Secure Device to Device Communications using Lightweight Cryptographic Protocol

Ajith Kumar V [1†] and Dr. K Satyanarayan Reddy[2††]

[1]Research Scholar, Department of Computer Applications, RRC, VTU Belguam, Karnataka, India,

[2]Professor and Head, Department of Information Science & Engineering, Cambridge Institute of Technology, Bangalore, Karnataka State, India

Abstract

The device to device (D2D) communication is an important and emerging area for future cellular networks. It is concerned about all aspect of secure data transmission between end devices along with originality of the data. In this paradigm, the major concerns are about how keys are delivered between the devices when the devices require the cryptographic keys. Another major concern is how effectively the receiver device verifies the data sent by the sender device which means that the receiver checks the originality of the data. In order to fulfill these requirements, the proposed system able to derive a cryptographic key using a single secret key and these derived keys are securely transmitted to the intended receiver with procedure called mutual authentication. Initially, derived keys are computed by applying robust procedure so that any adversary feel difficulties for cracking the keys. The experimental results shows that both sender and receiver can identify themselves and receiver device will decrypt the data only after verifying the originality of the data. Only the devices which are mutually authenticated each other can interchange the data so that entry of the intruder node at any stage is not possible.

Key words:

Confidentiality, Device to Device communication, Data integrity, Key derivation, Key robustness, Mutual authentication.

1. Introduction

The current trends of device-to-device communication to be incapable of meeting the exponential user demands. These rising demands based on the popularity of large-scale applications such as mobile computing, video streaming and large bandwidth cloud computing. Today, a trillion of wireless devices shall be proving various services for billions of people across the world. As a result, the fifth generation (5 G) device to device wireless communication play an important role in a growing technology and expected to meet the technical requirements of the next generation networks.

D2D communication directly refers the transmission of data between the intended devices without involving base station (BS). This type of data transmission improves efficiency and also system capacity. In order to transmit the data between the devices, we need to consider basic security requirements such as confidentiality, authenticity, data integrity along with privacy preservation. Confidentiality says that before sending the data to the intended user we must protect that data so that only authenticated user can capture the original data. Second important parameter of the security is a device authentication in which each device must identify themselves before sharing the data over the internet.

Data integrity is also playing a major concern in D2D communication in which receiver device can verifies the originality of the data sent by the sender device. In this context, various cryptographic protocols were implemented in a real time system, but these protocols have their own limitations with respect to the security concerns.

2. Related Work

Initially different research issues related to D2D communication such as user mobility, D2D synchronization, device discovery, interference management, resource allocation and security [1] have been discussed in detail. It is known that D2D communication is highly susceptible and vulnerable to many cryptographic, network attacks and surpasses the core network [2].

Mohammad Wazid et al. [3] [4] proposed an authentication and secure key management for fog computing. The fog computing is an extended version of cloud computing where in which inherits the security and privacy issues of cloud computing, and this scheme involves one-way cryptographic hash function and XOR operations in order to authenticate the smart devices distributed in a different geographical area. Authors have been simulated the results using widely-used NS2 simulator.

Gurjot Singh Gaba et al. [5] proposed a secure Device to- Device communications for IoT applications. In order to achieve the robustness and lightweightness the scheme

Manuscript received November 5, 2021
Manuscript revised November 20, 2021
https://doi.org/10.22937/IJCSNS.2021.21.11.48

designed as a commit/open pair. The scheme involves symmetric key cryptography, message authentication code (MAC), Diffie Hellman key exchange protocol for key generation and mutual authentication between the end devices. Security analysis involves mutual authentication, message freshness, confidentiality of the secret key in order to protect from DoS, replay attack and identical key establishment between the devices for avoiding erroneous use of secret keys.

Pimmy Gandotra et al. [6] [7] [8] [9] [10] [11] [12] conducted a survey on device-to-device communication considering security and architectural issues. Due to increase in number of devices connected radically, it is highly essential to improve the data rates with reduced latency along with system capacity. In order to acquire these, cellular networks need to undergo suitable changes regarding architectural design. In this context, the authors made a detailed survey on device-to-device communication by considering resource allocation, security and interference management to become a successful wireless network. Various attacks have been identified with solution strategies and future directions based on internet protocol security.

Yanbin Zhang et al. [13] [14] [15] proposed a mutual authentication scheme for D2D communication in smart cities. In this paper, a hybrid of medium access control (MAC) address used for establishing secure device-to-device communication sessions in IoT networks is presented to make edge-enabled smart cities safe and secure. Initiation of a right communication session between the device and base station is subjected to the authentication process so that at any stage entry of the intruder node is not possible. They are presented a security analysis by considering the parameters such as client impersonates attack, eavesdropping attack, perfect forward and backward secrecy and replay attack.

Ruhul Amin et al. [16] verification of mutual authentication and session key over an insecure communication. Mutual authentication is a one platform where user accesses several resources from the remote server at any time over internet channel. The protocol what the authors proposed can resist all kinds of security attacks. The performance of the scheme is relatively high in comparison with existing solutions and this protocol can be executable by anyone in multimedia big data environment for making a secure connection between the client and server.

Subramani Jegadeesan et al. [17] [18] proposed an efficient anonymous mutual authentication technique to ensure secure communication between mobile users and the service providers using cryptographic SSL protocol. The scheme may reduce the computation cost in great extent and the session keys are exchanged successfully with an anonymous authentication. They conducted a security analysis of forging attack, replay attack and collision attack with a mathematical proof. Maninder Singh Raniyal et al. [19] [20] [21] proposed a mutual authentication, secure key agreement for smart home and mobile device user. The proxy-based schemes (PPIDA-IC and PPIDA-PKI) for smart homes which are used to protect the private and secret keys using a passphrase or passphrase. The OTP generated using hash function is used to achieve a mutual authentication of devices. The key agreement protocol is designed to overcome the key exposure attack if any one user's device is hacked by some attackers. A secure channel is established between two end users for D2D communications in order to secure and robust to defend various attacks.

3. Design Consideration

The secure data transmission with data integrity check is a major concern of the proposed system. Both the sender and receiver device can mutually be authenticated with each other by interchanging their credentials. The notations and their meanings of the scheme as illustrated in Table 1.

Table 1: Notations and Meanings

Notations	***Meanings***
$SecKey^1$	secret key1
$SecKey^2$	secret key2
RandPart1	random character set1 from $SecKey^1$
RandPart2	random character set2 from $SecKey^1$
$DerKey^1$	derived key1
$DerKey^2$	derived key2
X	data file
Xe	Encrypted file block
H(Xe)	hash tags of Xe
Hcat(Xe)	concatenated H(Xe)
$Sigsen(DerKey^2(Hcat(Xe)))$	signature of Hcat(Xe)
$Sigrec(DerKey^2(Hrcat(Xe)))$	signature of Hrcat(Xe)
Hrec(Xe)	hash tags of Xe of receiver
Hrcat(Xe)	concatenated Hrec(Xe)
KUrec	receiver public key
KRrec	receiver private key
IDS	identity of the sender
IDR	identity of the receiver
E	encryption
N1,N2	nonce
HMAC	hash based message authentication code
SHA1	secure hash algorithm
strvar1 to strvar6	string variables

3.1 Design goals

The proposed system is implemented with the following design goals:

3.1.1 The proposed technique is designed in such a way that obtaining the secret key from the standard Blowfish algorithm to generate the two different derived keys so that these derived keys are applied for confidentiality and data integrity.

3.1.2 The system design includes mutual authentication between sender and receiver with a different message exchange.

3.1.3 In order to reduce the multiple secret keys, the system design includes random extraction of characters from the secret key and these characters are concatenated with a system generated random number that produces 160 bits hash code using SHA1 hash technique.

3.1.4 To increase the robustness of the key, 128 bits are extracted from 160 bits of hash code randomly so that it is difficult for intruder to crack the keys.

4. Design Methodology

4.1 From the sender device point of view, the following points are considered:

4.1.1 The $SecKey^1$ is computed using standard Blowfish algorithm in order to derive the future keys for confidentiality and data integrity.

4.1.2 The $SecKey^1$ consisting of string of characters are extracted randomly with a specific size.

4.1.3 Randomly extracted characters from the $SecKey^1$ is concatenated with the Random number1 (RandNum1) and concatenated results sent to the SHA1 hashing technique which produces 160 bits hash code.

4.1.4 The $DerKey^1$ is generated by extracting 128 bits randomly from the 160 bits of hash code.

4.1.5 Extract the random characters one more time from the $SecKey^1$ and concatenated with the Random number2 (RandNum2) and concatenated results sent to the SHA1 hashing technique which produces 160 bits hash code.

4.1.6 The $DerKey^2$ is generated by extracting 128 bits randomly from the 160 bits of hash code.

4.1.7 Sender encrypts the $DerKey^1$ and $DerKey^2$ using receiver public key (KU_{rec}) and send it to the receiver.

4.1.8 The entire data file is divided into number of blocks with fixed size and each block is encrypted using $DerKey^1$ and all the encrypted blocks are sent to the receiver device.

4.1.9 All encrypted blocks are concatenated together and generates the signature using $DerKey^2$ and signature of encrypted blocks sent to the receiver device.

4.2 From the receiver device point of view, the following points are considered:

4.2.1 The receiver device requests the sender to access the data with the request message which includes receiver identity (IDR) and receiver public key (KUrec).

4.2.2 Receiver accepts the encrypted key and decrypt it uses receiver private key (KRrec) and extract the $DerKey^1$ and $DerKey^2$.

4.2.3 Receiver receives the encrypted blocks, generates the signature using $DerKey^2$ and verifies the signature with the signature sent from the sender. If the signature is verified successful, then decrypt the encrypted blocks using $DerKey^1$. The high-level view of the proposed work is shown in Figure 1.

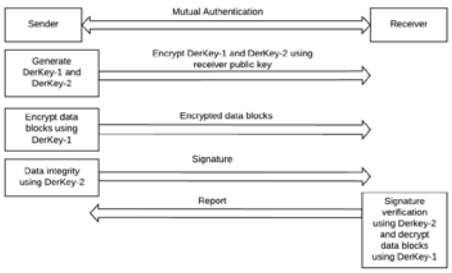

Fig. 1 System Architecture.

Algorithm 1 Generating $SecKey^1$ at the sender device

Input: Key generator algorithm (Blowfish)

Output: $SecKey^1$

1: generate key using Blowfish algorithm

2: initialize key with 128 bits

3: initialize $SecKey^1$ ← key

4: convert $SecKey^1$ into a string

5: record $SecKey^1$ for further processing

Algorithm 1 is designed in such way that SecKey1 is generated at the sender device. During the key generation, standard Blowfish algorithm of 128 bits is incurred and the secret key can be converted into any form such as

string, byte, binary and numerical value. In this paper, we are converting the secret key into a string for further processing.

Algorithm 2 Converting $SecKey^1$ into RandPart1, RandPart2 at the sender device
Input: $SecKey^1$
Output: RandPart1, RandPart2
1: receives $SecKey^1$ from **Algorithm 1**
2: **if** $SecKey^1$ is string
then
compute character length (l) of the $SecKey^1$ such that n=l($SecKey^1$)
3: extract randomly m=n/2 characters such that random(m ($SecKey^1$)) and store characters such that RandPart1 ← random(m($SecKey^1$))
4: extract randomly m=n/2 characters such that random(m ($SecKey^1$)) and store characters such that RandPart2 ← random(m($SecKey^1$))
5: initialize $Derkey^2$ ← $strvar^6$
6: record $Derkey^2$ of 128 bits at the sender device

Algorithm 2 is designed in such a way that the secret key is converted into two random parts such as RandPart1 and RandPart2 at the sender device. The algorithm takes input as a secret key and produces output as a random part. Initially, we are checking that the secret key is a string if so number of the characters in the string is computed. In the next step, we can extract the few numbers of characters from the secret key randomly and these random characters are considered as a RandPart1. The same step is repeated for extracting another few characters randomly from the secret key and these characters are called as a RandPart2.

Algorithm 3 Generate $DerKey^1$at the sender site
Input: RandPart1, $RandomNum^1$
Output: $Derkey^1$
1: receives RandPart1 from **Algorithm 2**
2: compute strvar1 ← (RandPart1 || RandNum1)
3: compute 160 bits strvar2 such that strvar2 ← SHA1(strvar1)
4: randomly extract 128 bits from strvar2 such that strvar3 ← random(strvar2)
5: initialize $Derkey^1$ ← strvar3
6: record $Derkey^1$ of 128 bits at the sender device

Algorithm 3 takes input as a RandPart1 and RandNum1 and produces $Derkey^1$ as output. The first step of the algorithm is to compute the concatenation of RandPart1 and RandNum1. The second step is to compute the 160 bits hash tag from the concatenated result using secure hash algorithm **SHA1**. The third step is very important that we are extracting 128 bits from 160 bits of hash tag using random procedure and these random characters are considered as a $DerKey^1$ which is recorded for further processing.

Algorithm 4 Generate $DerKey^2$ at sender site
Input: RandPart2, $RandomNum^2$
Output: $Derkey^2$
1: receives RandPart2 of $SecKey^1$ from Algorithm 2
2: compute strvar2 ← (RandPart2 || RandNum2)
3: compute 160 bits strvar5 such that strvar5 ← SHA1(strvar4)
4: randomly extract 128 bits from strvar5 such that strvar6 ← random(strvar5)
5: initialize $Derkey^2$ ← strvar6
6: record $Derkey^2$ of 128 bits at the sender device

Algorithm 4 is designed in such a way that computation of $Derkey^2$ at the sender device site. The first step of the algorithm is to concatenate the RandPart2 and $RandNum^2$ and retrieve 160 bits from the concatenated result. The $Derkey^2$ is defined as 128 bits of random characters extracted from the 160 bits of hash tag. The $DerKey^2$ of 128 bits is recorded for further processing.

Algorithm 5 Mutual authentication between sender and receiver
Input: ID_R, KUrec
Output: $Derkey^1$, $Derkey^2$, N_1, N_2
1: receiver send a request message to the sender that includes IDR, KUrec
2: sender responds to the receiver such that E(KUrec [IDS || $Derkey^1$ || $Derkey^2$ || IDR || N1])
3: receiver decrypt the message using KRrec record $Derkey^1$ and $Derkey^2$
4: receiver sends a message E(Derkey1(N2)) to the sender ← random(strvar5)
5: sender decrypt the message using $Derkey^1$ responds with E($Derkey^2$(f(N2)))

Algorithm 5 facilitates mutual authentication between sender and the receiver. The algorithm makes use of the request message from the sender that includes the identity of the receiver (ID_R) and receiver public key (KUrec). The sender device responds this message that consists of encrypted message. The sender device encrypts the message consisting of identity of the sender (ID_S), $DerKey^1$, $DerKey^2$, ID_R and nonce (N_1) and all these parameters are encrypted using KUrec. The receiver device decrypts the message using private key (KRrec) and extract the $DerKey^1$ and $DerKey^2$ along with nonce. The receiver uses the $DerKey^1$, encrypt the nonce (N_2) and

send this encrypted nonce to the sender device. The sender decrypts the nonce using $DerKey^1$ and computes $E(DerKey^1(f(N_2)))$ and send it to the receiver. After these steps of computations we ensure that both sender and receiver are mutual authenticated with each other before sharing the data.

Algorithm 6 confidentiality at the sender device
Input: Data file (X), $Derkey^1$
Output: Xe
1: divide data file X into blocks with fixed size such that X=b1,b2,b3……bn
2: receives $Derkey^1$ from **Algorithm 3**
3: compute file blocks into encrypted file blocks such that Xe=e1,e2,e3---en, where e1=$Derkey^1$(b1), e2= $Derkey^1$(b2) and so on
4: send encrypted file blocks (Xe) to the receiver.

Algorithm 6 facilitates data confidentiality at the sender device. The algorithm takes the entire data file (X) as input along with the $DerKey^1$ and produces the encrypted file blocks (X^e) before transmitted to the receiver device. The algorithm converts the data file into number of blocks with fixed size such that $X=b_1, b_2, b3....b_n$. Each and all fixed size blocks are converted into encrypted file blocks such that $X^e=e_1, e_2, e_3....e_n$ where $e_1=DerKey^1(b_1)$, $e_{2=}$ $Derkey^1(b_2)$and so on. These encrypted file blocks (Xe) are sent to the receiver.

Algorithm 7 data integrity at the sender device
Input: $Derkey^2$
Output: $H(X^e)$, $Sig_{sen}(DerKey^2(H_{cat}(X^e)))$
1: compute $H(X^e)=H(e_1), H(e_2), ...H(e_n)$
2: compute compute $H_{cat}(X^e)$ $H(e_1) || H(e_2) || H(e_3)...|| H(e_n)$
3: receives $Derkey^2$ from **Algorithm 4**
4: compute $Sig_{sen}(DerKey^2(H_{cat}(X^e)))$ such that $Sig_{sen}(DerKey^2(H_{cat}(X^e))) \leftarrow HMAC(DerKey^2(H_{cat}(X^e)))$
5: send $H(X^e)$, $Sig_{sen}(DerKey^2(H_{cat}(X^e)))$ to the receiver

Algorithm 7 facilitates data integrity for every data block at the sender device. To ensure data integrity we are using $DerKey^2$ for generating signature. The algorithm also takes input as encrypted data blocks (Xe) and produces the hash tags for all data blocks (H(Xe)) such that H(Xe)=H(e1), H(e2), H(e3) ...H(en). In the next step, we concatenate all the encrypted data blocks such that Hcat(Xe) H(e1) || H(e2) || H(e3) ...|| H(en). Further the signature tag is computed using HMAC hashing technique such that $Sigsen(DerKey^2(Hcat(Xe)))$ $HMAC(DerKey^2(Hcat(Xe)))$. After the successful computations of H(Xe) and $Sigsen(DerKey^2(Hcat(Xe)))$, both the hash tags of encrypted blocks along with a signature send to the receiver device.

Algorithm 8 Data integrity verification at receiver site
Input: H(Xe), $DerKey^2$,$sigsen(DerKey^2(Hcat(X^e)))$
Output: True or False
1: receives Xe from the sender device compute Hrec(Xe) = Hrec(e1), Hrec(e2), ……. Hrec(en)
2: for Hrec(Xe) from 1 to n do
if Hrec(ej) == Hsen(ej) **then**
data integrity success
end if
else
data integrity fails
end for
3: Hrcat(Xe)=Hrcat(e1) || Hrcat(e2) …. || Hrcat(en)
4: compute $Sigrec(DerKey^2(Hrcat(Xe)))$ such that Sigrec ($DerKey^2$ (Hca t(Xe))) ← HMAC ($DerKey^2$ (Hrcat (Xe)))
if $Sigrec(DerKey^2(Hrcat(Xe)))$ == $sigsen(DerKey^2(Hcat(Xe)))$
then integrity signature verified
end if
else
something went wrong
5: report Rrec to sender device

Algorithm 8 is designed at the receiver site in order to provide the data integrity verification. Before the receiver has to decrypt the entire data file, he/she allowed to check the integrity of each encrypted blocks and their signature. Once the receiver has received a hash tags of encrypted blocks H(Xe) along with the signature $Sigsen(DerKey^2(Hcat(Xe)))$, data integrity verification is carried out. The receiver receives the encrypted data blocks (Xe) from the sender device and hash tags of these encrypted blocks such that Hrec(Xe)= Hrec(e1), Hrec(e2), Hrec(e3)...Hrec(en) then the receiver checks the originality of all the encrypted blocks.

The receiver also verifies the signature of the sender by generating his/her signature such that $Sigrec(DerKey^2(Hrcat(Xe)))$ $HMAC(DerKey^2(Hrcat(Xe)))$ and compares this signature with the signature of the sender such that $sigrec(DerKey^2(Hrcat(Xe)))==Sigsen(DerKey^2(Hcat(Xe)))$. If both the signatures are verified successfully then data decryption will be carried out otherwise something went wrong, and the report is sent to the sender.

Algorithm 9 Decryption of data blocks at the receiver device

Input: X^e, $Derkey^1$
Output: Data file(X)
1: **if** step 2 and 5 of the **Algorithm 8** is successful
then
decrypt the data blocks such that $Derkey^1(X^e)$ = $b_1, b_2, b_3, \ldots\ldots b_n$ and so on
end if
2: perform operations on X

Algorithm 9 is designed at the receiver device for decrypting the data blocks which are shared by the sender device. The algorithm takes input as encrypted data file (X^e), $DerKey^1$ and produces the output as a corresponding data file (X). When the step 2 and step 5 of the Algorithm 8 is successful, the receiver receives the encrypted data blocks such as e_1, e_2.... e_n and decrypts them such that $b_1=DerKey^1(e_1)$, $b_2=DerKey^1(e_2)$ and so on.

4.3. Implementation

The proposed scheme is implemented using a Java-based JSP web application. The scheme is being tested in a physical cloud environment such as Amazon Web Services. Device setup includes Tomcat 8.5 with Corretto 11 running on 64bit Amazon Linux 2/4.1.2.

5. Experimental Results

In this section, the experimental results are described in detailed manner considering computational cost as a major factor. The number of results samples are collected from the internet as amazon web services and these results are depicted in a tabular format. The Table 2 shows the computational cost estimated from the proposed scheme and same as compared with an existing Blowfish algorithm. From the tabulated experimental results, the proposed method may consume very low computation cost while deriving the keys.

In order to analyze the experimental results, we specifically considered the computational values of $DerKey^1$ (Proposed method) and these values are compared with the $SecKey^1$ (Blowfish algorithm) is shown in Figure 2. From this graphical representation, we practically conclude that time taken for generating $DerKey^1$ consumes low computation cost in comparison with the $SecKey^1$.

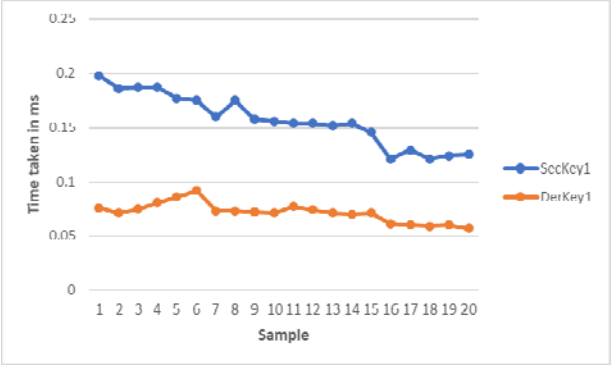

Fig. 2 Time taken (ms) SecKey1 vs DerKey1 (Proposed method).

In order to quantify the experimental analysis, the proposed scheme may also be analyzed with another existing key derivation. The computational cost for $DerKey^2$ (Proposed method) is compared with a $SecKey^2$ (Blowfish algorithm) with a graphical representation shown in Figure 3. From this graphical representation, we also conclude that time taken for generating $DerKey^2$ consumes low computation cost in comparison with the $SecKey^2$.

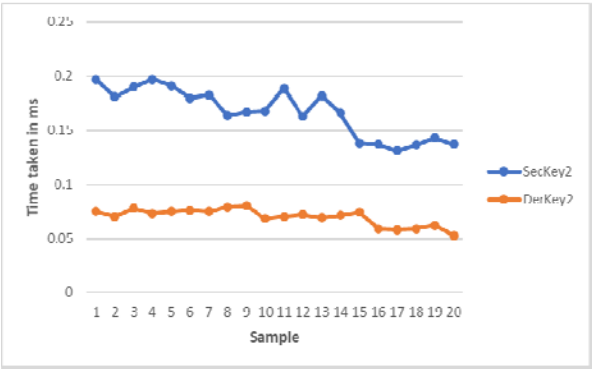

Fig. 3 Time taken (ms) $SecKey^2$ vs $DerKey^2$ (Proposed method).

In order analyze the storage cost introduced by the proposed scheme and it is compared with a Blowfish algorithm while deriving the session keys. The Table 3 shows the storage cost in terms of number of bytes consumed during key generation. These recorded values shows that the number of bytes consumed by the proposed scheme while deriving $DerKey^1$ and $DerKey^2$ are little bit low in comparison with the current solution shown in Figure 4 and Figure 5.

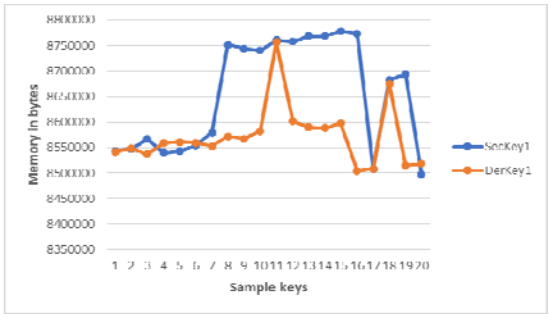

Fig. 4 Memory in bytes SecKey[1] vs DerKey[1] (Proposed method).

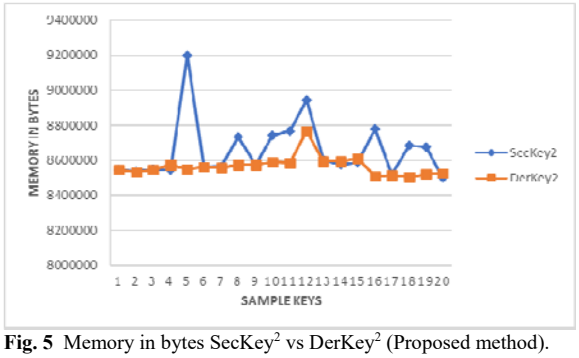

Fig. 5 Memory in bytes SecKey[2] vs DerKey[2] (Proposed method).

Table 2: Time Taken (ms) For Key Derivation (Blowfish Vs. Proposed Methodology)

Sample	SecKey[1]	SecKey[2]	DerKey[1]	DerKey[2]
1	0.198	0.197	0.076	0.075
2	0.186	0.181	0.071	0.07
3	0.187	0.19	0.075	0.078
4	0.187	0.197	0.081	0.073
5	0.177	0.191	0.086	0.075
6	0.175	0.18	0.092	0.076
7	0.16	0.183	0.073	0.075
8	0.175	0.164	0.073	0.079
9	0.158	0.167	0.072	0.08
10	0.156	0.168	0.071	0.068
11	0.154	0.189	0.077	0.07
12	0.154	0.163	0.074	0.072
13	0.152	0.182	0.071	0.069
14	0.154	0.166	0.07	0.071
15	0.146	0.138	0.071	0.074
16	0.121	0.137	0.061	0.059
17	0.129	0.131	0.06	0.058
18	0.121	0.136	0.059	0.059
19	0.124	0.143	0.06	0.062
20	0.125	0.137	0.057	0.053
Average	**0.156**	**0.167**	**0.071**	**0.069**

In order analyze the storage cost introduced by the proposed scheme and it is compared with a Blowfish algorithm while deriving the session keys. The Table 3 shows the storage cost in terms of number of bytes consumed during key generation. These recorded values shows that the number of bytes consumed by the proposed scheme while deriving DerKey[1] and DerKey[2] are little bit low in comparison with the current solution shown in Figure 4 and Figure 5.

Table 3: Memory Utilization in Bytes (Blowfish Vs. Proposed Methodology)

Sample	SecKey[1]	SecKey[2]	DerKey[1]	DerKey[2]
1	8543552	8544432	8541968	8545360
2	8547624	8538504	8548296	8534248
3	8566776	8548504	8537112	8544936
4	8540008	8547608	8559096	8573568
5	8542752	9197088	8561032	8548616
6	8554040	8562128	8559568	8562200
7	8579496	8564776	8552960	8557176
8	8752088	8734632	8571248	8574808
9	8743800	8573088	8567640	8573720
10	8740184	8744736	8582160	8590560
11	8760576	8766856	8756712	8587744
12	8757552	8943808	8602016	8765568
13	8768424	8602872	8589952	8595496
14	8768520	8574152	8588488	8595488
15	8778320	8588112	8597744	8609552
16	8773616	8777992	8504048	8507752
17	8507792	8521008	8508496	8514936
18	8683456	8686848	8675208	8505600
19	8693344	8676928	8514448	8522728
20	8497016	8501584	8518184	8526576
Average	**8654946.8**	**8659783**	**8571818.8**	**8566831.6**

6. Security analysis of the proposed scheme

Secure deceive to device (D2D) communication using lightweight cryptographic protocol is accomplished with four different phases i.e., key derivation at sender site, secure key delivery through mutual authentication, increasing key robustness and checking integrity of each data blocks. Moreover, a detailed analysis of various possible attacks and their prevention in the proposed method is described in realistic environment of D2D communication is presented below.

6.1 Scalability of key derivation

In the proposed scheme, the derived keys are extracted from the secret key that reduces the computational cost for another secret keys. It means that from one particular secret key, we can be able to computes number of derived keys.

6.2. Mutual authentication

Before data has been exchanged between the sender and receiver device, both of them mutually authenticates themselves in order to prevent entry of adversary node. The proposed method has securely established the connection between sender and receiver by exchanging the messages between them. In this paper Algorithm 5 is designed for secure mutual authentication between the sender and receiver device.

6.3. Secret key establishment

The secure key delivery or establishment is one of the important concerns in D2D communication. Even though the keys are transmitted in a secure channel, there is a possibility to capture the contents of the session keys. In the proposed method, the derived keys are encrypted using public key of the receiver and the encrypted keys are sent to the receiver. These encrypted keys are only decrypted by private key of the receiver and this private key is only know to the receiver device.

6.4. Message integrity

Message integrity says that there is no modification, deletion, addition or replay of the transmitted data. The proposed method is designed in such a way that each and every block undergone the originality check. However, signature has been constructed for entire data file and receiver is allowed for verifying the signature before decrypting each data block.

6.5. Eavesdropping attack

Eavesdropping attacks are applicable only the intruder device has the capability to intercept each, and every packet transmitted between the D2D communication via channels or paths which are not secure. In the proposed scheme, before the data transmission, derived keys are securely transmitted through mutual authentication and every device is bounded to transmit data in encrypted form and, thus, eavesdropping attacks are not feasible.

6.6. Impersonates attacks

In the proposed scheme, the intruder device will not derive the data, which is shared between the sender and receiver device, since the derived keys are encrypted using receiver public key. Even the intruder captures the encrypted key while sharing that cannot be decrypted because the decryption is only possible by using receiver private key. G. Perfect forward and backward secrecy. If the intruder device has a capacity to intercept a complete communication session between the sender and receiver device and even the intruder device somehow finds a way to encrypt the particular data using own key and sends the encrypted data along with a key to the receiver device, then the receiver.

7. Conclusion

Establishing keys and providing mutual authentication between two end users are dominate issues for a device to device (D2D) communication, which impacts the success of services offered in a next generation of D2D communications. In this paper, we proposed a secure device to device communications using lightweight cryptographic protocol. Initially, the scheme generates the secret key of 128 bits using standard Blowfish algorithm. The derived keys are generated using secret key using random numbers and SHA1 hashing technique which produces keys of 160 bits. In order to reduce the computational cost data confidentiality and data integrity, we reduce the key size from 160 bits into 128 bits by extracting characters using mathematical random procedure. The mutual authentication is achieved in a great extent between the sender and receiver device. The security analysis and key robustness proved that our lightweight cryptographic protocol is secure in order to defend various attacks. The experimental results shows that our scheme can be deploy-able in real time D2D communication.

References

[1] U. N. Kar and D. K. Sanyal, “A critical review of 3gpp standardization of device-to-device communication in cellular networks,” SN Computer Science, vol. 1, no. 1,p. 37, 2020.

[2] U. N. Kar and D. K. Sanyal, "An overview of device-to-device communication in cellular networks", ICT Express, volume 4, issue 4, December 2018, pages 203-208.

[3] M. Wazid, A. K. Das, N. Kumar, and A. V. Vasilakos, “Design of secure key management and user authentication scheme for fog computing services,” Future Generation Computer Systems, vol. 91, pp. 475–492,2019.

[4] A. P. G Lopes and P. R. Gondim, “Mutual authentication protocol for d2d communications in a cloud-based e-health system,” Sensors, vol. 20, no. 7, p. 2072, 2020.

[5] G. S. Gaba, G. Kumar, T.-H. Kim, H. Monga, and P. Kumar, “Secure device-to-device communications for 5g enabled internet of things applications,” Computer Communications, vol. 169, pp. 114–128, 2021.

[6] P. Gandotra, R. K. Jha, and S. Jain, “A survey on device-to-device (d2d) communication: Architecture and security issues,” Journal of Network and Computer Applications, vol. 78, pp. 9–29, 2017.

[7] M. Haus, M. Waqas, A. Y. Ding, Y. Li, S. Tarkoma, and J. Ott, “Security and privacy in device-to-device (d2d) communication: A review,” IEEE Communications Surveys & Tutorials, vol. 19, no. 2, pp. 1054–1079, 2017.

[8] F. Jameel, Z. Hamid, F. Jabeen, S. Zeadally, and M. A. Javed, “A survey of device-to-device communications: Research issues and challenges,” IEEE Communications Surveys & Tutorials, vol. 20, no. 3, pp. 2133– 2168, 2018.

[9] J. Wang, Y. Huang, S. Jin, R. Schober, X. You, and C. Zhao, “Resource management for device-to-device communication: A physical layer security perspective,” IEEE Journal on Selected Areas in Communications, vol. 36, no. 4, pp. 946–960, 2018.

[10] A. Khan, Y. Javed, J. Abdullah, J. Nazim, and N. Khan, "Security issues in 5g device to device communication," IJCSNS, vol. 17, no. 5, p. 366, 2017.

[11] S. T. Shah, S. F. Hasan, B.-C. Seet, P. H. J. Chong, and M. Y. Chung, "Device-to-device communications: A contemporary survey," Wireless Personal Communications, vol. 98, no. 1, pp. 1247–1284, 2018.

[12] O. N. Hamoud, T. Kenaza, and Y. Challal, "Security in device-to-device communications: a survey," IET Networks, vol. 7, no. 1, pp. 14–22, 2017.

[13] Y. Zhang, K. Cheng, F. Khan, R. Alturki, R. Khan, and A. U. Rehman, "A mutual authentication scheme for establishing secure device-to-device communication sessions in the edge-enabled smart cities," Journal of Information Security and Applications, vol. 58, p. 102683, 2021.

[14] J. Kim and J. Song, "A secure device-to-device link establishment scheme for lorawan," IEEE Sensors Journal, vol. 18, no. 5, pp. 2153– 2160, 2018.

[15] M. Waqas, M. Ahmed, Y. Li, D. Jin, and S. Chen, "Social-aware secret key generation for secure device-to-device communication via trusted and non-trusted relays," IEEE Transactions on Wireless Communications, vol. 17, no. 6, pp. 3918–3930, 2018.

[16] R. Amin, S. H. Islam, P. Vijayakumar, M. K. Khan, and V. Chang, "A robust and efficient bilinear pairing based mutual authentication and session key verification over insecure communication," Multimedia Tools and Applications, vol. 77, no. 9, pp. 11 041–11 066, 2018.

[17] S. Jegadeesan, M. Azees, P. M. Kumar, G. Manogaran, N. Chilamkurti, R. Varatharajan, and C.-H. Hsu, "An efficient anonymous mutual authentication technique for providing secure communication in mobile cloud computing for smart city applications," Sustainable Cities and Society, vol. 49, p. 101522, 2019.

[18] V. Kumar, M. Ahmad, and A. Kumari, "A secure elliptic curve cryptography based mutual authentication protocol for cloud-assisted tmis," Telematics and Informatics, vol. 38, pp. 100–117, 2019.

[19] M. S. Raniyal, I. Woungang, S. K. Dhurandher, and S. S. Ahmed, "Passphrase protected device-to-device mutual authentication schemes for smart homes," Security and Privacy, vol. 1, no. 3, p. e42, 2018.

[20] L. Wu, J. Wang, K.-K. R. Choo, and D. He, "Secure key agreement and key protection for mobile device user authentication," IEEE Transactions on Information Forensics and Security, vol. 14, no. 2, pp. 319–330, 2018.

[21] M. Wang, Z. Yan, and V. Niemi, "Uaka-d2d: Universal authentication and key agreement protocol in d2d communications," Mobile Networks and Applications, vol. 22, no. 3, pp. 510–525, 2017.

Ajith Kumar V currently a Ph.D. student at Visvesvaraya Technological University, Regional Resource Center of Belagavi. His research focus is on security for resource-constrained devices and application of lightweight cryptographic techniques for securing D2D communication. He obtained B.Sc. degree from University of Mysore in 1991 and his M.C.A., degree from Kuvempu University in 1999. His interest includes Computer Forensics, Cyber Security. He is life member of Cryptology Research Society of India (CRSI).

K. Satyanarayan Reddy currently working as Professor and Head of Information Sceince & Engineering, Cambridge Institute of Technology, Bangalore. His qualification includes Ph.D. in Computer Science (Dravidian University, Kuppam, AP), MTech in Computer Applications (Dept. Of CSE, ISM Dhanbad). He has worked as faculty in many Engineering Colleges. He has more than 25 Research Papers (National and International) in his credit and has chaired national and international conferences. Delivered Keynote address in few national level conferences.