and renewed on each session. Therefore, even though the attacker may gain physical access to the device, the encryption key is still secure.

c) Fake Node and Malicious

This security threat happens when the attacker injecting a new device or node into the system and then using it to send fake and malicious data. To prevent this kind of threats, the proposed protocol requires two devices to do authenticated key exchange before able to send any data. Suppose the attacker wants to send the key exchange request to Device B, the attacker also needs to send its authentication variable $AV_C$. Since Device B cannot verify the $AV_C$ value, the request will be rejected. Another scenario when the attacker impersonating Device A and want to send its own *Qa*. Even though the attacker can retrieve the value of $H_A = Hash(AV_A)$ transmitted over the public channel, the attacker still needs the original value of $AV_A$ to be able to calculate the acceptable $C_A = (Qa \oplus AV_A)$. Finding the $AV_A$ value from $H_A$ is known as hash collision and SHA-256 still did not have any collision found [24].

d) Replay Attack

This security threat occurs when the attacker eavesdrops all the conversation. In this case, the authentication data will be used later with the purpose of impersonating. This threat is prevented by using the different authentication variable *(AV)* for each session. The *AV* value is computed from the exclusive OR operation between the IP address and a nonce or random number. Suppose the attacker sends the *AV-n* on the *n+10* key exchange process, the *AV* will be invalid since the $AV_n \mathrel{!=} AV_{n+10}$.

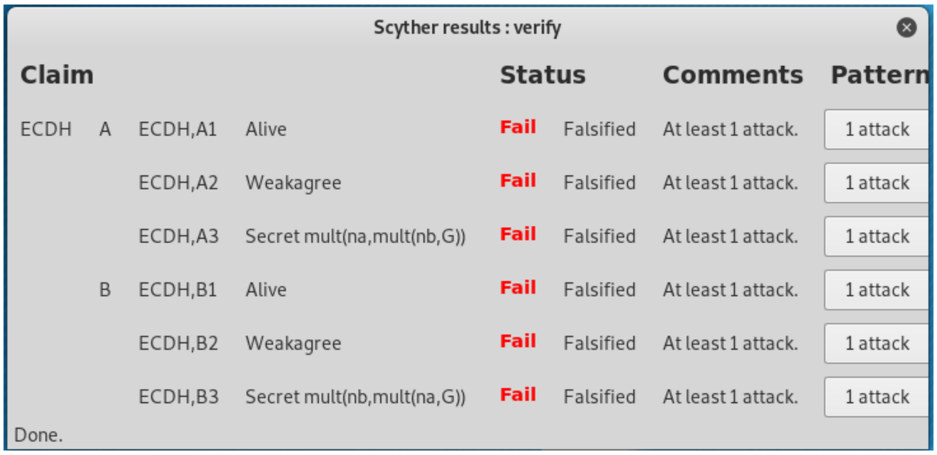
e) Timing Attack

This is a kind of side-channel attack that analyses the execution time of cryptographic algorithms. Every operation takes a varied certain time to execute. By analyzing the execution time, the attacker may try to deduce the plain text or even the encryption key. This kind of attack usually occurs on the public-key cryptography [15]. In this attack, the attacker sends arbitrary data to the target machine and then analyze the decryption time taken by the target. The proposed protocol requires any devices to authenticate itself first before able to send any valid data. This kind of authentication help on preventing the timing attacks.

### *B. Formal Verification*

In this section, the proposed key exchange protocol will be validated using the formal verification security protocol tool known as Scyther. Many kinds of research on developing a security protocol or scheme utilize the Scyther to detects the potential attack and verify the correctness of the protocol [28]-[30]. Scyther accepting input in the form of a description of the protocol written in Specification and Description Language (SPDL). Each participant in the protocol is defined with roles that contain a sequence of events such as sending or receiving terms [31]. The security property of the protocol can be verified using Scyther claim events. Scyther support two security properties claims which are secrecy and authentication [32].

The formal verification testing was performed by first translating the current existing protocol (ECDH) and proposed protocol specification into the SPDL language. The SPDL then will become the input on the Scyther tools that will evaluate and detect any potential attacks. In this evaluation, the secrecy properties claim is verified using the Secret event to verify the secrecy of the AES encryption key. The events used to claim the authentication properties are Aliveness and Weakagree. Aliveness is the weakest form of authentication that only ensures the communication party is alive [33]. Weakagree is the authentication claim that ensures the communication party is alive and there is an interaction between each other. Figure 5 shows the validation results of the ECDH key exchange protocol that does not provide the authentication scheme. Scyther successfully detects the potential attack on the authentication that is shown by the Fail status. On the other hand, Figure 6 presents the validation results of the proposed key exchange with identity-based authentication. The results show the potential attacks on the authentication can be eliminated by providing an authentication mechanism.

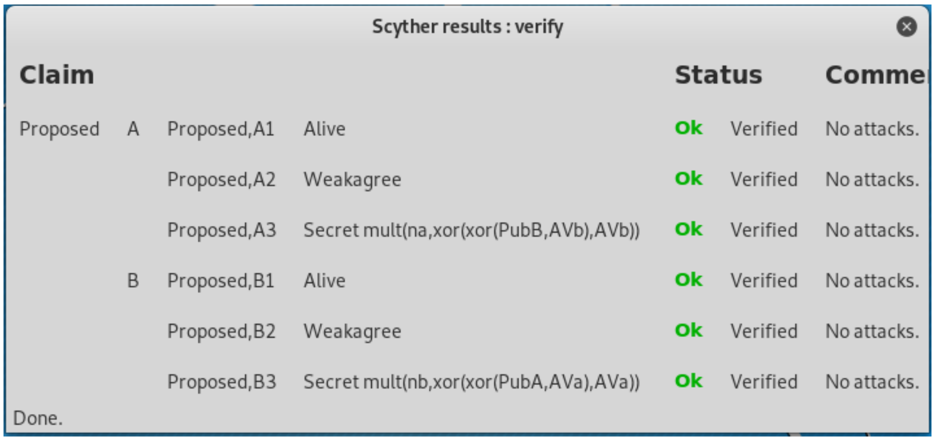


Scyther results : verify

| Claim | | | | Status | | Comments | Pattern |
|---|---|---|---|---|---|---|---|
| ECDH | A | ECDH,A1 | Alive | Fail | Falsified | At least 1 attack. | 1 attack |
| | | ECDH,A2 | Weakagree | Fail | Falsified | At least 1 attack. | 1 attack |
| | | ECDH,A3 | Secret mult(na,mult(nb,G)) | Fail | Falsified | At least 1 attack. | 1 attack |
| | B | ECDH,B1 | Alive | Fail | Falsified | At least 1 attack. | 1 attack |
| | | ECDH,B2 | Weakagree | Fail | Falsified | At least 1 attack. | 1 attack |
| | | ECDH,B3 | Secret mult(nb,mult(na,G)) | Fail | Falsified | At least 1 attack. | 1 attack |

Done.

Figure 5. The result of ECDH key exchange.



Scyther results : verify

| Claim | | | | Status | | Comme |
|---|---|---|---|---|---|---|
| Proposed | A | Proposed,A1 | Alive | Ok | Verified | No attacks. |
| | | Proposed,A2 | Weakagree | Ok | Verified | No attacks. |
| | | Proposed,A3 | Secret mult(na,xor(xor(PubB,AVb),AVb)) | Ok | Verified | No attacks. |
| | B | Proposed,B1 | Alive | Ok | Verified | No attacks. |
| | | Proposed,B2 | Weakagree | Ok | Verified | No attacks. |
| | | Proposed,B3 | Secret mult(nb,xor(xor(PubA,AVa),AVa)) | Ok | Verified | No attacks. |

Done.

Figure 6. The result of proposed key exchange.

*C. Security Testing*

In this section, the proposed key exchange protocol will be implemented and tested to prove its authentication mechanism and data confidentiality. The minimum hardware requirements for the proposed protocol are shown in Table 3. In this testing, the proposed protocol is implemented on the NodeMCU as the sensor device and Raspberry Pi as the gateway. The device's specifications are shown in Table 4 and Table 5. The topology and the implementation for this security testing is shown in Figure 7 and 8. To prove the authentication mechanism, an unauthenticated device will generate its own ECDH key pair and send the key exchange request to the Raspberry Pi. The Raspberry Pi that accepts the unauthenticated device's request then verifies the request. Since the Raspberry Pi cannot verify the unauthorized device's authentication variable, the key exchange request will be ignored and fail as shown in Figure 9.

Table 3. The minimum hardware requirements for protocol implementations.

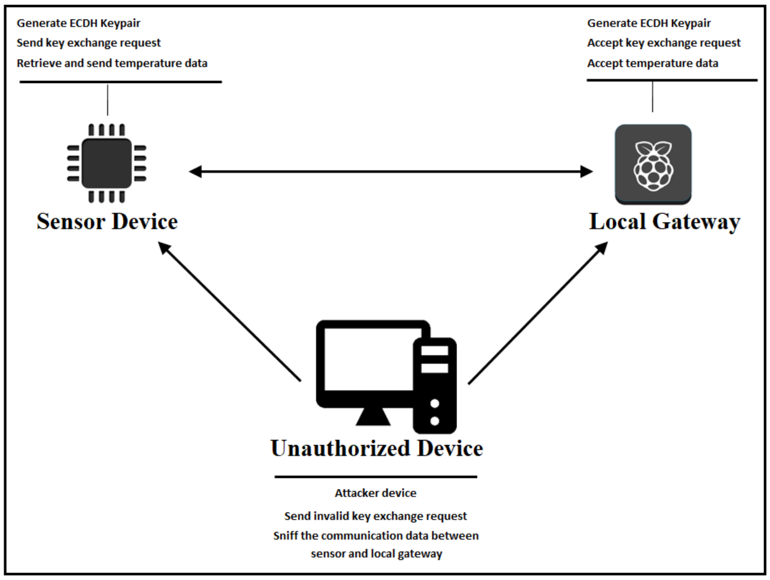
| Components | Minimum Requirements |
|---|---|
| Processor | 8 bit |
| Flash Memory | 300 KB |
| Communication Channels | 2 |
| Supported Curves | Secp160r1, secp192r1, secp224r1, secp256r1, and secp256k1 |

Table 4. The sensor device hardware specifications.

| Components | Specifications |
|---|---|
| MCU | Tensilica 32-bit RISC CPU Xtensa LX106 |
| Bluetooth | HC-05 Bluetooth v2.0 (External Component) |
| Wi-Fi | 802.11 b/g/n |
| SRAM | 64 KB |
| DRAM | 80 KB |
| Flash Memory | 4 MB |
| Frequency | 80 MHz |

Table 5. The local gateway hardware specifications.

| Components | Specifications |
|---|---|
| CPU | 1.4 GHz ARM Cortex-A53 |
| Bluetooth | BLE 4.2 |
| Wi-Fi | 802.11 b/g/n/ac |
| RAM | 1 GB |
| Storage | Micro SD |
| Operating System | Linux |


Generate ECDH Keypair
Send key exchange request
Retrieve and send temperature data
Generate ECDH Keypair
Accept key exchange request
Accept temperature data
Sensor Device
Local Gateway
Unauthorized Device
Attacker device
Send invalid key exchange request
Sniff the communication data between sensor and local gateway
Figure 7. Security testing topology.

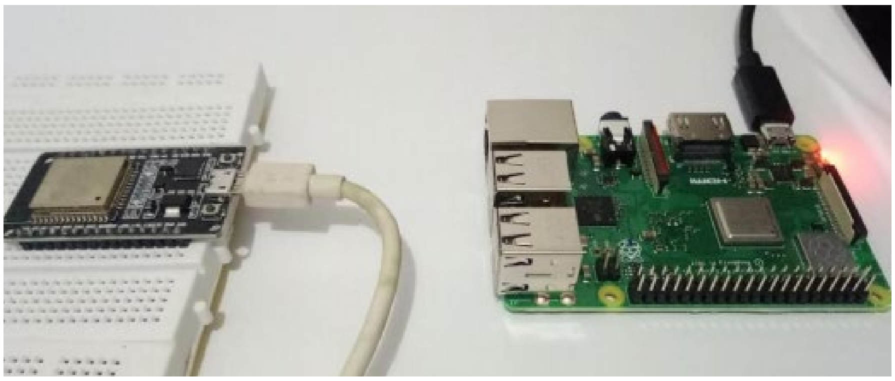
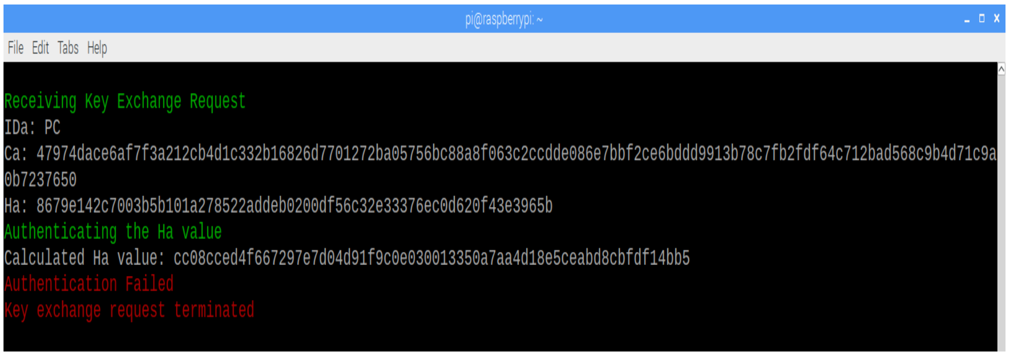
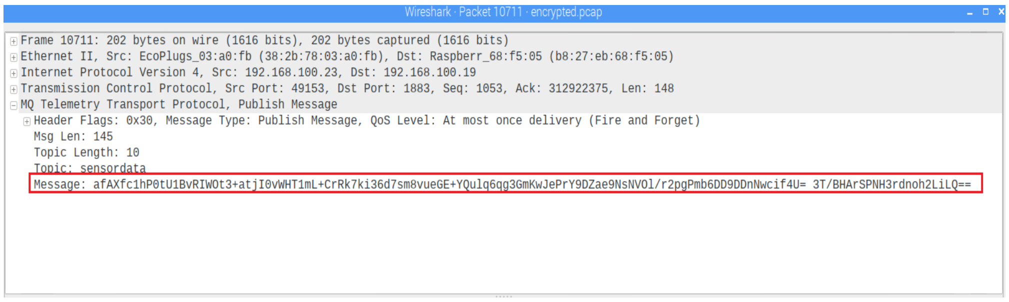


Figure 8. Implementation.



```
Receiving Key Exchange Request
IDa: PC
Ca: 47974dace6af7f3a212cb4d1c332b16826d7701272ba05756bc88a8f063c2ccdde086e7bbf2ce6bddd9913b78c7fb2fdf64c712bad568c9b4d71c9a
0b7237650
Ha: 8679e142c7003b5b101a278522addeb0200df56c32e33376ec0d620f43e3965b
Authenticating the Ha value
Calculated Ha value: cc08cced4f667297e7d04d91f9c0e030013350a7aa4d18e5ceabd8cbfdf14bb5
Authentication Failed
Key exchange request terminated
```

Figure 9. Authentication mechanism testing result.



```
⊞ Frame 10711: 202 bytes on wire (1616 bits), 202 bytes captured (1616 bits)
⊞ Ethernet II, Src: EcoPlugs_03:a0:fb (38:2b:78:03:a0:fb), Dst: Raspberr_68:f5:05 (b8:27:eb:68:f5:05)
⊞ Internet Protocol Version 4, Src: 192.168.100.23, Dst: 192.168.100.19
⊞ Transmission Control Protocol, Src Port: 49153, Dst Port: 1883, Seq: 1053, Ack: 312922375, Len: 148
⊟ MQ Telemetry Transport Protocol, Publish Message
   ⊞ Header Flags: 0x30, Message Type: Publish Message, QoS Level: At most once delivery (Fire and Forget)
     Msg Len: 145
     Topic Length: 10
     Topic: sensordata
     Message: afAXfc1hP0tU1BvRIWOt3+atjI0vWHT1mL+CrRk7ki36d7sm8vueGE+YQulq6qg3GmKwJePrY9DZae9NsNVOl/r2pgPmb6DD9DDnNwcif4U= 3T/BHArSPNH3rdnoh2LiLQ==
```

Figure 10. Data confidentiality testing result.

To prove the data confidentiality, the unauthorized device will capture or sniff the network traffic between the NodeMCU and Raspberry Pi using Wireshark tool. Figure 10 shows the traffic captured using Wireshark tool. The MQ Telemetry Transport Protocol is the name of transport protocol (OSI Layer 4) used by the NodeMCU to exchange data with the Local Gateway. Msg Len is the number of characters in the encrypted message which is 145 characters. Topic and topic length are the name of MQTT topic and the length. The Message field contains the data that being transferred using the MQTT protocol, in this case, the sensor data. In this evaluation, the NodeMCU will send the temperature and humidity data to the Raspberry Pi. Figure 10 shows the traffic that has been sniffed is completely encrypted and cannot be read without decrypting it. Since the unauthorized device did not have the encryption key, the decryption cannot be performed. Therefore, the proposed protocol has proved to secure the confidentiality of the sensor data. Table 6 shows the parameters used to implement the proposed key exchange protocol in this evaluation. The comparison between the authentication mechanism offered on the proposed protocol with the existing protocol that utilizing dedicated devices can be seen on Table 7. Both protocols offer the confidentiality, integrity, and data authentication. The differences between the two protocols lie on the authentication mode, communication channel and authenticated device requirement. The proposed protocol use point-to-point authentication which does not require a dedicated device as authenticator, however it requires two communication channels to exchange keys and data.

Table 6. Parameters length.

| Parameter Name | Size (bits) |
|---|---|
| $AV_A$, $AV_B$ | 256 |
| $Qa$, $Qb$ | 512 |
| $\alpha$, $\beta$ | 256 |
| $SK_A$, $SK_B$ | 256 |
| *AESkey* | 128 |

Table 7. Protocol Comparison.

| Factors | Dedicated device authenticator | Proposed Protocol |
|---|---|---|
| Confidentiality | Yes | Yes |
| Integrity | Yes | Yes |
| Authentication | Yes | Yes |
| Single point of authentication | Yes | No |
| Require Multi Channel | No | Yes |
| Require Dedicated Authenticator | Yes | No |

*D. Performance Testing*

This section presents the results of some experiments to evaluate the performance of the authenticated key exchange protocol in terms of time complexity and memory complexity. These parameters were measured based on the processes between NodeMCU as the sensor device and Raspberry Pi as the gateway. In this research, time complexity is a function that describes the amount of time needed to process the key exchange protocol. On the other hand, memory complexity is a function that describes the amount of memory needed to store the program, those are firmware size for NodeMCU and compiled code size for Raspberry Pi, and the amount of memory needed to execute the program.

From the experiments, the processes of key exchange protocol between two devices were run for 50 iterations. The average time needed to execute the key exchange protocol is 2284.65 ms. This amount of time consists of 115.14 ms for identity exchange, 2144.24 ms for key exchange, and 25.27 ms for encrypted data exchange. For the memory complexity, it takes 297 KB of memory to install the firmware into NodeMCU and 7.6 KB for the compiled program in the Raspberry Pi. The memory used to execute the programs is described as follows. To execute all the functions in NodeMCU, it requires 33.8 KB of memory. This memory is used to store the local and global variables. And for the Raspberry Pi, 12.88 KB of memory is required to execute variable exchange function, 14.52 KB is required to execute key exchange function, and 12.86 KB is required to execute the receiving and data decryption.

## 5. Conclusion

In this research, the key exchange protocol based on ECDH is enhanced with an identity-based authentication mechanism to provide data confidentiality, integrity and authentication for IoT sensor layers. The proposed protocol implements point-to-point authentication instead of single point of authentication. This authentication mechanism does not require a centralized device authenticator, however it requires multi-channel communication. The proposed protocol requires two communication channels, the main channel and the auxiliary channel. The main channel is used for key exchange and data communication. The auxiliary channel is used only to exchange identity information prior to the key exchange process. This identity information is used on the key exchange process as the authentication variable to provide an authentication mechanism. The data confidentiality is provided by AES using an encryption key derived from the shared secret key. For the evaluation, the protocol resistance was evaluated by doing security analysis against common security threats on IoT sensor layers. The proposed protocol has proved to secure against eavesdropping, node capture, fake node and malicious, replay attack and timing attack. The protocol was also verified by formal verification tool known as Scyther and the results showed that no potential attacks were detected. The proposed protocol was implemented on the NodeMCU and Raspberry Pi. The results on the Raspberry Pi terminal and Wireshark tools show that the proposed protocol successfully provides authentication mechanism and confidentiality of the data. For the future works, there's a research opportunity to study the communication channels on the proposed protocol, both the main and auxiliary channel, that best suits for IoT devices.

## 6. Acknowledgment

The authors would like to thank Mokhamad Arfan Wicaksono for doing part of experiments and collecting data.

## 7. References

[1]. A. Mosenia and N. K. Jha, "A Comprehensive Study of Security of Internet-of-Things," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 4, pp. 586-602, 2017.

[2]. Y. Yang, L. Wu, G. Yin, L. Li and H. Zhao, "A Survey on Security and Privacy Issues in Internet-of-Things," *IEEE Internet of Things J.*, vol. 4, no. 5, pp. 1250-1258, 2017.

[3]. I. Ali, S. Sabir, and Z. Ullah, "Internet of Things Security, Device Authentication and Access Control: A Review," *International Journal of Computer Science and Information Security,* vol. 14, no. 8, August 2016.

[4]. Shivraj V L, Rajan M A, M. Singh, and Balamuralidhar P, "One Time Password Authentication Scheme based on Elliptic Curves for Internet of Things (IoT)", *The 5th IEEE National Symposium on Information Technology: Towards New Smart World,* pp. 1-6, 2015.

[5]. O. Salman, S. Abdallah, I. H. Elhaji, A. Chehab, and A. Kayssi, "Identity-based authentication scheme for Internet of Things", *2016 IEEE Symposium on Computers and Communication,* 2016.

[6]. M. N. Aman, S. Taneja, B. Sikdar, K. C. Chua, and M. Alioto, "Token-Based Security for the Internet of Things With Dynamic Energy-Quality Tradeoff", *IEEE Internet of Things Journal,* 2018.

[7]. L. T. M. Blessing and A. Chakrabarti, *DRM, a Design Research Methodology*. New York, NY: Springer, 2008.

[8]. A. P. R. da Silva, M. H. T. Martins, B. P. S. Rocha, A. A. F. Loureiro, L. B. Ruiz, and H. C. Wong, "Decentralized Intrusion Detection in Wireless Sensor Networks", *Proc. 1st ACM Int. Workshop Qual. Service Amp Security Wireless Mobile Netw,* pp. 16-23, 2005.

[9]. J. Chen, S. Kher, and A. Somani, "Distributed Fault Detection of Wireless Sensor Networks", *Proc. Workshop Dependability Issues Wireless AdHoc Netw. Sensor Netw.*, pp. 65-72, 2006.

[10]. Y. Wang, X. Wang, B. Xie, D. Wang, and D. P. Agrawal, "Intrusion detection in homogeneous and heterogeneous wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 7, no. 6, pp. 698–711, Jun. 2008.

[11]. T. Goyal and V. Sahula, "Lightweight security algorithm for low power IoT devices", *2016 International Conference on Advances in Computing, Communications and Informatics,* pp. 1725-1729, 2016.

[12]. M. Usman, I. Ahmed, M. I. Aslam, S. Khan, and U. A. Shah, "SIT: A Lightweight Encryption Algorithm for Secure Internet of Things," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 1, 2017.
[13]. P. Nandhini and V. Vanitha, "A Study of Lightweight Cryptographic Algorithms for IoT", *International Journal of Innovations & Advancement in Computer Science,* vol. 6, issue 1, 2017.
[14]. G. Gaubatz, J. P. Kaps, E. Ozturk, and B. Sunar, "State of the art in ultra-low power public key cryptography for wireless sensor networks," *Proc. 3rd IEEE Int. Conf. Pervasive Comput. Commun. Workshops*, pp. 146–150, 2005.
[15]. W. Stallings, *Cryptography and Network Security Principles and Practice,* 6th ed. New Jersey: Pearson Education, Inc., 2014.
[16]. M. Amara and A. Siad, "Elliptic Curve Cryptography and Its Applications", *2011 7th International Workshop on Systems, Signal Processing and their Applications,* 2011.
[17]. S. K. Verma and D. B. Ojha, "A Discussion on Elliptic Curve Cryptography and Its Applications", *International Journal of Computer Science Issues,* vol. 9, Issue 1, no. 1, 2012.
[18]. H. Dahshan, "An Elliptic Curve Key Management Scheme for Internet of Things", *International Journal of Applied Engineering Research,* vol. 11, no. 20, pp. 10241-10246, 2016.
[19]. B. Anggorojati, N. R. Prasad, and R. Prasad, "Elliptic Curve Cryptography based Key Management for the M2M Local Cloud Platform", *2016 International Conference on Advanced Computer Science and Information Systems,* pp. 73-78, 2016.
[20]. R. K. Kodali and A. Naikoti, "ECDH based security model for IoT using ESP8266," *2016 International Conference on Control, Instrumentation, Communication and Computational Technologies,* pp. 629-633, 2016
[21]. S. Roy and C. Khatwani, "Cryptanalysis and Improvement of ECC Based Authentication and Key Exchange Protocols", *Cryptography Journal,* vol. 1, Issue 1, 2017.
[22]. Y. Bandung and A. P. Putra, "Development of Key Exchange Protocol to Enhance Security of Voice over Internet Protocol on Mobile Phone," *International Journal on Electrical Engineering and Informatics*, vol. 9, no. 1, March 2017.
[23]. R. O. Sinnott, D. W. Chadwick, T. Doherty, D. Martin, A. Stell, G. Stewart, L. Su, and J. Watt, "Advanced Security for Virtual Organizations: The Pros and Cons of Centralized vs Decentralized Security Models," *Eighth IEEE International Symposium on Cluster Computing and the Grid,* 2009.
[24]. A. Sahu and S. M. Ghosh, "Review Paper on Secure Hash Algorithm With Its Variants", *International Journal of Technical Innovation in Modern Engineering & Science,* vol. 3, Issue 05, 2017.
[25]. S. Mandal and S. Mohanty, "Multi-Party Key-Exchange with Perfect Forward Secrecy", *2014 International Conference on Information Technology,* pp. 362-367, 2014.
[26]. M. Burhan, R. A. Rehman, B. Khan, and BS. Kim, "IoT Elements, Layered Architectures and Security Issues: A Comprehensive Survey", *Sensors Journal,* vol. 18, 2018.
[27]. M. V. Bharathi, R. C. Tanguturi, C. Jayakumar, and K. Selvamani, "Node Capture Attack in Wireless Sensor Network: A Survey", *2012 IEEE International Conference on Computational Intelligence and Computing Research,* pp. 1-3, 2012.
[28]. N. Kahya, N. Ghoualmi, and P. Lafourcade, "Formal Analysis of PKM using Scyther Tool", *2012 International Conference on Information Technology and e-Services,* pp. 1-6, 2012.
[29]. U. U. Rehman and A. G. Abbasi, "Secured Layered Architecture for Session Initiation Protocol based on SIPSSO: Formally Proved by Scyther", *2015 12th International Conference on Information Technology – New Generations,* pp. 185-190, 2015.
[30]. H. Yang, A. Prinz, and V. Oleshchuk, "Formal Analysis and Model Checking of a Group Authentication Protocol by Scyther", *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing,* pp. 553-557, 2016.

[31]. Cas Cremers, *Scyther User Manual,* Helmholtz Center for Information Security (CISPA), 2014.
[32]. D. Basin, C. Cremers, and C. Meadows, *Model Checking Security Protocols*, Springer, 2015.
[33]. V. S. Gorbatov, I. Y. Zhukov, and O. N. Murashov, "Assessment of Threats to the Security of the Cryptographic Authentication Mechanisms of the Monitor Devices of Vehicles", *Breakthrough directions of scientific research at MEPhl: Development prospects within the Strategic Academics Units,* 2017.

**Yoanes Bandung** received the B.E., M.E., and D.E. degrees in Electrical Engineering from Institut Teknologi Bandung (ITB), Indonesia, respectively in 2000, 2002, and 2008. In 2009, he joined the School of Electrical Engineering and Informatics ITB as a lecturer and affiliated with Information Technology Research Group. In 2014, he was a post-doctorate researcher in Department of Computer Science and Engineering, Frederick University of Cyprus through the Erasmus Mundus STRoNG-TiES Program. He is a member of the Institute of Electrical and Electronics Engineers (IEEE). His research interests are in the areas of digital signal processing, multimedia computing and communication, information security engineering, and Internet of Things (IoT).

**Arvandy** graduated from Politeknik Caltex Riau (PCR), Pekanbaru and received Bachelor Degree in Informatics Engineering in 2013. In 2017, he continued his study to Master Degree in Electrical Engineering with specialization in Information Security Engineering and Management on Institut Teknologi Bandung (ITB). In 2018, he obtained a certification from Offensive-Security as the Offensive Security Certified Professional (OSCP). His research interests are in the areas of network and information security engineering.