International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9 Issue-6, April 2020

Preventing Cryptographic Attacks in Iot using Lightweight P-Box Permutation Encryption

Aruna Gawade, Narendra Shekokar

Abstract*— Internet of Things (IoT) is a leading technology, where numerous sensors with the ability to exchange information over the internet are involved. It supports various applications ranging from smart home to smart city including smart grid for calculating electricity tariff, real time monitoring and controlling traffic, Co2 emission level finding etc. Research advancements in IoT facilitates us to control or monitor the things remotely and take actions accordingly. Efficient working of the IoT, involves continuous exchange of information between the nodes which should be executed in trusted environment as this data is confidential and only trusted nodes should handle it. Current research work in IoT suggests the confidentiality mechanisms which are heavyweight and not suitable in IoT scenario. So there is a need for lightweight confidentiality mechanism. In this paper we have shown how lightweight P-Box permutation encryption algorithm can be effective in IoT. We have compared this encryption algorithm with PRESENT protocol and our results show that lightweight P-Box permutation algorithm took less time. To prove that our encryption algorithm is better w.r.t. security, we have also carried out known plaintext attack and chosen plaintext attack and shown that lightweight P-Box permutation encryption algorithm is resistant to both of these attacks. Further we have used this technique to transfer the smart meter data in Smart Grid and shown that the data exchanged between the two nodes in confidential manner.

***Index Terms*— Internet of Things, smart grid, PRESENT, Lightweight P-Box Permutation Encryption, attacks**

I. INTRODUCTION

IoT is key enabler for many smart applications ranging from smart home, smart building, smart city, smart healthcare, smart grid etc. Sensors and actuators along with nodes play vital role in all these applications. These nodes exchange data over the cloud periodically and allows user to access it over the internet using mobile application or website.

Smart Grid as an IoT application becomes popular in smart energy. Smart Grid offers various advantages such as:

- Reestablishment of electricity in case of power disturbances
- Reduced power costs and peak demand, in turn reducing electricity rates

Rapid deployment of Smart Grid has resulted in development of smart meter as a replacement to existing electromechanical meters which are read manually once a month. Smart meters/AMI can provide the same data every few minutes, in real time. Smart appliances are connected to Smart meters which in turn calculates and records our hourly, daily consumption of electricity, in regular intervals of time and communicates that information back to the utility server to calculate the tariff. It enables two-way communication between the meter and the utility server and gather data for remote reporting (on-demand and periodic), which is used by the utility server for billing and consumption monitoring purposes.Smart Grid along with Meter Data Management System(MDMS) plays a vital role in Smart meter communication network. This network carries the usage information from smart meter to MDMS and in turn to remote switch control.

One important concern here is the continuous transmitting nature of the smart meter i.e. 24/7 making it accessible to the attacker and revealing the exact details of consumer's daily activities. This information is very crucial as it reveals our daily personal behavior as when we leave home, when we enter the home etc. and in turn should be kept confidential. If this information is hacked and misused by the attacker then he/she will get full control of the home and can have devastating consequences. Few consequences are listed below:

- Modifications of smart meter reading
- Disconnecting smart meter
- Compromising supplier and DCC
- Sending inaccurate information to the utility to generate the wrong bills
- Adjusting bills
- Accessing payment information
- Cutting power to a home
- Exploding meters or house fires privilege

To take care of these concerns Smart Meter Communication Network should ensure data security and handle reliable data. Different ways for handling security are given in literature such as authenticating the nodes, maintaining confidentiality by encrypting the data being exchanged etc. In this work we have concentrated mainly on the confidentiality and encryption mechanism.

Revised Manuscript Received on April 30, 2020.
* Correspondence Author
Aruna Gawade*, Asst. Prof., Computer Engg. Dept D.J.sanghvi COE aruna.gawade@djsce.ac.in
Narendra Shekokar, Prof. Computer Engg. Dept, D.J.sanghvi COE narendra.shekokar@djsce.ac.in

In our earlier work [9] we have compared different lightweight encryption algorithms such as PRESENT, Clefia, Chacha20, TEA and AES and came to the conclusion that PRESENT being ultra lightweight cipher, is most suited in IoT environment. We have proposed a new lightweight P-box permutation encryption algorithm. We have tested our algorithm for different inputs of plaintext (stream of around 300inputs) and proved that light P-box permutation algorithm is better w.r.t. time than lightweight PRESENT protocol.

Moving ahead, in this paper we have actually tested this algorithm in sensor based environment, i.e. actual node to node communication over the cloud environment and proved that our algorithm not only takes less time but also resistant to attacks.

The rest of the paper is organized as follows: section II describes different types of attack scenarios in Smart Grid Environment given in literature, section III describes system architecture used along with details of NodeMCU and BLYNK app and algorithms, in section IV we have shown encryption and decryption of the smart meter parameters in the IoT environment and in section 5 we have explained different types of attacks possible w.r.t. cryptography and finally in section 6 we have shown that Lightweight P-Box Permutation encryption algorithm is resistant to Chosen plaintext attack(CPA) and Known plaintext attack(CPA) .

II. LITERATURE SURVEY

In this section we have concentrated on research work done on possible smart grid attacks where attack vector ranges from targeting smart meter to bringing the entire grid down.

Satin Asri et al. [1] suggests security concerns like data theft & manipulation caused because of access by unauthorized parties, leaking private and valuable information, under-report energy use. Information and communication infrastructure which is prone to different attacks are also discussed. A DoS attack and vulnerability attacks are discussed and shown UDP flood attack.

Zubair Baig et al. [2] Elaborates different types of attacks like smart meter, physical memory, Network layer etc. considering confidentiality, Integrity, Availability, Non-Repudiation, Eavesdropping, Jamming, Injecting Requests/ Restricting Access, Injection Attacks

Ping Yi et al. [3] discusses a DoS attack called Puppet attack, where attack is applied in mesh network of smart meter. As mesh network is having limited communication bandwidth and route packets are top priority in all packets. Excess route packets will exhaust limited communication bandwidth and result in network congestion, causing a denial of service attack in AMI network.

David G. et al. [4] surveys AMI threats and attacks possible on them. A threat analysis is done considering the objectives of attack and the steps taken for attacks for eg. energy theft by eavesdropping. This analysis gives the attack detection information. Authors also proposed the need for hybrid sensing infrastructure to monitor the attacks.

Dong Chen et al. [5] Authors used a statistical approach to deal with private information leakage. Based on user pattern of electricity usage, private information i.e. regular working hours, vacation timing etc. can be correlated and thus occupancy of the premise can be detected. This is very crucial information and many companies are mining this data to detect power signatures and thus detects occupancy. A method to modulate a water heater's power usage to create illusion that always some or other person occupies home is used.

Yongheguo et al.[6] discusses various types of attacks like 1) Attacks on protocol such as TCP SYN flooding attack on data collecting unit. 2) Attacks on infrastructure: Disrupting the routing tables to deteriorate the performance of packet delivery.
3) Attacks on bandwidth: Excessive communication packets sent to the victim force it to drop the legitimate packets.

From the Literature survey it is obvious that the data being exchanged in IoT environment can be hacked and misused. To keep it confidential, application should use proper encryption mechanism so that attackers will not be able to access it.

III. SYSTEM ARCHITECTURE

As IoT is a collection of sensors with ability to exchange the sensed data to cloud platform, we set up the same environment for testing purpose. As shown in fig. 1, site1 with NodeMCU (an open source IoT platform) is exchanging data with site2. At site 1 we have used a dataset[10] about the information which is getting stored in smart meter.i.e. current time and date, Gas and electricity usage and spend, Meter readings, Smart meter's serial number etc. Meter reading is transferred by NodeMCU in an encrypted format, through cloud server environment to site2. At site2 NodeMCU arduino interface is configured to read and decrypt this data and provide it as the output to the BLYNK mobile app.

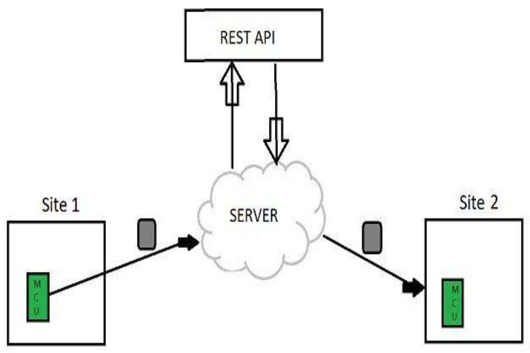


Fig 1. IoT environment Server MCU communication

For Encryption and Decryption a PUF based secret key generation technique is used. This key is then feed to PRESENT encryption algorithm and Lightweight P-Box Permutation encryption algorithm. The setup includes NodeMCU as the IoT platform and BLYNK as the mobile application. The hardware components description is as follows:

1. **NodeMCU** (Node MicroController Unit)**:** Node MCU is same as arduino with built-in wifi module.

It's a software and hardware development environment with System-on-a-Chip (SoC) called the ESP8266. It is used in IoT for developing applications using Arduino development environment. It also has analog pin, digital pins and serial communication protocols.

2. **BLYNK:** It is a mobile application which uses Blynk libraries and facilitates us to remotely control the hardware, display data and store data coming from thousands of devices. BlynkCloud or server handles the communication between smartphone and hardware.

The steps are as follows:

1. Two nodes exchanged a 32 char shared key between them using PUF based authentication mechanism[7]
2. NodeMCU retrieve the readings periodically and they are converted into 8 bit plain text
3. As shown in fig.2, NodeMCU encrypts these readings using
   a. PRESENT encryption algorithm[8]
   b. Lightweight P-Box Permutation encryption algorithm[9]
4. This encrypted data is sent to the cloud and timings are recorded.
5. Attacks are carried out for recovering key.

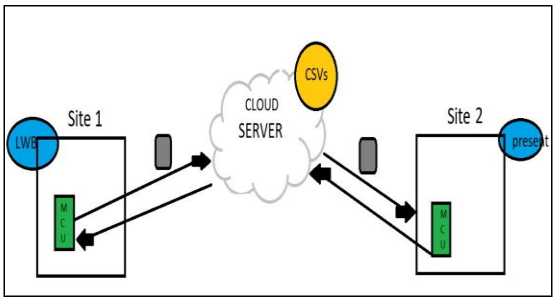


Fig 2. IoT environment with protocols

IV. WORKING MODEL

This section describes the working of nodes on sending and receiving side. We have used the plain text readings from the sensor as 8 bit data and 32 char key to encrypt this data. Both sender and reciver node assumed to have shared key which can be generated using PUF based authentication mechanism. Once both parties calculate the shared key they are ready for exchanging the data in encrypted format.

Here we have considered two scenarios:

1. Encryption of sensed data using PRESENT protocol.
2. Encryption of sensed data using lightweight P-Box protocol

1.**Encryption of sensed data using PRESENT protocol**

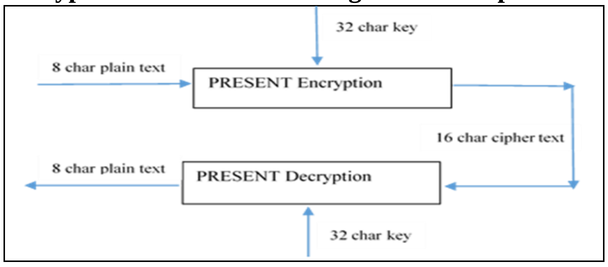
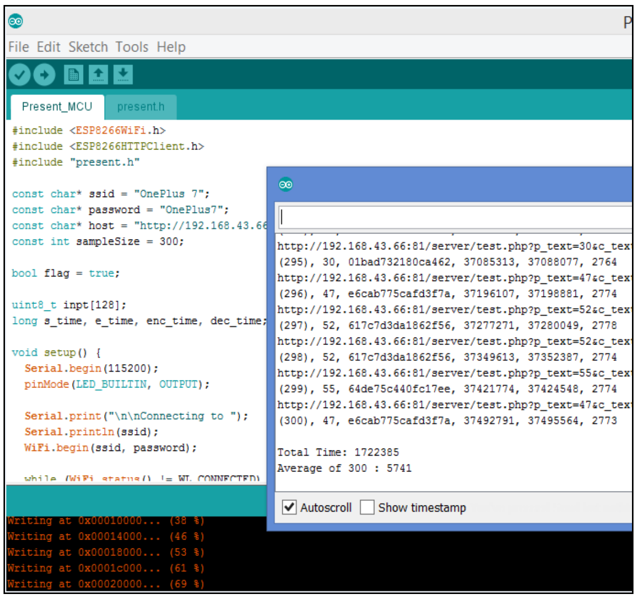


Fig 3. PRESENT protocol encryption

Fig. 3 shows the block diagram and Fig. 4 shows the results of PRSENT protocol implementation when run on Arduino platform.



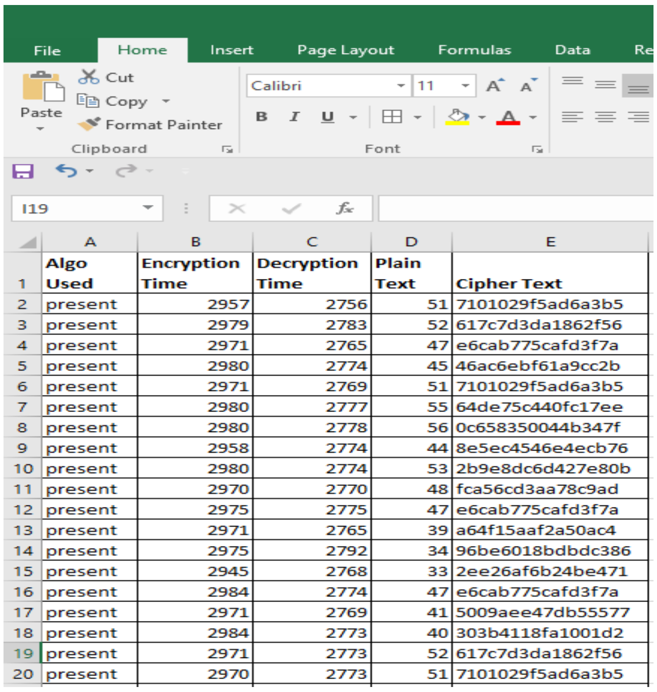
```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include "present.h"

const char* ssid = "OnePlus 7";
const char* password = "OnePlus7";
const char* host = "http://192.168.43.66
const int sampleSize = 300;

bool flag = true;

uint8_t inpt[128];
long s_time, e_time, enc_time, dec_time;

void setup() {
  Serial.begin(115200);
  pinMode(LED_BUILTIN, OUTPUT);

  Serial.print("\n\nConnecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
```

```
http://192.168.43.66:81/server/test.php?p_text=30&c_tex
(295), 30, 01bad732180ca462, 37085313, 37088077, 2764
http://192.168.43.66:81/server/test.php?p_text=47&c_tex
(296), 47, e6cab775cafd3f7a, 37196107, 37198881, 2774
http://192.168.43.66:81/server/test.php?p_text=52&c_tex
(297), 52, 617c7d3da1862f56, 37277271, 37280049, 2778
http://192.168.43.66:81/server/test.php?p_text=52&c_tex
(298), 52, 617c7d3da1862f56, 37349613, 37352387, 2774
http://192.168.43.66:81/server/test.php?p_text=55&c_tex
(299), 55, 64de75c440fc17ee, 37421774, 37424548, 2774
http://192.168.43.66:81/server/test.php?p_text=47&c_tex
(300), 47, e6cab775cafd3f7a, 37492791, 37495564, 2773

Total Time: 1722385
Average of 300 : 5741
```

```
Writing at 0x00010000... (38 %)
Writing at 0x00014000... (46 %)
Writing at 0x00018000... (53 %)
Writing at 0x0001c000... (61 %)
Writing at 0x00020000... (69 %)
```

Fig 4. PRESENT protocol on Arduino

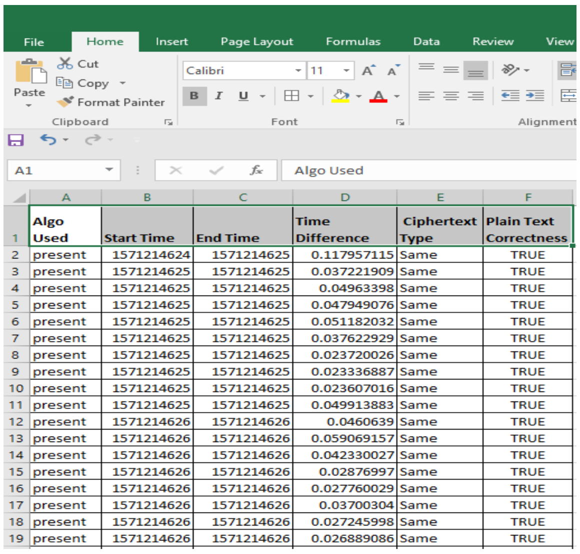
Fig. 5 shows the results of PRSENT protocol implementation on cloud with details of node time recordings for encryption and decryption along with plaintext and cipher text generated


File
Home
Insert
Page Layout
Formulas
Data
Cut
Copy
Format Painter
Paste
Clipboard
Calibri
11
Font
I19

	A	B	C	D	E
1	Algo Used	Encryption Time	Decryption Time	Plain Text	Cipher Text
2	present	2957	2756	51	7101029f5ad6a3b5
3	present	2979	2783	52	617c7d3da1862f56
4	present	2971	2765	47	e6cab775cafd3f7a
5	present	2980	2774	45	46ac6ebf61a9cc2b
6	present	2971	2769	51	7101029f5ad6a3b5
7	present	2980	2777	55	64de75c440fc17ee
8	present	2980	2778	56	0c658350044b347f
9	present	2958	2774	44	8e5ec4546e4ecb76
10	present	2980	2774	53	2b9e8dc6d427e80b
11	present	2970	2770	48	fca56cd3aa78c9ad
12	present	2975	2775	47	e6cab775cafd3f7a
13	present	2971	2765	39	a64f15aaf2a50ac4
14	present	2975	2792	34	96be6018bdbdc386
15	present	2945	2768	33	2ee26af6b24be471
16	present	2984	2774	47	e6cab775cafd3f7a
17	present	2971	2769	41	5009aee47db55577
18	present	2984	2773	40	303b4118fa1001d2
19	present	2971	2773	52	617c7d3da1862f56
20	present	2970	2773	51	7101029f5ad6a3b5

Fig 5. PRESENT protocol node time recording

Fig. 6 shows the results of PRSENT protocol implementation on cloud with details of cloud time recordings for encryption and decryption along with plaintext and cipher text correctness.



	Algo Used	Start Time	End Time	Time Difference	Ciphertext Type	Plain Text Correctness
2	present	1571214624	1571214625	0.117957115	Same	TRUE
3	present	1571214625	1571214625	0.037221909	Same	TRUE
4	present	1571214625	1571214625	0.04963398	Same	TRUE
5	present	1571214625	1571214625	0.047949076	Same	TRUE
6	present	1571214625	1571214625	0.051182032	Same	TRUE
7	present	1571214625	1571214625	0.037622929	Same	TRUE
8	present	1571214625	1571214625	0.023720026	Same	TRUE
9	present	1571214625	1571214625	0.023336887	Same	TRUE
10	present	1571214625	1571214625	0.023607016	Same	TRUE
11	present	1571214625	1571214625	0.049913883	Same	TRUE
12	present	1571214626	1571214626	0.0460639	Same	TRUE
13	present	1571214626	1571214626	0.059069157	Same	TRUE
14	present	1571214626	1571214626	0.042330027	Same	TRUE
15	present	1571214626	1571214626	0.02876997	Same	TRUE
16	present	1571214626	1571214626	0.027760029	Same	TRUE
17	present	1571214626	1571214626	0.03700304	Same	TRUE
18	present	1571214626	1571214626	0.027245998	Same	TRUE
19	present	1571214626	1571214626	0.026889086	Same	TRUE

Fig 6. PRESENT protocol cloud time recording

1. Encryption of sensed data using lightweight P-Box protocol

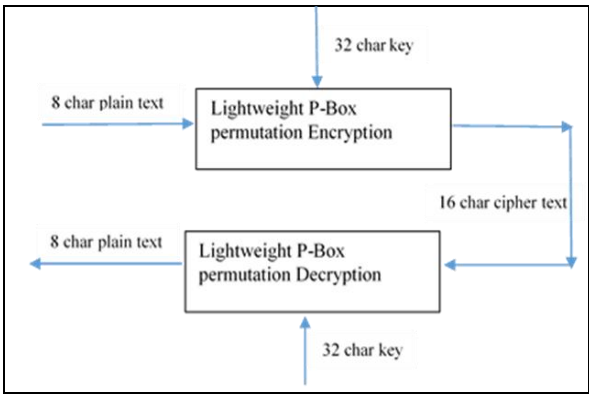
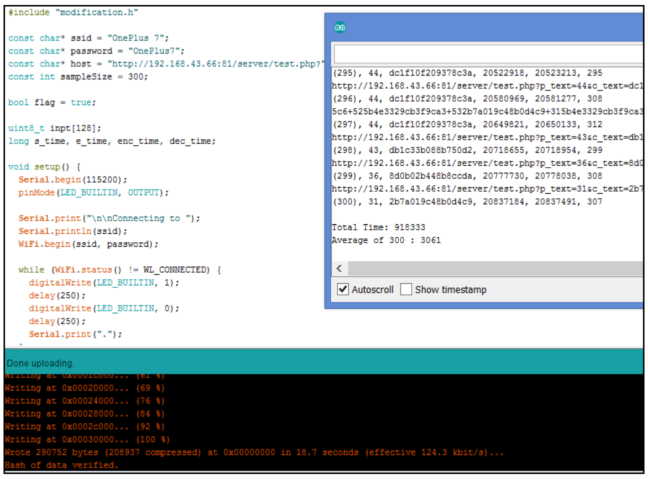

32 char key
8 char plain text
Lightweight P-Box permutation Encryption
16 char cipher text
8 char plain text
Lightweight P-Box permutation Decryption
32 char key

Fig 7. Lightweight P-Box protocol encryption

Fig. 7 shows the block diagram and Fig. 8 shows the results of lightweight P-Box protocol implementation when run on Arduino platform.



```
#include "modification.h"

const char* ssid = "OnePlus 7";
const char* password = "OnePlus7";
const char* host = "http://192.168.43.66:81/server/test.php?"
const int sampleSize = 300;

bool flag = true;

uint8_t inpt[128];
long s_time, e_time, enc_time, dec_time;

void setup() {
  Serial.begin(115200);
  pinMode(LED_BUILTIN, OUTPUT);

  Serial.print("\n\nConnecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    digitalWrite(LED_BUILTIN, 1);
    delay(250);
    digitalWrite(LED_BUILTIN, 0);
    delay(250);
    Serial.print(".");
```

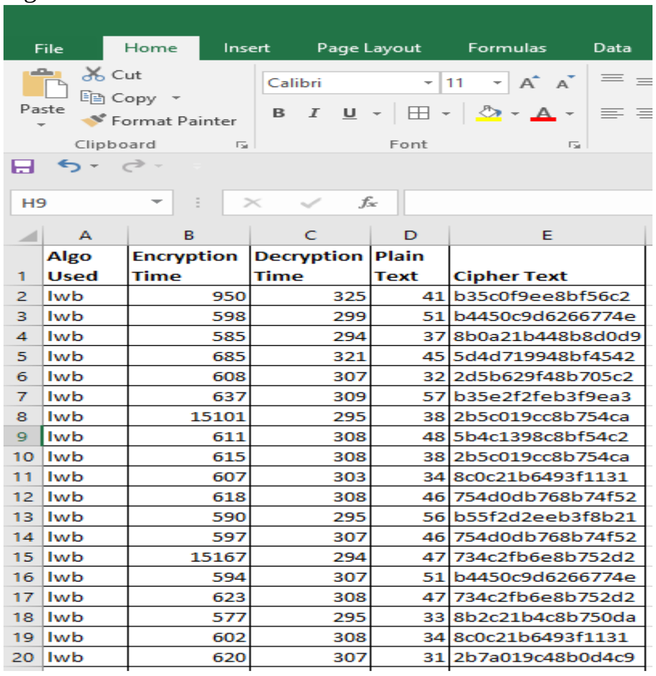
```
(295), 44, dc1f10f209378c3a, 20522918, 20523213, 295
http://192.168.43.66:81/server/test.php?p_text=44&c_text=dc1
(296), 44, dc1f10f209378c3a, 20580969, 20581277, 308
5c6+525b4e3329cb3f9ca3+532b7a019c48b0d4c9+315b4e3329cb3f9ca3
(297), 44, dc1f10f209378c3a, 20649821, 20650133, 312
http://192.168.43.66:81/server/test.php?p_text=43&c_text=db1
(298), 43, db1c33b088b750d2, 20718655, 20718954, 299
http://192.168.43.66:81/server/test.php?p_text=36&c_text=8d0
(299), 36, 8d0b02b448b8ccda, 20777730, 20778038, 308
http://192.168.43.66:81/server/test.php?p_text=31&c_text=2b7
(300), 31, 2b7a019c48b0d4c9, 20837184, 20837491, 307

Total Time: 918333
Average of 300 : 3061
```

[✓ Autoscroll] Show timestamp

```
Done uploading.
Writing at 0x0001c000... (61 %)
Writing at 0x00020000... (69 %)
Writing at 0x00024000... (76 %)
Writing at 0x00028000... (84 %)
Writing at 0x0002c000... (92 %)
Writing at 0x00030000... (100 %)
Wrote 290752 bytes (208937 compressed) at 0x00000000 in 18.7 seconds (effective 124.3 kbit/s)...
Hash of data verified.
```

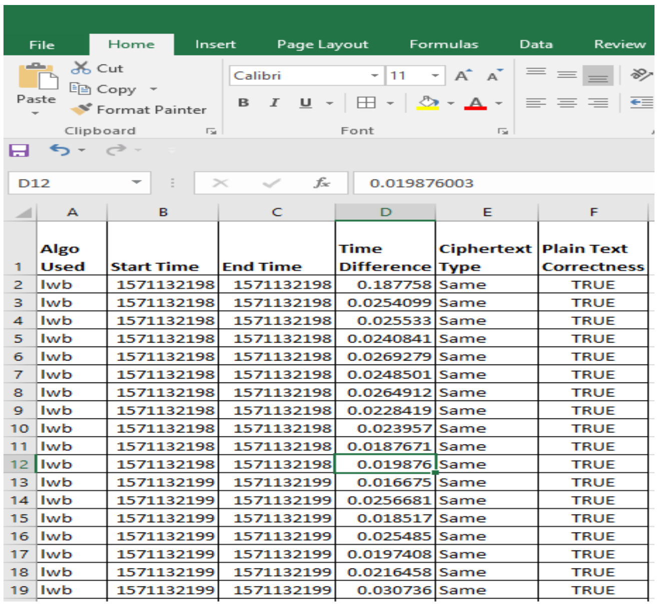
Fig 8. Lightweight P-Box protocol on Arduino

Fig. 9 shows the results of lightweight P-Box protocol implementation on cloud with details of node time recordings for encryption and decryption along with plaintext and cipher text generated



	Algo Used	Encryption Time	Decryption Time	Plain Text	Cipher Text
2	lwb	950	325	41	b35c0f9ee8bf56c2
3	lwb	598	299	51	b4450c9d6266774e
4	lwb	585	294	37	8b0a21b448b8d0d9
5	lwb	685	321	45	5d4d719948bf4542
6	lwb	608	307	32	2d5b629f48b705c2
7	lwb	637	309	57	b35e2f2feb3f9ea3
8	lwb	15101	295	38	2b5c019cc8b754ca
9	lwb	611	308	48	5b4c1398c8bf54c2
10	lwb	615	308	38	2b5c019cc8b754ca
11	lwb	607	303	34	8c0c21b6493f1131
12	lwb	618	308	46	754d0db768b74f52
13	lwb	590	295	56	b55f2d2eeb3f8b21
14	lwb	597	307	46	754d0db768b74f52
15	lwb	15167	294	47	734c2fb6e8b752d2
16	lwb	594	307	51	b4450c9d6266774e
17	lwb	623	308	47	734c2fb6e8b752d2
18	lwb	577	295	33	8b2c21b4c8b750da
19	lwb	602	308	34	8c0c21b6493f1131
20	lwb	620	307	31	2b7a019c48b0d4c9

Fig 9. Lightweight P-Box protocol cloud time recording



	Algo Used	Start Time	End Time	Time Difference	Ciphertext Type	Plain Text Correctness
2	lwb	1571132198	1571132198	0.187758	Same	TRUE
3	lwb	1571132198	1571132198	0.0254099	Same	TRUE
4	lwb	1571132198	1571132198	0.025533	Same	TRUE
5	lwb	1571132198	1571132198	0.0240841	Same	TRUE
6	lwb	1571132198	1571132198	0.0269279	Same	TRUE
7	lwb	1571132198	1571132198	0.0248501	Same	TRUE
8	lwb	1571132198	1571132198	0.0264912	Same	TRUE
9	lwb	1571132198	1571132198	0.0228419	Same	TRUE
10	lwb	1571132198	1571132198	0.023957	Same	TRUE
11	lwb	1571132198	1571132198	0.0187671	Same	TRUE
12	lwb	1571132198	1571132198	0.019876	Same	TRUE
13	lwb	1571132199	1571132199	0.016675	Same	TRUE
14	lwb	1571132199	1571132199	0.0256681	Same	TRUE
15	lwb	1571132199	1571132199	0.018517	Same	TRUE
16	lwb	1571132199	1571132199	0.025485	Same	TRUE
17	lwb	1571132199	1571132199	0.0197408	Same	TRUE
18	lwb	1571132199	1571132199	0.0216458	Same	TRUE
19	lwb	1571132199	1571132199	0.030736	Same	TRUE

Fig 10. Lightweight P-Box protocol cloud time recording

Fig. 10 shows the results of lightweight P-Box protocol implementation on cloud with details of node time recordings for encryption and decryption along with plaintext and cipher text correctness.

V. CRYPTOGRAPHIC ATTACKS

Cryptography is a term to indicate the relation between plain text and cipher text. All cryptographic operations requires a secret key which can be symmetric or asymmetric. To generate a secured symmetric shared key between two nodes, we have used PUF[7] based technique.

PUF technique is lightweight and based on digital footprint concept making it most suitable in IoT applications. This secret keys plays an important role in retrieving the original data. If the attacker becomes successful in retrieving the plain text from cipher text, then it will become easier to get access to the secret key used in encryption, making the system compromised.

Various attacks are categorized as follows:

- Cipher text only attacks
- Chosen plaintext attack
- known-plaintext attack
- Dictionary attack
- Brute force attack

Cipher text only attacks: Attacker has access to the set of cipher text but not the corresponding plaintext. If attacker somehow manages to get the key then he/she can get access to the plaintext.

The Chosen plaintext attack(CPA): In this model the attacker can choose random plaintext and obtain the corresponding cipher text. Here the attacker has cipher-plain text pair of his choice making it simple to retrieve the encryption key. The goal of the attack is to gain some further information which reduces the security of the encryption scheme.

The known-plaintext attack (KPA): In this model attacker has knowledge of plain text for some specific cipher text. Based on this information attacker tries to reveal rest of the cipher text. These can be used to reveal further secret information

Dictionary attack: Attacker maintains a dictionary of retrieved plaintext and cipher text over a period of time. During attack if he/she found similar cipher text then from dictionary plaintext can be found.

Bruteforce attack: An attacker, based on the key length tries to determine all the possible key combinations and for each possible key, attacker also checks cipher text to plain text conversion.

As Lightweight P-Box permutation algorithm uses symmetric key, we have demonstrated Chosen plaintext attack and known-plaintext attack as these are most suitable in symmetric key algorithm.

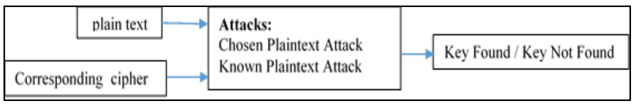


Fig 11. Cryptographic attacks

VI. RESULTS

This section illustrates the results w.r.t. time taken by PRESENT algorithm and Lightweight P-Box Permutation algorithm in IoT environment as well as security analysis w.r.t. different types of attacks

Time analysis:

We have tested these algorithms with two test cases:

1. Transmitting 8 bit data encrypted with 32 char key using PRESENT algorithm
2. Transmitting 8 bit data encrypted with 32 char key using Lightweight P-Box Permutation

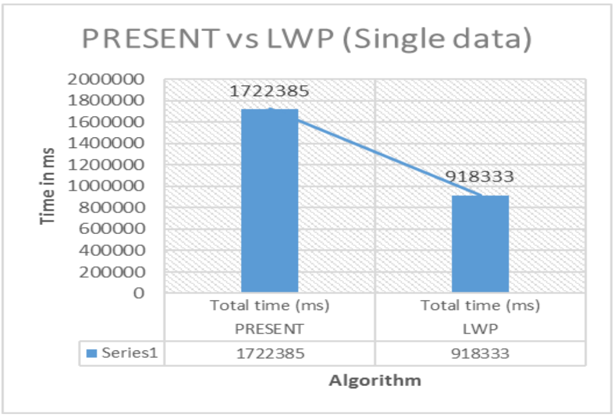


Fig 12. PRESENT VS LWP(Single data)

We have also tested these algorithms with bulk of data:

1. Transmitting total 300 readings each of 8 bit data encrypted with 32 char key using PRESENT algorithm
2. Transmitting total 300 readings each of 8 bit data encrypted with 32 char key Lightweight P-Box

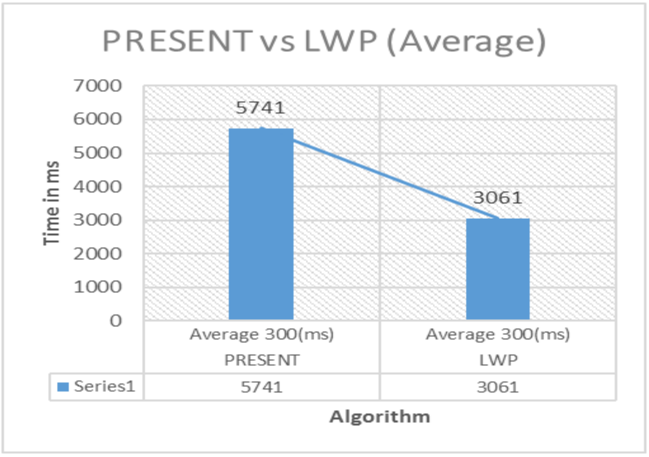


Fig 13. PRESENT VS LWP (300 cases data)

From the results it is proved that lightweight P box permutation is much lighter with respect to time.

Security Analysis:

Lightweight P-Box permutation algorithm is based on PRESENT[8] ultra lightweight cipher, where we used the same ‘S’(Substitution) box as of PRESENT. In [8] it is proved that to mount key recovery attack Linear and Differential would require large data which exceeds the available text. Also we have used GRP as bit permutation operation in Lightweight P-Box Permutation [9] which is resistant to linear and differential cryptanalysis and has better differential property when compared with other bit permutation operations.

We have carried out two types of attacks on the system and shown that our system is resistant to these attacks.

Both Chosen plaintext attack(CPA) and known-plaintext attack (KPA) carried out with different combinations of keys, sample of plain text and cipher text. The attack tried for possible recovery of keys ranges from keys/1sec, keys/5sec and keys/10sec as shown in following tables:

1. Chosen plaintext attack(CPA):

Sr. No	Key	S ample	Cipher Text	Keys /1 Sec	Key s/5 Sec	Key s/10 Sec
1	1234567812345678	abcdefgh	6f24e777bf308c5d	239977	1170935	2330736
2	1234567812345678	12345678	12182f5f2620d38f	227080	1193708	2381189
3	0000000000245585	neelpate	8ad006ebfeba20f0	235637	1189315	2366312
4	0000000000245585	pratiksp	041856d53d54c82a	232463	1197129	2357326
5	1111111122222222	98653214	855454d179f7c8ed	210740	1157169	2376122
6	1111111122222222	44448888	9556a6549650657b	218350	1148291	2367187
Average				227375	1176091	2363145

Table 1. CPA test

I. <u>Infeasibility Proof</u>

Using the above recorded values and the details of the algorithm we can calculate the following observations:

Observation I:

Rate of Keys Scanned [Avg] = 227375 Keys/Sec $\cong 2^{18}$ *Keys/Sec*

Observation II:

Time required to scan 2^{128} *Keys* = $2^{128} / 2^{18} = 2^{110}$ *Sec ------------[Total Keys Possible =* 2^{128}*]*

∴ *Time required in years* = $2^{110} / 2^{25} = 2^{85}$ *Years ----------[1 Year = 31,536,000 Sec* $\cong 2^{25}$ *Sec]*

Hence, by calculating observation I & II and observing the magnitude of the above mentioned “Time required in years” we can say that – Finding the key is infeasible in the case of CPA.

1. Known-plaintext attack (KPA)

The known-plaintext attack (KPA) carried out as shown in following tables:

Sr. No	Key	Sample	Cipher Text	Keys/1 Sec	Keys/5 Sec	Keys/10 Sec
1	1234567812345678	abcdefgh	6f24e777bf308c5d	250215	1247085	2498580
2	1234567812345678	12345678	12182f5f2620d38f	248064	1257093	2499022
3	0000000000245585	neelpate	8ad006ebfeba20f0	248262	1247441	success
4	0000000000245585	pratiksp	041856d53d54c82a	237079	1249360	success
5	1111111122222222	98653214	855454d179f7c8ed	251277	1248441	2487004
6	1111111122222222	44448888	9556a6549650657b	248510	1242500	2474324
			Average	247235	1248653	2489733

Table 1. KPA test

I. <u>Infeasibility Proof</u>

Using the above recorded values and the details of the algorithm we can calculate the following points:

Observation I:

Rate of Keys Scanned [Avg] = 247235 Keys/Sec $\cong 2^{18}$ *Keys/Sec*

Observation II:

Time required to scan 2^{128} *Keys* = $2^{128} / 2^{18} = 2^{110}$ *Sec ------------[Total Keys Possible =* 2^{128}*]*

∴ *Time required in years* = $2^{110} / 2^{25} = 2^{85}$ *Years ---------[1 Year = 31,536,000 Sec* $\cong 2^{25}$ *Sec]*

Hence, by calculating observation I & II and observing the magnitude of the above mentioned “Time required in years” we can say that – Finding the key is infeasible in the case of KPA.

Thus both CPA and KPA both shows that attacks on this encryption mechanism are not possible.

Algorithm	Observation I	Observation II
Chosen-plaintext attack (CPA)	*Rate of Keys Scanned [Avg] = 227375 Keys/Sec* $\cong 2^{18}$ *Keys/Sec*	*Time required in years* = $2^{110} / 2^{25} = 2^{85}$ *Years*
Known-plaintext attack (KPA)	*Rate of Keys Scanned [Avg] = 247235 Keys/Sec* $\cong 2^{18}$ *Keys/Sec*	*Time required in years* = $2^{110} / 2^{25} = 2^{85}$ *Years*

VII. CONCLUSION

Applications in IoT environment needs exchange of crucial data at regular intervals which can be attacked and misused by the attacker, leading to devastating consequences. To deal with such situations, the data is always encrypted before being transmitted over the network. In our earlier work we have used lightweight P-box permutation mechanism while exchanging data between two nodes and proved that time taken to encrypt the data using PRESENT is more than Lightweight P-Box permutation mechanism. In this paper, we have considered a system in which smart meter data is exchanged at regular intervals using lightweight encryption mechanism. We have elaborated different types of cryptographic attacks and tested cryptographic attacks such as Chosen Plaintext Attack and Known Plaintext attack on Lightweight P-Box permutation encryption and proved that our algorithm is resistant to attacks and attacks does not reveal the secret key used between the nodes. We have also proved that finding key is not feasible as the time required in years.

REFERENCES

1. Satin Asri, Bernardi Pranggono, “Impact of distributed denial-of-service attack on advanced metering infrastructure”, Springer Wireless Press Communication, 2015
2. Zubair Baig,Abdul-Raoof Amoudi, “An analysis of smart grid attacks and countermeasures”, Journal of communications, August 2013
3. Ping Yi, Ting Zhu, Qingquan Zhang,Yue Wu,Jianhua li, “A Denial of service attack in advanced metering infrastructure network”,IEEE, 2014
4. David G. ,Jun H, “AMI Threats,Intrusion Detection Requirements and Deployment Recommendations”, Smart Grid Communications,2012
5. Dong Chen, Sandeep Kalra, David Irwin, “ Preventing occupancy detection from Smart Meters”, IEEE transactions on smart grid, Sept 2015

6. Yonghe Guo,Chee-Wooi Ten, " Modeling Distributed Denial of Service Attack in Advanced Metering Infrastructure", IEEE, 2015
7. Urbi Chatterjee, Rajat Subhra Chakraborty, And Debdeep Mukhopadhyay, "A PUF-Based Secure Communication Protocol for IoT", Indian Institute of Technology Kharagpur, ACM , 2017
8. A. Bogdanov , L.R. Knudsen, G. Leander , C. Paar , A. Poschmann , M.J.B. Robshaw, Y. Seurin , and C. Vikkelsoe , "PRESENT: An Ultra-Lightweight Block Cipher", Ches Springer-Verlag Berlin Heidelberg, Lncs 4727, Pp. 450–466, 2007.
9. Aruna Gawade, Narendra Shekokar, "Lightweight Cipher Using GRP Bit Permutation & Tweak", ISDA,Springer,2018
10. https://data.london.gov.uk/dataset/smartmeter-energy-use-data-in-london-households

AUTHORS PROFILE

Prof. Aruna Gawade completed M.Tech (Computer Engg)from NMIMS University, Mumbai. Currently persuing Phd. in Computer Engg (IoT Security) under Mumbai University. Working as a assistant professor in Computer Engg. Dept at SVKM's Dwarkadas J. Sanghvi College of Engineering, Mumbai (Autonomous college affiliated to University of Mumbai).Her Specialization is in IoT ,Networking, Security, etc. She has published more than 15 papers at various International & National conferences and renowned journals.

Dr. Narendra Shekokar has received his PhD in Engineering (Network Security) from NMIMS University, Mumbai and he is working as a Professor,Dept. of Computer Engineering at SVKM's Dwarkadas J. Sanghvi College of Engineering, Mumbai (Autonomous college affiliated to University of Mumbai). He was a member of Board of Studies at University of Mumbai for more than 5 years and he has also been a member of various committees at University of Mumbai. His total teaching experience is 22 years. Dr. Narendra Shekokar is Ph.D guide for 6 research fellows and more than 25 students at Post Graduation level. He has presented more than 50 papers at International &National conferences and has also published more than 20 research papers in renowned journals. He has received the Minor Research Grant twice from University of Mumbai for his research projects. He has delivered expert talk and chaired a session at numerous events and conferences.