future internet

MDPI

Article

Secure WiFi-Direct Using Key Exchange for IoT Device-to-Device Communications in a Smart Environment

Zakariae Belghazi [1], Nabil Benamar [2,*], Adnane Addaim [1] and Chaker Abdelaziz Kerrache [3]

[1] Systems Engineering Lab, ADSI Team, ENSA Kenitra, Ibn Tofail University, Kénitra 14000, Morocco; zaka.belghazi@gmail.com (Z.B.); adnane.addaim@uit.ac.ma (A.A.)
[2] School of Technology, Moulay Ismail University of Meknes, Meknes 50050, Morocco
[3] Department of Mathematics and Computer Science, University of Ghardaia, Ghardaia 47000, Algeria; kr.abdelaziz@gmail.com
* Correspondence: n.benamar@est.umi.ac.ma

Received: 31 October 2019; Accepted: 18 November 2019; Published: 2 December 2019

Abstract: With the rapid growth of Internet of Things (IoT) devices around the world, thousands of mobile users share many data with each other daily. IoT communication has been developed in the past few years to ensure direct connection among mobile users. However, wireless vulnerabilities exist that cause security concerns for IoT device-to-device (D2D) communication. This has become a serious debate, especially in smart environments where highly sensitive information is exchanged. In this paper, we study the security requirements in IoT D2D communication. In addition, we propose a novel authentication approach called Secure Key Exchange with QR Code (SeKeQ) to verify user identity by ensuring an automatic key comparison and providing a shared secret key using Diffie-Hellman key agreement with an SHA-256 hash. To evaluate the performance of SeKeQ, we ran a testbed using devices with a WiFi-Direct communication interface. The obtained results depict that our proposal can offer the required security functions including key exchange, data confidentiality, and integrity. In addition, our proposal can reach the same security performances as MANA (Manual Authentication) and UMAC (Universal-Hashing Message Authentication Code) but with 10 times fewer key computations and reduced memory occupancy.

Keywords: Internet of Things; D2D communication; WiFi-Direct; smart environments; man-in-the-middle attack; key exchange

1. Introduction

The proliferation of smart objects and their connectivity have contributed to the realization of Internet of Things (IoT) paradigm; their features and services have revolutionized our daily life. These smart devices are equipped by different means of communication to access the Internet [1], creating what it is called the smart environment. Different technologies such as WiFi, Bluetooth, NFC, and recently WiFi P2P (WiFi-Direct) can be used to enable IoT device-to-device (D2D) communication and providing more functionalities to enhance the user experience [2].

Currently, all data traffic must pass through a fixed infrastructure in the different wireless cellular network generations, regardless of the device's location. However, to avoid overloading the infrastructures, it is better to enable direct inter-device communication when possible. When devices are in proximity of each other, problems of radio access network load and core network load are observed [3]. One of the methods used to overcome these limitations and effectively manage local group

communication while offloading data from the cellular network is to establish direct communication between the nearby devices, and this kind of communication is called IoT D2D communication.

Unlike the conventional scenario, where cellular operators are the main middleware handling all inter-device communication within their communication range, IoT D2D communication is a new computing paradigm that allows two proximate mobile devices to directly connect and communicate with each other without the presence of any intermediate authority [4].

The IoT D2D concept is directly related to the used wireless technologies. D2D can be implemented with technologies such as Bluetooth, WiFi-Direct, and LTE Direct, to name a few. Each of them is characterized by its maximum transmission distance, maximum data rate, and the devices' discovery mode, as summarized in Table 1.

Table 1. Comparison of wireless technologies.

Technology	Bluetooth	WiFi-Direct	LTE Direct
Range	100 m	200 m	500 m
Bandwidth	24 Mbps	250 Mbps	13.5 Mbps
Discovery Mode	Manual Pairing	ID Broadcast	Service Broadcast

In this work we consider the use of WiFi P2P technology known as WiFi Direct due to its high bandwidth value with 250 Mbps surpassing the LTE Direct and Bluetooth [5], also for its agreeable range which can expand to 200 m, and finally, for the easiest way to discover neighbors for D2D users using the ID Broadcast -the MAC address of mobile devices in most of cases.

Since IoT D2D communications are over wireless channels, which are broadcast in nature, many existing attacks can affect them and exploit their vulnerability. Hence, security solutions ensuring efficient authentication and confidentiality are a must [6,7]. To perform an efficient authentication between users, establishing a shared secret key is an important step. The common solution is the use of the traditional Diffie-Hellman key agreement; however, it suffers from its vulnerability to the man in the middle (MITM) Attack. On the other hand, relying on a hash value of the shared secret key can provide secrecy to the communication protocol, but the size of the information to be compared by IoT D2D users poses a problem. In previous works, the shared secret key must be compared visually or verbally by D2D users. We propose a solution to avoid this human mediation.

In this paper, we propose the Secure Key Exchange with QR Code (SeKeQ) protocol. Our proposal serves as a key establishment protocol between IoT D2D users using Diffie-Hellman for key exchange and SHA256 hash function for digital signature of the shared secret key. Furthermore, our proposal is fully autonomous and does not require human mediation. Finally, we use the generated secret key to ensure data confidentiality by encrypting the communication traffic over wireless channel. The performance evaluation comparing SeKeQ to other existing solutions shows that it can ensure a secure communication with fast key generation and less resource consumption.

The remainder of this paper is organized in the following fashion: Section 2 presents an overview about WiFi-Direct and some cyber-attacks that may affect D2D communication. Afterwards, we discuss the related works in Section 3. Section 4 introduces the proposed protocol called SeKeQ for key exchange with QR Code. Section 5 shows the experimental setup represented by an implementation of our proposaed solution in an Android application. Section 6 summarizes our results and comparison with other proposals and security analysis. Finally, Section 7 concludes the paper.

2. Overview of WiFi-Direct and Cyber Attacks

2.1. WiFi-Direct

WiFi-Direct is also known as the Adhoc mode in WiFi technology allowing two mobile devices to directly communicate without passing by an intermediate access point. However, mobile users must

manage all D2D services such as auditing or logging by themselves. The WFD has inherited a lot of features and advantages from the infrastructure mode. The main feature is that one selected device on the P2P group assumes the Access Point functionalities and named as Group Owner. We list in the following section the main phases of a WFD Standard Group Formation including Discovery mode, Group Formation and WPS Provisioning, all these steps are illustrated in Figure 1.

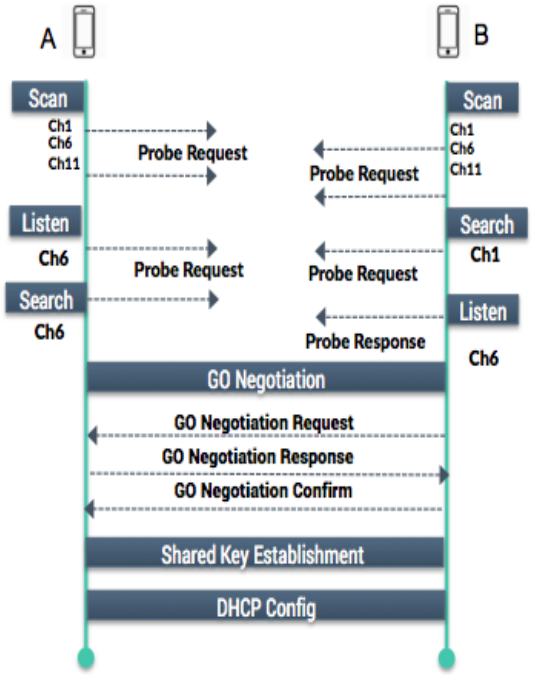

Figure 1. The WiFi-Direct Protocol.

2.2. Cyber Attacks

2.2.1. Device Discovery

This step represents the initialization phase for every mobile device. Each user starts a normal 802.11 scan on its device to find a previously formed P2P group, or for finding WiFi-Direct devices that also want to form a new group. For the purpose, they perform the search and listen phases over wireless channels by sending Probe Requests to get a response by a Probe Response [3]. Once the devices discover each other over the same channel, they move to the Group Formation phase.

2.2.2. Group Formation

At this stage, the P2P groups are not formed yet. All WiFi-Direct group devices implement both roles of client and Access Point. These roles are dynamically assigned during the group formation. This phase begins with 3-way handshake, for the Group Owner negotiation Request, Response and Confirmation. We note that the user with the highest intent value-generally the MAC Address, is elected as Group Owner (GO) [8]. The GO plays the role of an access point to provide clients with connectivity. It is also a communication node and can be a source or a destination of higher layer applications. The P2P Group can be extended with more than 2 devices. Other devices detect the existent of the GO without passing by the discovery phase and join their actual group.

2.2.3. WPS Provisioning

Once the election of GO is done, P2P group devices try to form secure communication by performing the WPS Provisioning phases. The first phase consists of creating security keys using WPS-WiFi Protected Setup-which is based on WPA-2 Security and a PSK-PreShared Key-generated

randomly. In the second phase, devices playing the role of P2P Clients disconnect from the existing group and re-join it using the new calculated credentials [9]. The GO verifies the required network credentials of any P2P client wishing to join the group. However, the WPS is limited to its PSK based on the PIN exchange mechanism composed of eight-digit, which can be predictable within minutes with nowadays computers performances, exploiting the brute-force vulnerability [10]. This flaw makes security improvements needed for secure end to end communication of WFD mobile devices.

2.2.4. IP Address Configuration

After the agreement on the pre-shared key, the P2P device acting as GO, starts a DHCP exchange [11] for automatic IP address allocation of all other group members. The DHCP process begins with a discovery requests from the clients. Then, the P2P GO responds to the request offering an IP address for the client followed by another request from the client choosing one of the offered address. Finally, the P2P GO running the DHCP server, answers and acquits the client. Any external communication is insured by the P2P GO interconnecting all P2P Group clients. In addition, the P2P group is deleted if the P2P GO leaves the group. A new P2P group has to be created selecting a new GO using the Standard Group Formation.

Furthermore, and with the emergence of 5G technology the world became full of connected objects and devices that are always a target of attackers who exploit different wireless vulnerabilities to perform physical [1], configuration and protocol attacks, or even attacks on core network. The insensibility of mobile users can also cause a cyber-attack. In the following, we present some known attacks that can provide data loss, fake identity and channel communication disrupt. Attacks on D2D communication can be classified in 2 types: first, attacks on the D2D system over wireless channel such as Distributed denial of service or Jamming in which the attacker tries to perturb the communication over wireless channel [12] regardless of the D2D users. Second, attacks affecting cellular mobile users or D2D users; such as man in the middle, eavesdropping or even masquerading; the attacker here aims to manipulate legitimate D2D users using multiple methods such as identity theft or social engineering in order to stole users' credentials or even inject false messages to multiple group users.

In addition, an attacker targeting a D2D communication can vary according to the attacker privileges over the wireless channel and what type of attack it is trying to launch [12]. Indeed, a local attacker is already a P2P group member and can communicate with authenticated users. Whereas, an outsider attacker has less privileges, which forces him to first search for potential vulnerabilities to exploit in order to gain access to a specific P2P group. Furthermore, an active attacker applies modifications on the transmitted data such as creating, deleting or replying the data on the wireless channel or on mobile devices to obtain sensitive information. This type of attacker can be quickly detected through the scan of the modifications it makes. However, a Passive attacker is considered as the evilest type of threaders due to its inactivity over the wireless channel, acts by listening and sniffing transmitted traffic between legitimate mobile devices which generally makes it hard to be detected.

2.2.5. DDOS Attack

A distributed Denial of Service attack is a type of attack that is aimed at bringing down the D2D system by flooding wireless channel with generated messages [13]. A hacker or multiple hackers attempt to make wireless channels unavailable to legitimate D2D users by consuming system resources by performing a set of request to the system. The system then is waiting for acknowledgment that causes a denial of service resulting in some occasions to the distraction of the P2P group.

2.2.6. Jamming Attack

This attack always targets D2D systems and is performed over wireless channel due to its broadcast nature. An active adversary who is also mobile user aims at generating a noise signal to overload the channel and perturb group communication between legitimate D2D users, which includes

a loss of the exchanged messages lost [14]. In performing this attack, the jammer tries to reduce the signal-to-noise ratio by transmitting multiple radio signals.

2.2.7. Masquerading Attack

Masquerading attacks are considered as a vulnerability exploited to target D2D users of a specific group. An attacker, generally a mobile user joins the P2P group pretending to be a legitimate and an authenticated user by using others' identity [14]. The attacker first sends a request to one of the P2P group users. Then, the legitimate P2P user responds using its ID broadcast which is the MAC Address of the mobile device. The attacker finally, uses this MAC Address to communicate with other group users and claims to be the legitimate user.

2.2.8. Mitm Attack

In a man in the middle attack (MITM), an active adversary, positioned between legitimate group users succeeds in joining the P2P group and sniffs the traffic transmitted over wireless channels [13]. The attacker sends a message to a P2P user, receives a response, then utilizes these credentials to communicate with other group users. The P2P users can not differentiate between a legitimate and a hacker because of the correct credentials used. MITM attack remains one of the main vulnerabilities of D2D communications, which we try to solve in the current study.

2.2.9. Eavesdropping Attack

It is similar to the MITM Attack, the threader is using the legitimate users' credentials to provide manipulation. Its position in the middle of the wireless communication allows it to read transmitted messages, steal users information, with the addition that the attacker views, steals, modified, and deletes [15].

3. Related Works

Security and Privacy from Mobile Ad-hoc Networks to D2D has been discussed in [14]. Security and Privacy requirements ensure that only authorized group users are able to consume group services, that data exchanged among group users must not be sent in clear nor changed or modified during transmission. Taking into account that D2D communication is mainly based on group concept, the most significant step is a shared secret among group users for key agreement procedure.

The authors in [16] proposed key agreement for P2P based online social networks in a face to face pre-authentication through a location limited channels for identifying legitimate P2P users. They provide a Message Authentication Code as a customized hash function for digital signature ensuring simultaneous users identification for better efficiency and less messages to be authenticated. However, their proposal is limited by the weakness of the old used hash function and the key size. An implementation of [17] has proved that an adversary can bypass the authentication as a normal group user exploiting the man in the middle attack. Another drawback that their hash-based protocol is not cost effective which makes it inadequate for devices with limited resources.

The authors in [18] proposed a key agreement protocol based on Diffie-Hellman and Commitment Scheme. The aim was to overcome the vulnerability of MITM Attack that infects Diffie-Hellman key agreement in a first place, and to settle the comparison of large number of bits (128 bits-32 hexadecimal digits-) as the Hash Value of Diffie-Hellman key. The proposed commitment scheme is secure and provides a 20 bits (5 Hexadecimal digits) as the shared key and is quietly. However, this protocol requires a visual or verbal comparison to be mutually authenticated, which makes it a non-trivial task and remains a critical issue.

Another solution was discussed in [19] where Diffie-Hellman key exchange (DHKE) and HMAC digital signature was proposed to provide authentication and to verify the identity of each user in D2D communication underlying LTE-Advanced networks (LTE-A). They also proposed a confidentiality protocol that encrypts data from source to destination by symmetric key encryption. However, many

intermediarery nodes are involved in the key generation and key exchange processes, namely the the eNB (Evolved NodeB) and the GW (Gateway) that are responsible for this task. Since the channel might be compromised for a direct communication among devices, this solution is not feasible to secure in hostile environment.

In [20] an ID-based AK protocol with signature is proposed. The security of AK protocols has relied on the intractability of the Diffie-Hellman problem and the related problem of computing discrete logarithms. Thereafter, they propose two efficient secure AK protocols satisfying perfect forward secrecy based on tripartite ID-based authenticated key agreement protocol with signature, and smart protocol based on ID-based authenticated key agreement protocol with key confirmation. Although Protocol 1 provides desirable security attributes. The second proposed protocol appears to have the security attributes of known-key security, perfect forward secrecy, unknown key-share and key control. However, both protocols stay limited to the less efficiency on message bandwidth where each entity sends two key components.

The SHC protocol proposed in [21] refers to Short Hash Code represented by a Diffie-Hellman key agreement with MD5 Hash, this protocol takes as an input the DH key exchanged among mobile devices, then provides a message digit composed of 128 bits as an output. However, it stays limited to its hash code length, therefore, using at least 48 bits as the hexadecimal size of hash remains a necessary solution to prevent the man-in-the-middle attack.

In [22] an authentication protocol called MANA algorithm which stands for Manual Authentication, and is based on a conventional MAC function which is HMAC [23] in Key exchange; mobile devices generate Diffie-Hellman parameters with the addition of a commitment represented by a MAC (Message Authentication Code) value on each device and are considered as an output to be displayed on both devices for authentication but it remains unknown to an external attacker. However, its limitation remains on that the key size remains very short (around of 16–20 bits) and can be rapidly predicted within seconds by an adversary due to nowadays huge computers capacity, also this protocol shows its inefficiency on wireless channel with the addition of the high time length during the key exchange the DH parameter with unique MAC value generation on each group device.

The other solution, named UMAC [24] (Universal-Hashing MAC) and it is a one-time authenticator; a long sub-key is calculated once for authenticating mobile users and have not to be revealed to an attacker, it is based on HMAC with SHA1 hash algorithm. However, this solution takes more time for performing authentication credentials among mobile devices due to the HMAC function and its slow nature on generating the message authentication code. In addition, this UMAC occupies memory resources more than the ordinary during key generation by performing the HMAC value in first step followed by the SHA1 with its 160 bits.

The approach we present in the current study outpeforms other proposals in term of security, performance, time lapse and memory consumption. First, the designed SeKeQ protocol is very strong regarding security considerations because of the SHA256 hash algrothim used that is hard to be broken by adversaries with the addition of AES protocol used for encryption of transmitted messages during communication of legitimate users. In addition, the protocol shows the fastest time of key generation and exchange time compared to other algorithms while maintaining an optimized consumption of memory size allocation and CPU usage. All these metrics results the best performance of the SeKeQ protocol compared to SHC, MANA and UMAC.

Our proposed SeKeQ outperforms the existing works in term of security performance, time lapse, and memory occupancy thanks to the use of SHA256 hash with the encryption of transmitted inter-users messages using AES, thus, making SeKeQ hard to break by the different kinds of adversaries. In addition, the protocol shows the fastest time of key generation and exchange compared to the existing works while maintaining an optimized occupancy of memory and acceptable CPU usage.

4. SeKeQ: Secure Key Exchange with QR Code Protocol

Our protocol SeKeQ works as follows; Devices A and B agree on finite group G from Z/pZ with multiplication modulo a prime number p. In addition, users generate random numbers for key exchange on Z/pZ field. They exchange these random numbers modulo p on our key agreement process.

Devices A and B agree in clear over wireless channel on prime large p and a generator g from Z/pZ field, we call them the key pair. They are exchanged as follows, A creates its own Diffie-Hellman key pair with 2048 bits key size and initializes its Diffie-Hellman Key Agreement object. Then, A encodes its public key, and sends it over to B. On device B, it has received A public key in an encoded format, it instantiates a Diffie-Hellman public key from the encoded key material. B gets the Diffie-Hellman parameters associated with A public key. Then, it must use the same parameters when it generates its own key pair. At this stage, both A and B have completed the Diffie Hellman agreement on the key pair p and g.

Then A selects its private random number a and calculates $g^a mod p$ and sends the result publicly to B. Now, B selects its private random number b and calculates $g^b mod p$ and sends this result to A. In the next step, the calculation of the shared secret begins, A takes B public result and raises it to the power of its private number a mod p to obtain the shared secret, called $S1$. Then, B takes A public result and raises it to the power of its private number b mod p resulting on a shared secret called $S2$. It is worth noticing that both devices did the same calculation. As we see, it is an efficient key agreement protocol but still vulnerable to the Man in the Middle attack. Which makes the studies fundamental to enhance the secrecy of the algorithm.

In the next step, and to provide a high security level, Devices A and B put their shared keys $S1$ and $S2$, respectively, to a one way hash function, which is, in our case, SHA-256. They generate $h(S1)$, the hash value of $S1$ for device A and $h(S2)$ the hash value of $S2$ for device B. The hash value is a large number. In our case, the hash value is represented by 32 Hexadecimal digits.

We should obtain $h(S1) = h(S2)$, since $S1$ and $S2$ are equal which is the adequate case for verifying the integrity of the shared secret key. However, the hash value have to be compared manually by D2D devices. The authors in [18] proposed a key agreement based on Diffie-Hellman and commitment scheme. They succeed at reducing the large size of the hash value $h(k)$, resulting 5 Hexadecimal digits as the size of the shared secret key. However, this proposal is still inefficient because it requires that the shared secret have to be compared visually or verbally by D2D users. To overcome this limitation, we propose an automatic key comparison process of the shared secret key by the use of the QR Code as an additional step of our SeKeQ protocol.

The QR code is a type of two-dimensional bar code (or matrix code) consisting of black modules or points arranged in a square with a white background. The arrangement of these points defines the information contained in the code. This information can be a website, text, phone number or an SMS [25]. It is first generated containing the desired data which can go up to 4000 characters, and displayed as a picture on a computer, smartphone or printed on a paper. Then, with nowadays large mobile devices, using the device camera, the QR is scanned and results its content at high speeds. The QR Code have multiple application uses, such as storing information of patients in the medical field, scanning passenger's coordination for accessing a facilities and private zones. In addition, it can be used for reliable verification of products before packaging. Companies use the QR Code for their banners, to make clients scan it to get more information about the publicity, or even it can be used for sending an email, setting a professional appointment and nowadays, stores uses the QR code for allowing their clients to know about a product by scanning a QR and getting more information. In our protocol, it is used to store the hash value of the shared secret key calculated on the P2P GO device.

Automatic key comparison process works as follows, since $H(S_1)$ and $H(S_2)$ should be equal, device A that is the Group Owner will fetch the value of its calculated result $H(S_1)$ into a new generated QR Code. By the following, device B, acting as a P2P client—with the help of the back camera—can scan the generated QR Code that contains the device A result $H(S_1)$ and save it as a new string called N. Now, a system comparison will be in device B, comparing the scanned value N with the value of its result $H(S_2)$. If it matches, the communication between A and B is performed automatically without any verbal or visual comparison and both devices can start sharing their data now. The scenario of key exchange between users is shown in Figure 2.

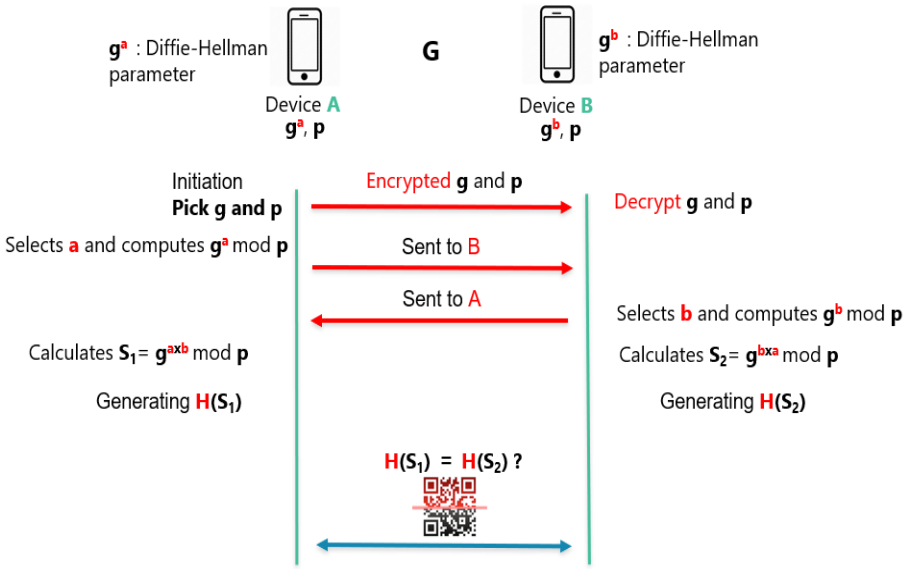

Figure 2. The proposed Secure Key Exchange with QR Code (SeKeQ) protocol.

Providing a shared key as an authentication proposal among devices is securely efficient. However, while we generate the key exchange, we use it to guarantee data confidentiality. Making sure that data is not sent in clear over public channels and avoiding the possibility of a Man in the middle or Eavesdropping attack.

Since the devices are mutually authenticated, End-to-end encryption is used based on the generated shared secret key. Thereafter, we convert the key in an AES key format as a symmetric encryption key from the shared key. If the communication between users is a chat scenario, the messages are encrypted before getting ready to be sent. A user who wants to send a message, uses its shared secret key to provide an AES symmetric key for message encryption. After receiving the message in an encoded format, and while we are using symmetric encryption, which means that the encryption and decryption key are the same. The receiver uses the shared key to generate the AES key and decrypts the message.

5. Experimental Setup

After finishing our protocol design, and to affirm our security analysis, we implemented our SeKeQ protocol on Android. This latter is the most used open source operating system around the world [26]. We developed our application using java language on Android Studio environment based on WiFi-P2P. Furthermore, the used framework is composed of four classes [27]:

5.1. MainAcitity.Java

This class performs these functionality:

- Handles user navigation on the application.
- Manages application display.
- Intermediary between multiple classes.
- Displays the messages on users' devices.
- Performs automatic comparison of the QR Code.

5.2. GO.Java

This is the GO class and performs these functionality:

- Manage communication between GO and other clients.
- Initiates TCP Socket for communication with clients.
- Generates shared secret key on the GO side.
- Generates the QR Code.

5.3. Client.Java

This is the Client class and performs these functionality:

- Manage communication between all clients and the GO.
- Initiates TCP Socket for communication with GO.
- Generates shared secret key on the client side.
- Manage QR Code scan.

5.4. Wifibroadcastreceiver.Java

This class performs these functionality:

- Listens for nearby WFD devices.
- Displays discovered devices.
- Remove disconnected devices.
- Notify MainActivity.java about discovered devices.

The smartphones used for the application test are OnePlus2 with its 3.24 GHz processor and 4 GB of RAM, the second phone was a Samsung Galaxy Note4 with 2.7 GHz processor and 3 GB of RAM. Both smartphones support the Android 5.0 Lollipop version. These two mobile devices supporting WiFi-Direct technology want to communicate with each other and share their data in a highly secure manner. The proposed devices have the capacity to perform shared key computation. When both devices are launching the application, they can discover each other by performing standard group formation, starting by listening and searching over the same channel and beginning a 3-way handshake (Probe Request/Response/Confirmation) to agree on which device is the group owner as illustrated in Figure 1. Then the GO is required to run DHCP exchange (Discover, Offer, Request, ACK) with the client to provide IP address allocation. One device is elected as a GO, generates the shared secret and the QR Code. The other device proceeds also by generating the shared secret and performs the automatic comparison scanning the QR Code with its camera. In addition, based on [28], we enabled WiFi-Direct functionalities using Google public APIs for Android applications. Figure 3 shows the interface of our protocol running over an Android application.

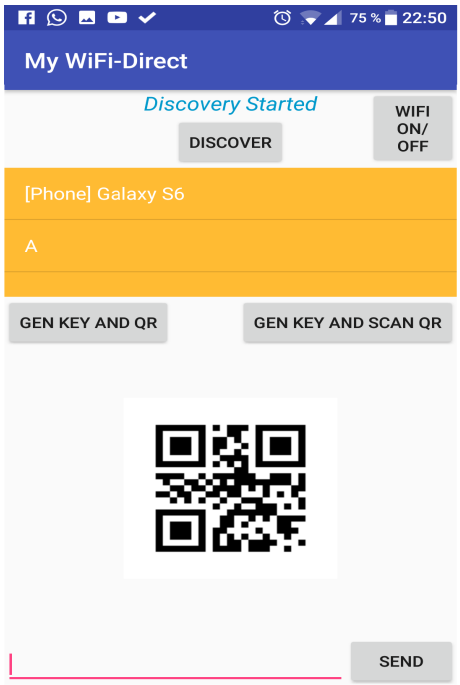

Figure 3. Android interface of the proposed Internet of Things (IoT) device-to-device (D2D) secure communication protocol.

6. Results and Discussion

The design of the key exchange protocol is based on the traditional Diffie-Hellman key agreement protocol, SHA256 hash function for digital signature of the shared secret and AES encryption to provide data confidentiality.

6.1. Security Analysis

Mobile users A and B calculate S_1 and S_2 respectively as the short authentication string for mutual authentication and then, they apply hash function for the shared key.

We assume that another device E is launching a Man In The Middle Attack by pretending to B to be A. First, it listens on the wireless channel and collects g and p, known as The key pair. It selects e and calculates $g^e mod p$. Next, it sends the hash value to B. After receiving B's answer (hash value of $g^e mod p$), E cannot edit hash value and resend it to A. Thus, in the mutual authentication step, A and B compare $H(S_1) = H(g^{a^b} mod p)$ with $H(S_2) = H(g^{b^a} mod p)$ but S_3 is different from S_1 and S_2.

As a result, the key exchange scenario has been performed correctly. To provide confidentiality of all transmitted messages, the calculated shared secret is used to ensure AES encryption generating the symmetric key with a size of 128 bits that is hard to be broken by adversaries.

6.2. Performance Evaluation

To show the efficiency of our proposal, a comparison with the related solutions is needed. To this end, our experiments are based on the study in [29], in which the authors provided a comparison between different key exchange protocols considering different metrics: Performance, Experiments, Disk space and Security.

For an efficient comparison, we implement our SeKeQ protocol with the related solutions. Taking first, the SHC (Short Hash Code) using Diffie-Hellman key agreement with MD5 Hash. Secondly, the MANA (Manual Authentication) algorithm that actually use Diffie-Hellman for key exchange and HMAC for digital signature. The third solution, UMAC (Universal Message Authentication Code); the one-time authenticator based on HMAC with SHA1.

We used Python Scripts for implementing the studied algorithms (SeKeQ, UMAC, MANA and SHC) to compare time lapse, memory size allocation, CPU consumption and network throughput during the calculation and generation of the shared secret key between users. The experimentation for comparison were made on virtual devices and were not implemented for Android OS.Implementation results are shown in Figures 4–8. The figure below shows the cryptographic operations involved in the calculation of the shared key between users of each protocol. Comparing the amount of time for generating keys with the energy consumption of the shared secret with the time needed for generating keys on different bandwidths for each of the studied protocols. Time generation is represented in milliseconds, while size in memory is represented in Megabytes. Whereas, The network throughput is represented in kilobytes per second (Kb/s).

As known, one of the features of key based authentication, their ability to provide smaller key sizes as presented in Figure 5. In addition, their rapidity on generating keys that is done within seconds as illustrated in Figure 4. The MD5 hash used in SHC has 128 bits size of hash, it occupies 7 Mb during its 6 ms for key exchange. The length of the keys with HMAC as used in MANA can not surpass 20 bits, however it needs more than 8 Mb and remains the slowest with 40 ms when exchanging keys. While HMAC based SHA-1 proposed for UMAC algorithm is 160 bits and shows the same value in memory as the MANA while it takes 30 ms for key exchange. Finally, our protocol SeKeQ has 256 bits in length and we can obtain a better power and delay performance by representing 5.8 MB of key size allocation in memory and a perfect time for establishing the shared secret among devices with just 4 ms and shows good values compared to the others. Key size in memory has also an impact on the wireless channel performance, the more it is reduced, the more we get less communication overhead, less data overload, and better bandwidth on exchanging data among cellular devices.

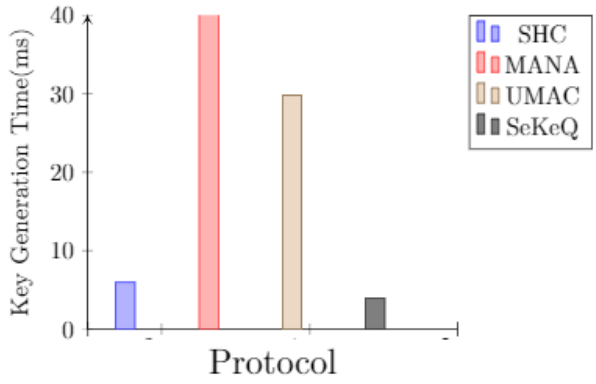

Figure 4. Key Generation Time for one-to-one communication for different protocols.

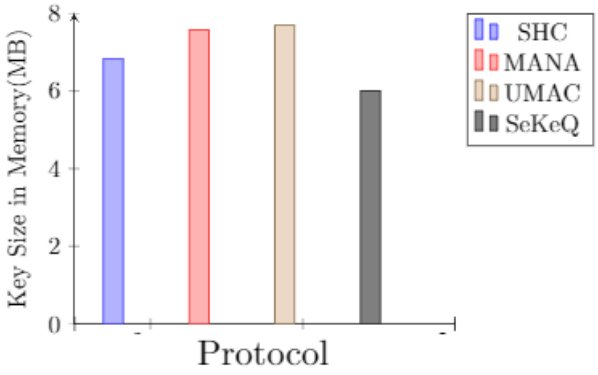

Figure 5. Memory Size Allocation of key generation for different protocols.

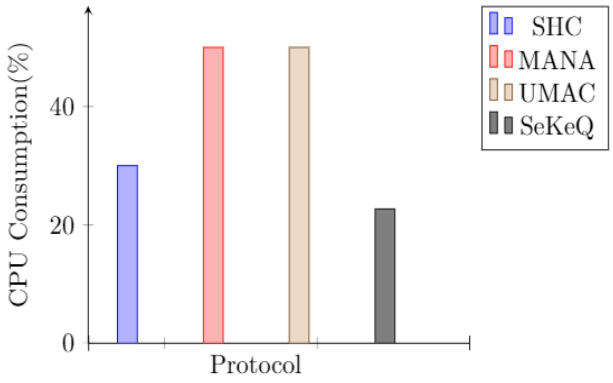

Figure 6. CPU Consumption during key exchange for different protocols.

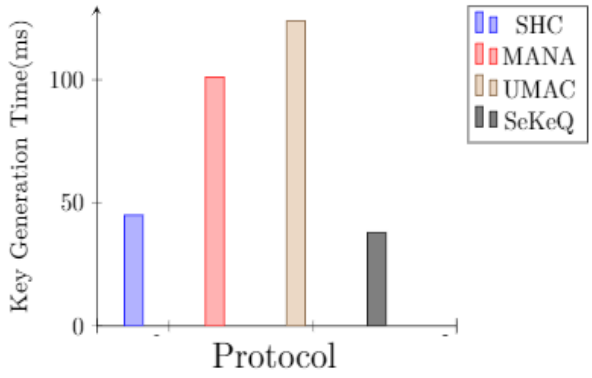

Figure 7. Key Generation time for 3 group users.

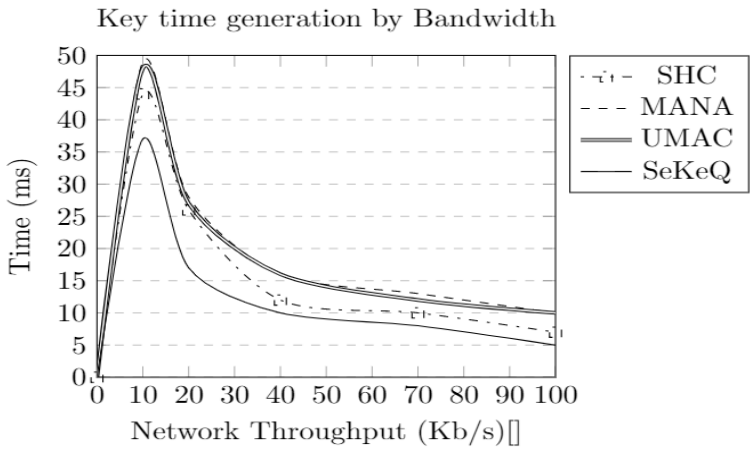

Figure 8. Key Generation Time of different protocols for different Throughputs.

The results of CPU consumption are presented in Figure 6. We can notice that the consumption of our scheme represented by 22.7% is less than CPU usage of SHC how stands with 30%, MANA and UMAC with the large 50% usage. Accordingly, these outperformed results of Time generation, Memory consumption and CPU usage, prove that the key exchange can be performed efficiently on mobile devices.

When fixing the bandwidth from low to high throughput in order to test the proposals in different situations, e.g., in environments with lower bandwidth. Figure 8 shows that the generation time increases when the bandwidth decreases since the data transmission time increases rapidly when network throughput decreases for all protocols. However, the amount of time taken by our proposed SeKeQ protocol in different throughput degrees stays more stable compared to the others and can

enhance largely the achievable data rate on each device. At the initial time, and with lowest throughput 10 Kb/s, all protocols have nearly the same key generation time. However, MANA and UMAC remains slower than the others for key generation even when network throughput increases. SHC results a medium duration of key generation on highest bandwidth.

We extend the comparison for a group with 3 users. First, all 3 users are from the same group. For that, the standard group formation is performed. The device elected as the GO acquits the other 2 clients with IP addresses from the same range. Thereafter, the GO have to establish a shared secret key with each of 2 other clients. In addition, the clients establish a shared secret for a secure direct communication between them. The number of needed keys depends on the following formula:

$$NK = (N(N-1))/2 \quad (1)$$

where N is the number of group users and NK is the total number of keys for this group. In this case, 3 shared secret keys are needed regarding the 3 group users. Figure 7 shows the amount of time elapsed for performing different keys generation. Our protocol SeKeQ surpass SHC, MANA and UMAC. The other protocols still take more time for establishing the shared secret. Due to the large size of hash in UMAC and its 2-key components. MANA also requires high time during generation regarding its inefficiency on wireless channel. Finally, SHC maintains good results with a small key size.

We can notice that the metrics for our protocol compared SekeQ with SHC are nearly equal. However, the research study in [21] proved that an MD5 hash can be broken within seconds. For success, it requires an attacker to use significant resources, i.e., to perform one billion trials in less than 1 s. The time out of mobile users is assumed to be 10 s, which makes it easy for the attacker to, by a performing man in the middle with brute-force attack, prevent the 32 hexadecimal digits as a hash of MD5.

A comparison summary of different protocols is presented in Table 2. Our results show clearly that our protocol SeKeQ outperforms SHC, MANA, and UMAC in terms of time generation of the shared secret maintaining less memory allocation and perfect time generation on different network throughput values. While SHC also has less memory allocation for key generation, it is limited to the brute-force attack, in which an attacker can predict the value of the short hash. MANA is also vulnerable to the man in the middle attack and provides long key generation time. UMAC shows a strong key size but with more key components due to its HMAC digital signature first, followed by SHA in a second step for providing a shared secret key.

Table 2. Comparison with related proposals.

Protocol	Weakness	Strength
SHC	Brute-force, Short Hash	Less memory allocation
MANA	Key generation time, MITM	–
UMAC	More memory allocation, key generation time	Strong key size
SeKeQ	–	Less memory , short key generation time

This discussion forces us to go back to the cyber attacks. Our proposal avoids the man in the middle and eavesdropping attacks due to its efficiency ensuring secure key establishment among devices offering digital signature without giving strange users the ability to access D2D communication. In addition, DDoS and Jamming are settled by allowing only authorized users to use D2D services over wireless channel.

Besides the achieved performance, our proposal can be further improved through the use of a more lightweight hash function. We also plan to make the proposal flexible for large scale networks through the incorporation of Blockchain technology [30].

7. Conclusions

In this paper, we study the IoT D2D communication aspects, considering group formation, peer discovery, and security challenges. Our proposed SeKeQ key exchange allows proximate users to establish a secure communication over public channels and optimize the shared key verification with automatic comparison. We also compared our protocol with other proposals by implementing the different protocols and derived experiments in term of time to generate the exchanged keys, their size consumption in memory, and time elapsed to generate keys on different network throughputs. Data confidentiality is insured through the encryption of all inter-user communications. We also implemented our protocol in an Android application on real smartphones using the WiFi-Direct technology. Experimentation results depict that the proposed protocol is efficient in term of ensuring authentication, data confidentiality, and short communication delay. To the best of our knowledge, the results are efficient, but it will be open to new works and challenges of providing strong key optimization, flexible key exchange, and minimized key generation delay with less memory consumption.

Author Contributions: The authors contributed equally to this work.

Funding: This research received no external funding

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kar, U.N.; Sanyal, D.K. An overview of device-to-device communication in cellular networks. In *ICT Express*; Elsevier: Amsterdam, The Netherlands, 2017.
2. El Alami, M.; Benamar, N.; Younis, M.; Shahin, A.A. A Framework for Enabling Internet Access Using Wi-Fi Peer-to-Peer Links. In *Wireless Personal Communications*; Springer: Berlin, Germany, 2019; pp. 1–18.
3. El Alami, M.; Benamar, N.; Younis, M.; Shahin, A.A. A framework for hotspot support using Wi-Fi direct based device-to-device links. In Proceedings of the 13th International Wireless Communications and Mobile Computing Conference (IWCMC), Valencia, Spain, 26–30 June 2017; pp. 552–557.
4. Asadi, A.; Mancuso, V. WiFi-Direct and LTE D2D in action. In Proceedings of the IFIP Wireless Days (WD), Valencia, Spain, 13–15 November 2013; pp. 1–8.
5. Lee, J.S.; Su, Y.W.; Shen, C.C. A comparative study of wireless protocols: Bluetooth, UWB, ZigBee, and Wi-Fi. *Ind. Electron. Soc.* **2007**, *5*, 46–51.
6. Rivera, D.; García, A.; Martín-Ruiz, M.L.; Alarcos, B.; Velasco, J.R.; Oliva, A.G. Secure Communications and Protected Data for a Internet of Things Smart Toy Platform. *IEEE Internet Things J.* **2019**, *6*, 3785–3795. [CrossRef]
7. Pourghebleh, B.; Wakil, K.; Navimipour, N.J. A Comprehensive Study on the Trust Management Techniques in the Internet of Things. *IEEE Internet Things J.* **2019**. [CrossRef]
8. Iskounen, S.; Nguyen, T.M.T.; Monnet, S. WiFi-direct simulation for INET in OMNeT++. *arXiv* **2016**, arXiv:1609.04604.
9. Camps-Mur, D.; Garcia-Saavedra, A.; Serrano, P. Device-to-device communications with Wi-Fi Direct: Overview and experimentation. *IEEE Wirel. Commun.* **2013**, *20*, 96–104. [CrossRef]
10. Svensson, R. Exploiting Vulnerabilities. In *From Hacking to Report Writing*; Springer: Berlin, Germany, 2016; pp. 89–152.
11. Arnaboldi, V.; Campana, M.G.G.; Delmastro, F. Context-aware configuration and management of WiFi-Direct groups for real opportunistic networks. In Proceedings of the IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), Orlando, FL, USA, 22–25 October 2017; pp. 266–274.
12. Wernke, M.; Skvortsov, P.; Dürr, F.; Rothermel, K. A classification of location privacy attacks and approaches. *Pers. Ubiquitous Comput.* **2014**, *18*, 163–175. [CrossRef]
13. Gandotra, P.; Jha, R.K.; Jain, S. A survey on device-to-device (D2D) communication: Architecture and security issues. *J. Netw. Comput. Appl.* **2017**, *78*, 9–29. [CrossRef]
14. Haus, M.; Waqas, M.; Ding, A.Y.; Li, Y.; Tarkoma, S.; Ott, J. Security and privacy in device-to-device (D2D) communication: A review. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 1054–1079. [CrossRef]

15. Bettini, C.; Riboni, D. Privacy protection in pervasive systems: State of the art and technical challenges. *Pervasive Mob. Comput.* **2015**, *17*, 159–174. [CrossRef]
16. Yeh, L.Y.; Huang, Y.L.; Joseph, A.D.; Shieh, S.W.; Tsaur, W.J. A batch-authenticated and key agreement framework for p2p-based online social networks. *IEEE Trans. Veh. Technol.* **2012**, *61*, 1907–1924. [CrossRef]
17. Yang, H.; Oleshchuk, V.A. An improvement of the batch-authentication and key agreement framework for P2P-based online social networks. In Proceedings of the International Conference on Privacy and Security in Mobile Systems (PRISMS), Aalborg, Denmark, 11–14 May 2014; pp. 1–4.
18. Shen, W.; Hong, W.; Cao, X.; Yin, B.; Shila, D.M.; Cheng, Y. Secure key establishment for device-to-device communications. In Proceedings of the IEEE Global Communications Conference, Austin, TX, USA, 8–12 December 2014; pp. 336–340.
19. Zhang, A.; Chen, J.; Hu, R.Q.; Qian, Y. SeDS: Secure data sharing strategy for D2D communication in LTE-Advanced networks. *IEEE Trans. Veh. Technol.* **2015**, *65*, 2659–2672. [CrossRef]
20. Choie, Y.J.; Jeong, E.; Lee, E. Efficient identity-based authenticated key agreement protocol from pairings. *Appl. Math. Comput.* **2005**, *162*, 179–188. [CrossRef]
21. Gehrmann, C.; Mitchell, C.J.; Nyberg, K. Manual authentication for wireless devices. *RSA Cryptobytes* **2004**, *7*, 29–37.
22. Gehrmann, C.; Nyberg, K. Security in personal area networks. *IEE Telecommun. Ser.* **2004**, *51*, 191–229.
23. Krawczyk, H.; Canetti, R.; Bellare, M. HMAC: Keyed-Hashing for Message Authentication. Available online: https://tools.ietf.org/html/rfc2104 (accessed on 30 November 2019).
24. Black, J.; Halevi, S.; Krawczyk, H.; Krovetz, T.; Rogaway, P. UMAC: Fast and secure message authentication. In *Annual International Cryptology Conference*; Springer: Berlin, Germany, 1999; pp. 216–233.
25. Falas, T.; Kashani, H. Two-dimensional bar-code decoding with camera-equipped mobile phones. In Proceedings of the Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PerComW'07), White Plains, NY, USA, 19–23 March 2007; pp. 597–600.
26. Bala, K.; Sharma, S.; Kaur, G. A study on smartphone based operating system . *Int. J. Comput. Appl.* **2015**, *121*. [CrossRef]
27. Android Developers Forum. Available online: https://developer.android.com/guide/topics/connectivity/wifip2p (accessed on 30 September 2019).
28. Android Developers Forum. Available online: https://developer.android.com/develop/index.html (accessed on 30 September 2019).
29. Mokhtarnameh, R.; Muthuvelu, N.; Ho, S.B.; Chai, I. A Comparison Study on Key Exchange-Authentication protocol. *Int. J. Comput. Appl.* **2010**, *7*, 5–11. [CrossRef]
30. Barka, E.; Kerrache, C.A.; Benkraouda, H.; Shuaib, K.; Ahmad, F.; Kurugollu, F. Towards a trusted unmanned aerial system using blockchain (BUAS) for the protection of critical infrastructure. *Trans. Emerg. Telecommun. Technol.* **2019**, e3706. [CrossRef]