

Lightweight Cryptographic Algorithms for Secure IoT Devices

Kollipara Hemanth, Kalakota Koushik

Ruby D (Professor VIT, Vellore)

venkata.nagahemanth2018@vitstudent.ac.in, kalakotasai.koushik2018@vitstudent.ac.in

Abstract

The Internet of Things (IoT) has become an emerging technology and is expected to connect billions of more devices to the internet in the near future. With the passage of time, more and more devices like wearables, smart home systems, industrial automation devices are getting connected to the internet. These devices are often embedded systems housing a low power processor chip that acts as the brains of the system and is connected to a variety of sensors, that collect valuable data. This data can often be critical and thus raising the question of security threats. In order to maintain the assurance of this technology, it is vital to ensure the security of such low power devices. Also, IoT devices mostly transfer data using wireless communication networks, introducing more vulnerabilities like man-in-the-middle-attack, eavesdropping. These security problems are common for any device communicating over the internet because of its intrinsic open nature. Encryption would solve this problem of data security for regular devices like computers, smartphones, but the use of the same cryptographic algorithms for IoT devices which are usually embedded systems is questionable since the hardware architecture in these devices is smaller and low powered. Conventional cryptographic algorithms used in typical systems are power-hungry and computationally intensive, making them infeasible to be used in IoT devices as present many challenges in designing algorithms since they run on low powered chips limiting the performance, limited memory, limited transmission bandwidth. Hence there is a requirement to adopt lightweight cryptographic algorithms which satisfy the security requirements using low power consumption, unlike conventional algorithms that use a lot of energy and computational power performing many rounds of encryption, which is the constraint in the given scenario. Hence, we propose an end-to-end secured IoT system which ensures the integrity of the system is never compromised using lightweight cryptographic algorithms. We propose a three-module system, where one handles authentication, another handles data encryption, and another ensures of the integrity of data is not compromised. This kind of system is designed to deal with security challenges in IoT devices, ensuring adequate security to the data at the same time reducing the computational footprint with the use of lightweight cryptography.

Keywords: *IoT, Security, Multi-factor Authentication, Time-based OTP, Encryption, Decryption, Key-generation, Three-prime RSA, SITS, Digital signatures, Hash function*

I. Introduction

IoT is slowly becoming an integral part of our daily life. As people use more and more smart devices which include smartwatches, fitness trackers that collect personal data to smart home products like smart refrigerators, locks, fire systems, security systems which transmit critical data around the internet. As the cost of connectivity to the internet is getting cheaper every day, and with accessibility increases allowing more and more people to connect to the internet along with their smart devices, contributing to the growth of IoT technology. These devices are part of a complex network of similar devices, exchanging lots of information over the network. Each device in the network gathers data from their corresponding sensors which could be ranging from a temperature sensor that collects home temperature to camera sensors that monitor traffic. These sensors generate a humongous amount of data, and this data is communicated between other devices over the internet. Thus, IoT is changing the landscape of the conventional internet to the next level by connecting everything to the internet. So, security concerns related to data confidentiality, integrity and authentication must be considered seriously. These devices must be able to safeguard the privacy of the user data and should not compromise the integrity of the system. Nevertheless, IoT, due to its openness in terms of its connectivity, introduces new security challenges since its inherently vulnerable to various threats like information leakage and unauthorized usages. One backdrop of IoT devices is that they are prone to physical attacks resulting in exposure of data stores in the components. Also, IoT devices are most commonly connected via wireless networking, making them vulnerable to security attacks, unauthorized access which might result in data loss, data leakage and damage to the entire network. Hence to address these security concerns, there is a need to use the appropriate cryptographic algorithms to maintain the integrity of the system. However, conventional cryptographic algorithms are suitable for these smart devices since they are constrained by energy consumption, computational power, memory utilization, network bandwidth. Hence the cryptographic solutions must be appropriate for low-resource hindered devices.

This project was prepared in such a way to meet the above requirements as efficiently and reliably as possible. The proposal is to implement various lightweight cryptographic encryption and security algorithms so that the data being transferred between these IoT devices is very secure and is not vulnerable to breaches. The main focus is to implement these algorithms into one shared platform for all IoT devices from secure authentication to data encryption.

II. Literature Survey

A. Time-based OTP Authentication via Secure Tunnel

The implementation is carried out at the application level. Two applications, a server application that is running and a client application are used in the entire process. The server application keeps accepting connection requests and when a connection request from client is sent to the server, it gets accepted and connects to server. The client application first needs to go through a one-time registration process with the server application in order to get itself registered in the server database. The pre-shared secret key that used to generate the TOTP is now shared between the client and server, now enabling both of them to generate the same OTP independently. Once the process is complete, the client application can use this secret key and the present timestamp at which the TOTP is triggered to generate the current TOTP that is used to establish an authenticated connection. The authors have used the time-based OTP for the secure authentication over the TLS/SSL tunnel using SHA based hashing and two-fish encryption algorithm to add a higher level of security. With the drawbacks being, secret keys at the registration process are exchanged using two fish encryption algorithm which is more vulnerable to brute force attack. Also, the server does not take into consideration the device from which the connection is established, making it vulnerable to spoofing attackers made my attacker who could have managed to break into the network [1].

B. OTP Authentication based on Identity-based ECC for IoT

In this paper, the author has proposed an end-to-end authentication in low power IoT devices using OTP based algorithm that is lightweight, robust and scalable by using the concept of Identity based elliptic curve cryptography. In this proposed methodology, a client device first requests the respective server application to generate OTP. The server responds to the request by generating OTP and then shares this to the request remote device. For the client application to get authenticated, it must first submit its basic credentials, login and password and then the OTP to the server application. The server device verifies the received credentials and then proceeds to perform OTP validation. The algorithm first generates two dissimilar integral prime numbers p and q where $p = (2)n \pm c$, where $c \leq \log 2n$ and $p \equiv 3(\text{mod}4)$ and $q|p+1$, super-singular elliptic curve, torsion point and a secret key. The device with device id Dev_id obtains a public key which is a torsion point on elliptic curve $P_{\text{Dev-id}}$ and a private key Dev-id which is also a torsion point. In the validate phase, the client device communicates with the server application submitting to the OTP request and the server then validates the OTP received from the client with the one generated by itself. Upon successful validation it accomplishes the job of authentication between client device and remote server. The proposal also includes optimizations in the computation of various cryptographic parameters over the elliptic curve and making the scheme more lightweight. Thus, the use of OTP along with the existing authentication scheme is sufficient and necessary requirement to mitigate the security as mentioned earlier risks. Drawbacks being, OTPs are generated on the server and sent to the user, thus relying on external infrastructure for getting the OTP [2].

C. Lightweight RSA Algorithm using Three prime numbers

This RSA algorithm utilizing three primes rather than two prime standard RSA algorithm. First public key is generated and published from the intended side where the message is originated. The public key published is used to encrypt the plain text and this cipher can only be decrypted by using the private only which is not disclosed. The public key open to everyone cannot be used to decrypt the transmitted message making the transmission secure, enabling the exchange of messages in a protected way maintaining data privacy. The methodology proposed, is an improvement to the existing RSA algorithm in terms of speed in key generation and decryption. This proposal takes advantage of three prime numbers and the concept of Chinese Remainder Theorem. A new prime number was added to the algorithm making the process of factorization a little bit easier reducing the time taken for key generation. This added component increases the complexity of the algorithm making the analysis of factorizing the Euler's totient much more difficult than the traditional algorithm that uses two primes. This method not only reduces the time for generation, it also has the added benefit of enhancing the algorithm by protecting it from a few attacks namely timing attack, modulus attack, plaintext attack, cipher-text attack. Drawback being, even though the proposed modified RSA algorithm was faster than the standard RSA especially when taking in considerations key generation and decryption steps it doesn't speedup the process of encryption significant enough compromising its efficiency a little bit [3].

D. Lightweight Security algorithm for low-power IoT using ECC

The proposed methodology is based on the Elliptic Curve Diffie-Hellman Key Exchange scheme. It focuses on encryption and decryption algorithms that are lightweight and low power consuming algorithm but at the same time is robust and secured. Also, the paper proposed some enhancements in the computation process of the ECDH Key Exchange. This kind of key exchange on an unsecured channel using elliptic curves is widely used and use usually uses the Diffie-Hellman Key agreement mechanism to exchange the keys. This public-key cryptography, when compared to others like RSA, offers the same security while using smaller keys lowering the power consumption and reducing the computational load on the system generating the keys. This allows calculations with much more speed and also using low memory. ECC utilizes an elliptic curve defined over a finite field F_q , which and contains the affine points $(x,y) \in F_q \times F_q$ that satisfy the Weierstrass equation. Starting with a number q , which could be a prime or one that satisfies the 2^m and two parameters for the elliptic curve, namely a and b . These values define the elliptic curve equation $E_q(a,b)$. After forming the equation, a base point (x_1,y_1) in $E_q(a,b)$. A small positive integer that satisfies $nG=0$ of order n . The calculation of finding the modulus of a fractional number is a little harder, but an easy and simple method is proposed. Thus, the proposed methodology was found to be better than other comparisons made, consuming low power but robust. The main challenge in the proposed lightweight Elliptic Curve Diffie-Hellman Key Exchange is the complexity of the operations involved and tricky to implement securely, particularly the standard curves [4].

E. Design and Implementation of Tiny Encryption Algorithm

Tiny encryption algorithm is a block cipher algorithm, and it is one of the most simple and efficient algorithms used for data encryption. Moreover, the algorithm is designed following the memory constraints in IoT devices and have Shannon's properties of complete confusion and diffusion by implementing substitution and permutation boxes. Tiny Encryption Algorithm takes 64 data bits using a 128-bit key with 32 rounds. From the previous rounds, left and right inputs are derived, and from the 128-bit key, a subkey is derived. In each round, delta multiples are used where delta is a constant, which is 2654435769, and to ensure subkeys used in each round are different golden ratio is used. TEA is a Feistel cipher that splits the plain text into two halves, and the subkey will be applied to one half of plaintext in the round function, F. Then, the output of F will be XOR with another half before the two halves are swapped. All the same, patterns are applied to all the rounds except the last round, where there is no swap. The algorithm consists of dual bit shifting, which mixes data regularly, and XOR and ADD operations provide diffusion and confusion for the secure block cipher. TEA has few weaknesses. It experiences identical keys attack that is each key is equal to three others. Microsoft's Xbox game console is hacked due to this weakness. The related-key attack can also be a danger for the tea encryption [5].

F. SIMON: Block Ciphers for the Internet of Things.

The SIMON cipher is a symmetric block cipher algorithm with features like simple mathematical computations SIMON is a bit-based algorithm. Utilize the following procedures on the n-bit word: Bitwise Adding, Bitwise XOR, right circular shift and Bitwise left for encryption and decryption. The round function present in the block diagram depicts SIMON; the block's upper and lower words are x_{i+1} and x_i , which is a bit word. These two words hold the initial input, which is called plaintext. As bitwise XORing, bitwise ANDing and left circular shift operation are performed in round function. Circular left the shifting, and bitwise AND operation are performed in every round on the upper word, and it is XOR with lower word and the round key. The upper word is written. As a result, value and its value is shifted to the lower word. Until the given number of rounds is reached, round functions continue to run repeatedly. Simon Block Cipher has few weaknesses. Known-key distinguishing attack model is one major issue with Simon block cipher, and Related-key attacks have a fair chance of breaking into the encryption of Simon block cipher [6].

G. SIT: A Lightweight Encryption Algorithm for Secure Internet of Things

SIT is a lightweight encryption algorithm that takes 64-bit plain text as an input and outputs a 64bit ciphertext. The algorithm is based on an asymmetric key, and it consists of 5 encryption rounds. Each encryption round consists of various mathematical functions to achieve Shannon properties of both confusion and diffusion. The proposed algorithm consists of only five rounds. It is energy efficient, and the algorithm uses the Feistel network of permutation and substitution functions to create sufficient confusion and diffusion to face the attacks. The generation of the key is an essential process in symmetric key algorithms. The key generation involves complex mathematical operations as the encryption algorithm consists of five rounds, and a separate key is used for each round. The proposed algorithm takes an input of 64 bits of data and a 64-bit key to encrypt the data. The key used to encrypt the data is taken as input from the user. This key will be given as input to the key expansion block. In the key expansion procedure, several operations create confusion and diffusion required to confront attacks and finally generate five different keys each of 16 bits. The generated keys are used in each round of the encryption and decryption process, and these are strong enough to remain the same during an attack. The key expansion architecture consists of several F functions, which ensures sufficient shuffling of bits to ensure no dependency between input key [7].

H. Message Authentication using Blake-2

Whenever client machine sends the message to the server, the Message will get hashed utilizing BLAKE2 then it will send the message and hash value to the server. Server will again hash the got message and check if got hash is matching or not with received hash. Blake2 is an improvised version of SHA-3 it comes in two flavors blake2b and blake2s, blake2b does 12 rounds against 16 for blake2. Blake2b rotation is also optimized that is replacing 16, 12, 8 and 7 with 32, 24, 16 and 63. Blake2b is using little-endian which may provide a little increase in speed, and makes implementations process a little simple. And it is counting bytes rather than bits which reduces the risk of errors [8]. A 16-word state V_0 to V_{15} is generated such that the inputs output different initial states. The state is served as a 4×4 matrix that is filled as follows:

chain value $h = h_0, h_1, h_2, h_3, h_4, h_5, h_6, h_7$

constants = IV_0 to IV_7

flags = f_0 to f_1

Once the state v is initialized, the compression function iterates a series of 12 rounds.

A round is a transformation of the state v that computes

$G_0(v_0, v_4, v_8, v_{12}), G_1(v_1, v_5, v_9, v_{13})$

$G_2(v_2, v_6, v_{10}, v_{14}), G_3(v_3, v_7, v_{11}, v_{15})$

$G_4(v_0, v_5, v_{10}, v_{15}), G_5(v_1, v_6, v_{11}, v_{12})$

$G_6(v_2, v_7, v_8, v_{13}), G_7(v_3, v_4, v_9, v_{14}).$

I. Lightweight Hash Function Photon

Whenever client machine sends the message to the server, the Message will get hashed utilizing Photon then it will send the message and hash value to the server. Server will again hash the got message and check if got hash is matching or not with received hash. Photon uses the sponge construction, it is composed of c bits of capacity and r bits of bitrate which are firstly initialized with initial value and the hash message after padding is divided into r -bit blocks. Then during the absorbing phase all the message blocks have been handled after that r bits of internal states have been extracted before applying internal permutation p . It gives provable protection from classical linear/differential cryptanalysis, and opposes all known and late attacks against hash functions with an enormous security edge. In this hash function the width $b=r+c$ characterizes the block size that is used for the permutation P . This block size is the amount of the rate r and the capacity c . The rate r is the block length used in the algorithm. It is the length of one message part. The capacity c straightforwardly impacts the security level of the hash function. It is the part of the previous permutation that is utilized for the next one. The output length n is in other construction methods the main parameter for the security level [9].

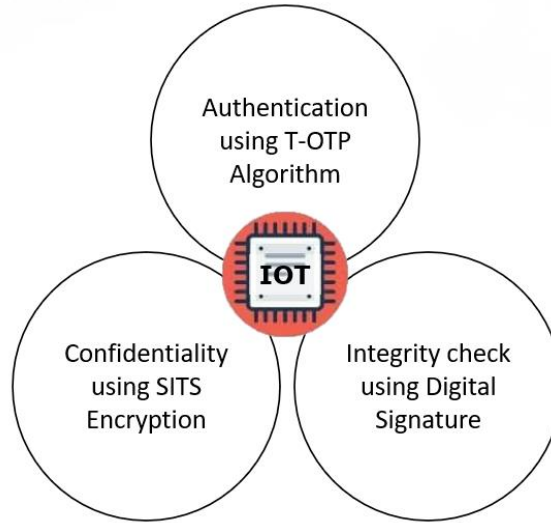
J. Lightweight Hash Function Quark

Whenever client machine sends the message to the server, the Message will get hashed utilizing Quark then it will send the message and hash value to the server. Server will again hash the got message and check if got hash is matching or not with received hash. Quark uses the sponge construction, which is parametrized by a rate (or block length) r , a capacity c , and a yield length n . The width of a sponge construction is the size of its internal state $b = r + c \geq n$. Quark uses the permutation function p inspired by the stream grain cipher and by the block cipher katan. The function p also relies on the three Boolean function f , g and h . Initialization: Message is padded by single bit 1 followed by many zeroes to reach length a multiple of r . Absorbing phase: The r -bit message blocks are XORed into the last r bits followed by the permutation. Squeezing phase: the last r bits of the state are returned as output [10].

III. System Design

The proposed system architecture consists of three basic modules, where one handles authentication, another handles data encryption, and another ensures of the integrity of data is not compromised. Authentication module uses Time-based OTP as a multifactor authentication method apart from a regular ID and password. Data Encryption module implements a lightweight symmetric key cryptography that used for encryption of data for both storage and transmission. The symmetric key is exchanged using a lightweight RSA algorithm suitable for IoT devices. The last module verifies the signature of the message in order to verify the integrity and ensure non-repudiation. This kind of system is designed to deal with security challenges in IoT devices, ensuring adequate security to the data at the same time reducing the computational footprint with the use of lightweight cryptography.

1. Architecture



2. Algorithms Used

A. User Authentication using Time-based OTP

Password is considered to be the weakest in any information security system and is the most vulnerable which can be easily cracked. But this problem has been around for while and this process of authentication using password can be further strengthened by using multi-factor authentication such as one-time-password based authentication. One such type of OTP based authentication is time-based OTP. Here the authentication between the client and server depends on a pre-shared secret key which could be exchanged between the client and server using public key cryptography if it was not done previously. Conventional OTP's are usually generated on the server and are then sent to the client by means of SMS or email, but in this time-based one time password both the server and the client can generate the OTP using the pre-shared key and the time stamp at which the OTP was generated. These time-based OTP are valid only for a specific interval of time and expire after that time interval for greater security.

Algorithm:

- First the current time stamp at which T-OTP is taken.
- It is converted into an integer and stored in an integer counter (TC) by dividing the epoch time with the specific time step (TS) which is the specific time interval after which the OTP expires.

$$TC = (\text{Epoch time})/TS$$

- $TOTP = HOTP(p_sk, TC)$
- $TOTP = TOTP \bmod 10^d$

Here , p_sk is the pre-shared secret key

d is the number of digits required in the OTP

$$HOTP(key, count) = \text{Truncate}(HMAC(key, count)) \& 7FFFFFFF$$

$$HMAC(key, value) = SHA1(key \oplus 5C5C\dots || SHA1(K, \oplus 3636\dots || value))$$

HOTP is a token and a pre-shared symmetric key based validation algorithm that generates a one-time-password. Each time an OTP is requested the token count increases and HMAC-SHA1 is performed generating a 160-bit output which then truncated and a 4-byte binary code is extracted from the truncated output.

The algorithm uses two fish encryption which is replaced by using more secure encryption efficient and lightweight encryption SIT, the authentication process can be made more secure. Furthermore, IMEI number of the client device is associated with the secret key by signing it with the IMEI number thus preventing from any attackers who managed to break into the network as they need the physical device in order to authenticate. Thus, the proposed authentication system adds an extra layer of security to the existing password stack by generating one-time-passwords using T-OTP algorithm.

B. Key Exchange in unsecure channel using Lightweight RSA using Three Prime numbers

This type of key exchange security mechanism is commonly known as public key cryptography. This mechanism uses two different types keys, one public key which is published and can be seen by everyone and another private which is stored secretly by the publisher. First the sender views the published public key and encrypts the plain text with it and send to the receiver who generated the keys. The receiver decrypts the cipher text using the secret private key that is not disclosed anywhere into plain text. The cipher text the sender sends can only be decrypted by using the private key maintaining the confidentiality of the transmitted message. Since the information disclosed by the public key is not enough to decrypt the cipher text, this mechanism is widely used for key exchange in a protected manner over an unsecured channel. One such algorithm is RSA which uses public key cryptography. The proposed algorithm is a modification of the standard algorithm which usually uses only two primes, but here we use three prime numbers. This proposal takes advantage of three prime numbers and the concept of Chinese Remainder Theorem to enhance the speed of the algorithm while key generation and decrypting the cipher text. Also, this enhancement prevents the algorithm from a few known attacks along the way.

Algorithm:

A new prime number s was added to the algorithm to increase its complexity but reducing the computational time, also incrementing the security of the algorithm as it is much harder to analysis the patterns and perform attacks.

Key Generation

- Consider three prime number p, q, s
- Calculate $N = (p \times q \times s)$
- Calculate Euler's Totient, $\phi(N) = (p-1)(q-1)(s-1)$
- Choose e , such that $1 < e < \phi(N)$, $\text{GCD}(e, \phi(N)) = 1$
- Find d such that $(e \times d) = 1 \bmod \phi(N)$
- Find d_p such that $(e \times d_p) = 1 \bmod (p-1)$, d_q such that $(e \times d_q) = 1 \bmod (q-1)$
- Find Q_{in} such that $(q \times Q_{in}) = 1 \bmod p$, if $p > q$
 $(p \times Q_{in}) = 1 \bmod q$, if $q > p$
- Public Key, $K_{pub} = (e, N)$
- Private Key, $K_{pr} = (Q_{in}, d_p, d_q, p, q)$

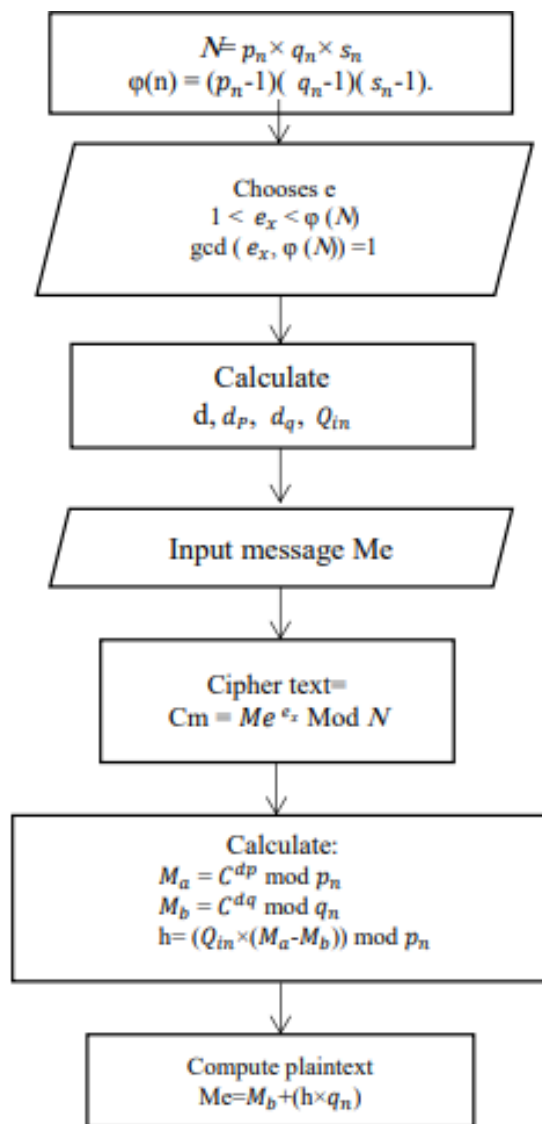
Encryption

- Cipher Text $C_m = M^e \bmod N$

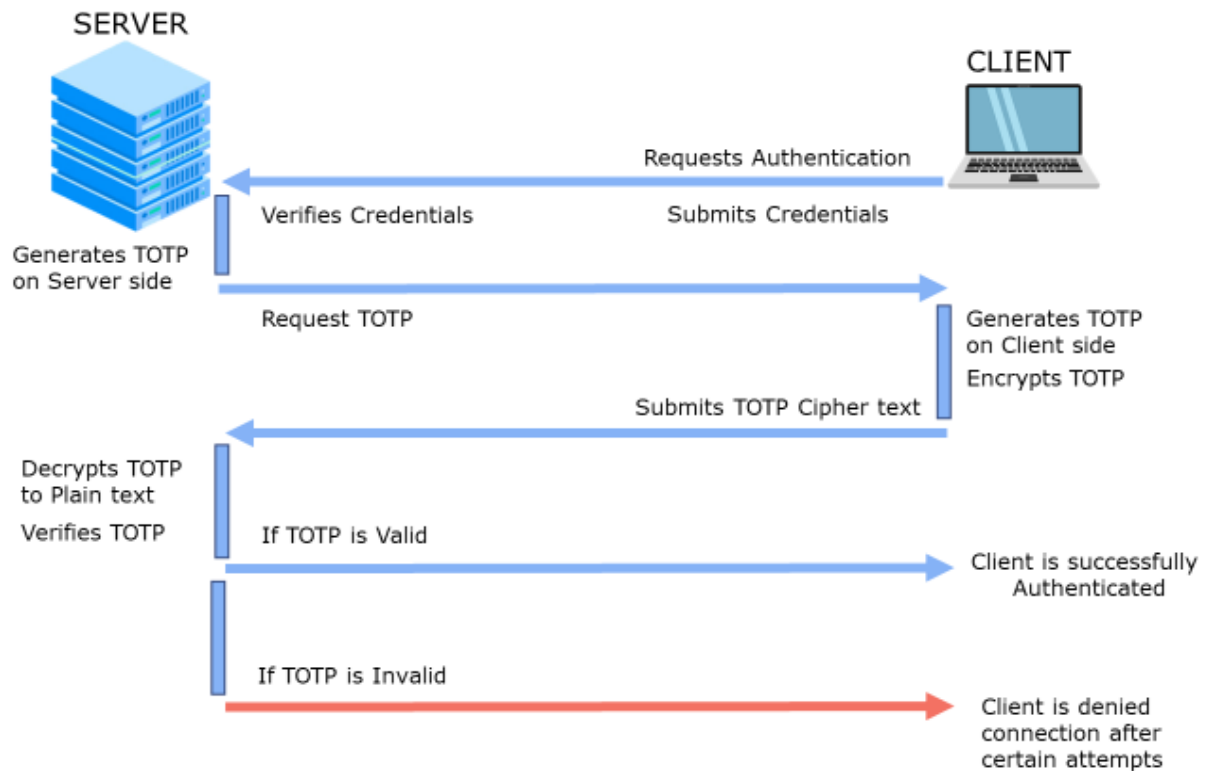
Decryption

- $M_a = C^{d_p} \bmod p$
- $M_b = C^{d_q} \bmod q$
- $h = (Q_{in} \times (M_a - M_b)) \bmod p$
- Plain Text $M_e = M_b + (h \times q)$

Block diagram of RSA using three prime numbers



User Authentication using Time-based OTP and Enhanced RSA Key exchange scheme



C. Data encryption SIT: A Lightweight Encryption Algorithm for Secure Internet of Things

This type of encryption algorithm is known as lightweight cryptography. These algorithms are designed for implementation in a constrained environment as the application involves an IoT device. Memory size and energy consumption are important factors to be considered. The proposed algorithm takes 64 bit of plain text and uses five-round encryption to encrypt the data, and each round uses a unique key generated from the key expansion block. As the algorithm consists only of five rounds and 64-bit data encryption, it is perfectly suited for IoT devices' security.

Algorithm:

Key Generation

Key generation is one of the critical steps in the SIT algorithm. The generation of keys is done through various complex mathematical operations and shuffling of bits. SIT algorithm is a Feistel based encryption algorithm, and in Feistel structure, each round requires a separate key. The encryption and decryption of the above algorithm have five games. Therefore, it requires five different keys. The initial input of a 64-bit cipher key is taken as input, and this key can be used for further key expansion.

Key expansion

- The given input 64-bit key is divided into 16 parts, each of 4 bit
- Then, an initial substitution is performed on the parts of the cipher key to send these bits to the f function, which operates on 16-bit data

$$Kb1f = Kc0 + Kc4 + Kc8 + Kc12$$

$$Kb2f = Kc1 + Kc5 + Kc9 + Kc13$$

$$Kb3f = Kc2 + Kc6 + Kc10 + Kc14$$

$$Kb4f = Kc3 + Kc7 + Kc11 + Kc15$$

- The f-function in the key expansion architecture consists of P and Q tables, which results in Shannon properties of confusion and diffusion
- The output of F-function Kaif, which is 16-bit, is written as a 4x4 matrix by means of row. These four matrices are Km1, Km2, Km3, Km4.
- The round keys, K1, K2, K3, K4, are obtained from these matrices where 16-bits are transformed, and the arrangements of theses bits can be observed from the block diagram

$$K1 = a4 ++ a3 ++ a2 ++ a1 ++ a5 ++ a6 ++ a7 ++ a8 ++ a12 ++ a11 ++ a10 ++ a9 ++ a13 ++ a14 ++ a15 ++ a16$$

$$K2 = b1 ++ b5 ++ b9 ++ b13 ++ b14 ++ b10 ++ b6 ++ b2 ++ b3 ++ b7 ++ b11 ++ b15 ++ b16 ++ b12 ++ b8 ++ b4$$

$$K3 = c1 ++ c2 ++ c3 ++ c4 ++ c8 ++ c7 ++ c6 ++ c5 ++ c9 ++ c10 ++ c11 ++ c12 ++ c16 ++ c15 ++ c14 ++ c13$$

$$K4 = d13 ++ d9 ++ d5 ++ d1 ++ d2 ++ d6 ++ d10 ++ d14 ++ d15 ++ d11 ++ d7 ++ d3 ++ d4 ++ d8 ++ d12 ++ d16$$

$$K5 = K1 \text{ (XOR) } K2 \text{ (XOR) } K3 \text{ (XOR) } K4$$

P and Q tables

Kci	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
P(Kci)	3	F	E	0	5	4	B	C	D	A	9	6	7	8	2	1

P-Table

Kci	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Q(Kci)	9	E	5	6	A	2	3	C	F	0	4	D	7	B	1	8

Q-Table

Block diagram of Key expansion

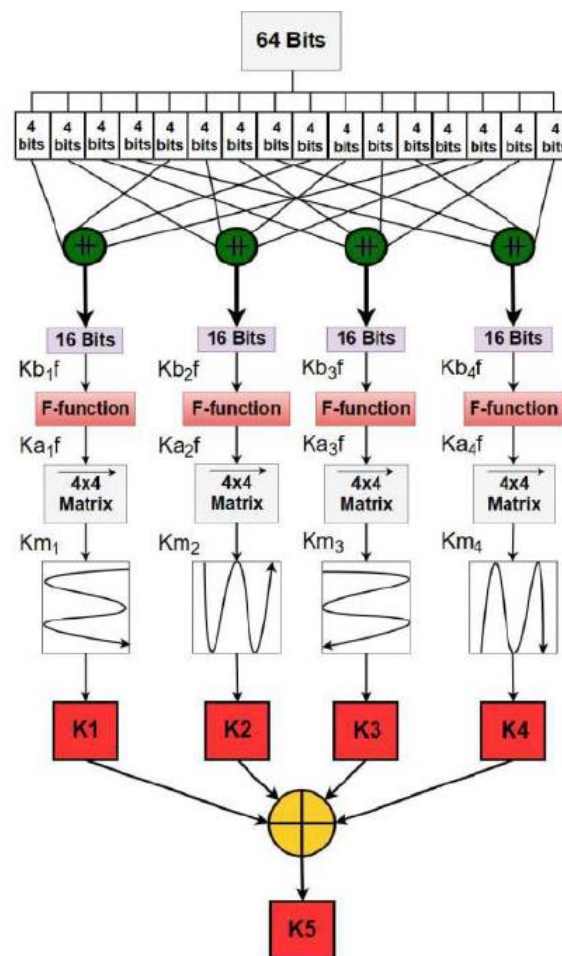


Fig. 1: Key Expansion

Encryption

After generating a five-round keys encryption algorithm consists of left shifting of bits and swapping groups of 4 bits and substitution.

- The given input plain text is divided into 16 bits of four segments
- At the end of each round swapping, the operation is applied to diminish data originality
- Bitwise XNOR is operation is performed between the respective round key and Px0-15, and the same is applied on respective round key and Px48-63 resulting in Ro11 and Ro14. The output of XNOR is then passed into f-function resulting in Efl1 and Efr1, and then Bitwise XOR function is applied between Efl1 & Px32-47 to obtain Ro12 and Efr1 & Px16-31 to obtain Ro13.

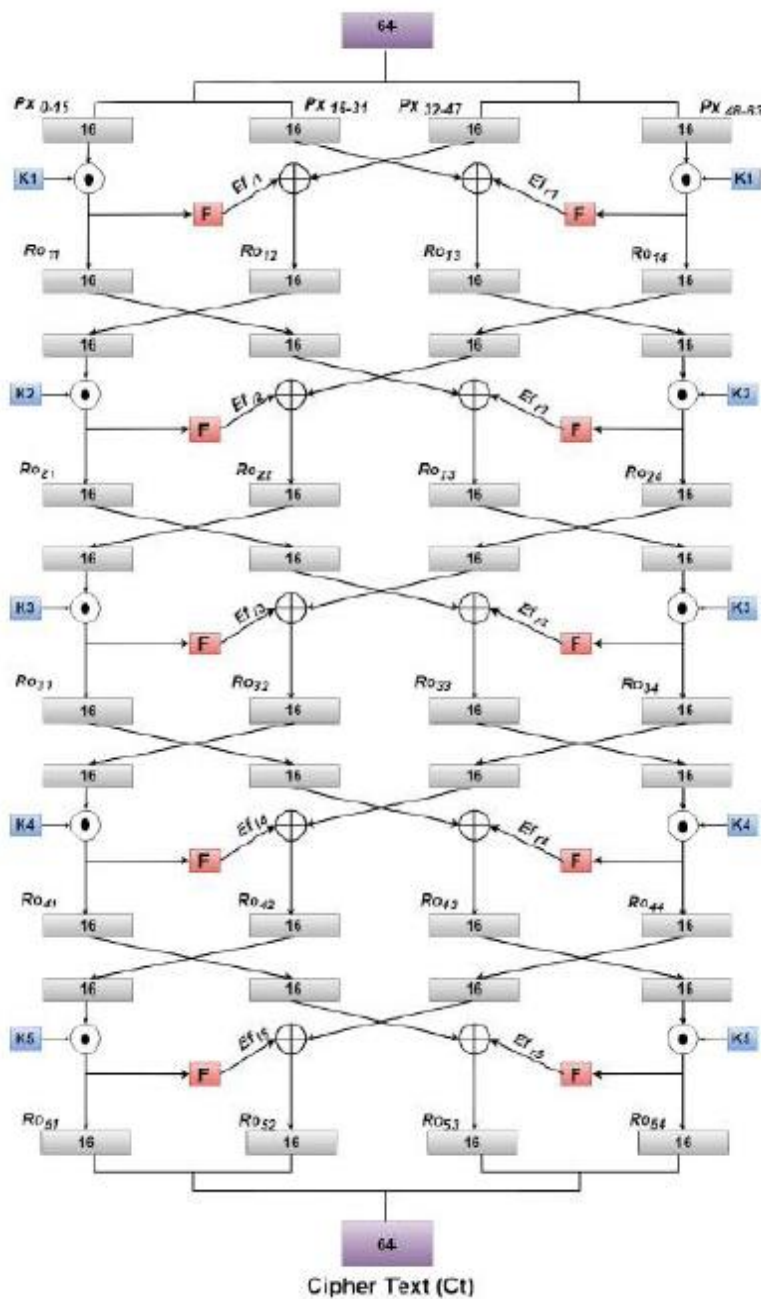
- Finally, shuffling of adjacent 16 bits are done

$$Ro_{i,j} = \begin{cases} Px_{i,j} \odot K_i & ; j = 1 \& 4 \\ Px_{i,j+1} \oplus Ef_{li} & ; j = 2 \\ Px_{i,j-1} \oplus Ef_{ri} & ; j = 3 \end{cases}$$

- Same steps are repeated for the remaining four rounds. Results of the final round are concatenated to get cipher result.

Ciphertext = R51 ++ R52 ++ R53 ++ R54.

Block diagram of SIT Encryption



D. Data Integrity verification using Digital Signatures

A digital signature is the detail of an electronic report that is utilized to distinguish the individual that communicated information. Digital signature makes it possible whether the signer is who he or she claims to be and it also checks whether the content has been tampered or not and it also helps to prove the origin of signed content. It is very important and useful to achieve information security. Digital signature makes use of key pairs for the purpose of signing or verifying. The basic idea behind digital signature is to generate cryptographic value with the help of user's private key the result obtain is often called as signature and then this signature + data passed to the verifier the verifier uses the public key with some sort of algorithm to decrypt it and thus based on the result verifier decides whether the digital signature is valid or not. DSA is one of the digital signature algorithms that works in the system of public key cryptosystem and it depends on the algebraic properties of modular exponentiation, along with the discrete logarithm problem which is viewed as computationally obstinate. A signature is created in private but can be verified in public. In other words, there is only one user that can create a signature added to a message, but anyone can check whether the signature is correct or not.

Algorithm:

Key generation

- Generate random integer prime integer p .
- Generate q such that it is prime divisor of $p-1$.
- Generate a random integer h such that $1 < h < p-1$
- Compute g as $h^{(p-1)/q} \bmod p$.
- Generate an random integer x (private key) such that $0 < x < q$.
- Compute y (public key) as $g^{**x} \bmod p$.

Creating Digital Signature

- Generate message digest h , using a hash algorithm blake2b
- Generate any random integer k such that $0 < k < q$.

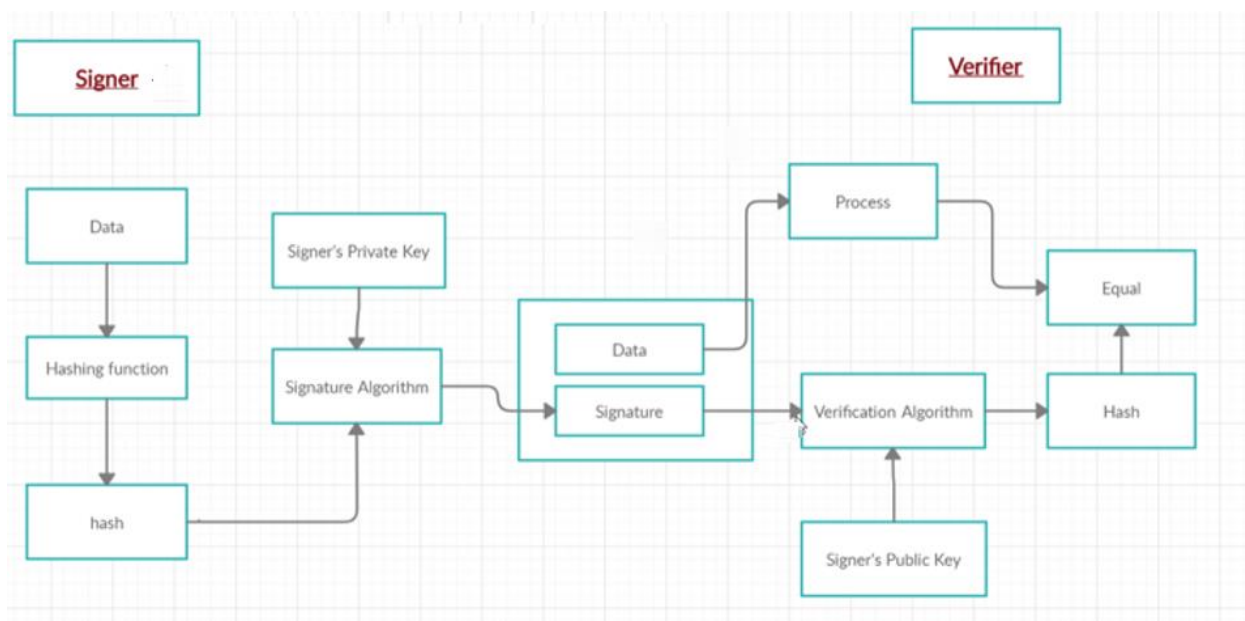
Signing

- Compute r as $(g^{**k} \bmod p) \bmod q$
- Compute s as $[k^{-1}(H(M)+xr)] \bmod q$
- Signature = (r, s)

Signature Verification

- Generate the message digest h , using the blake2b.
- Compute w as $s^{-1} \bmod q$
- Compute u_1 as $[H(M) * w] \bmod q$
- Compute u_2 as $(r * w) \bmod q$
- Compute v as $[(g^{u_1} g^{u_2}) \bmod p] \bmod q$
- If $v == r$ then signature is valid else is invalid

Block Diagram

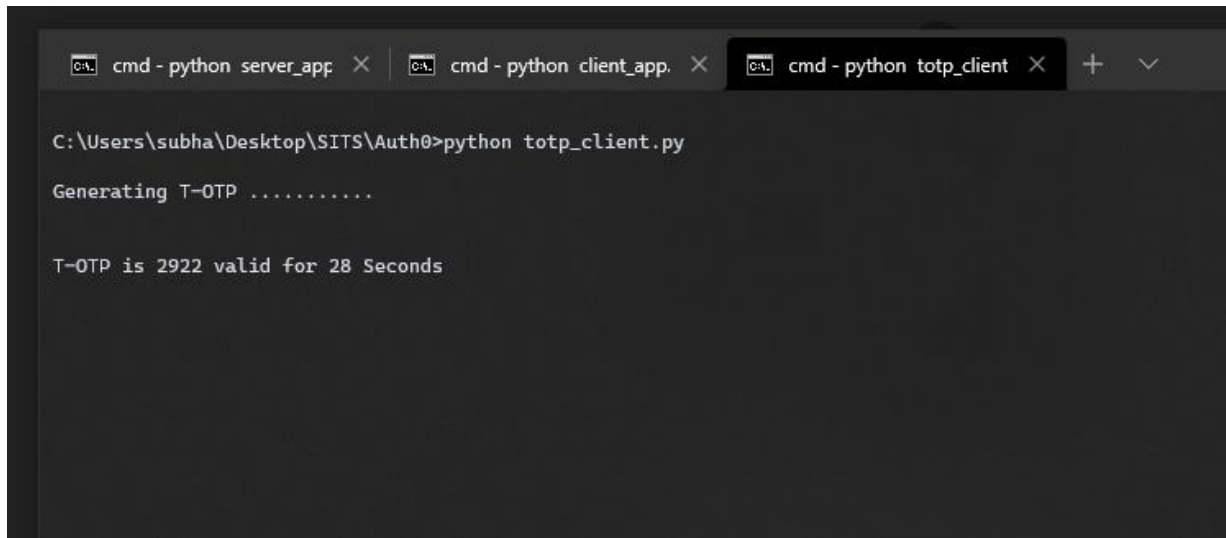


IV. Experimental Results and Analysis

A. User Authentication using Time-based OTP

Results

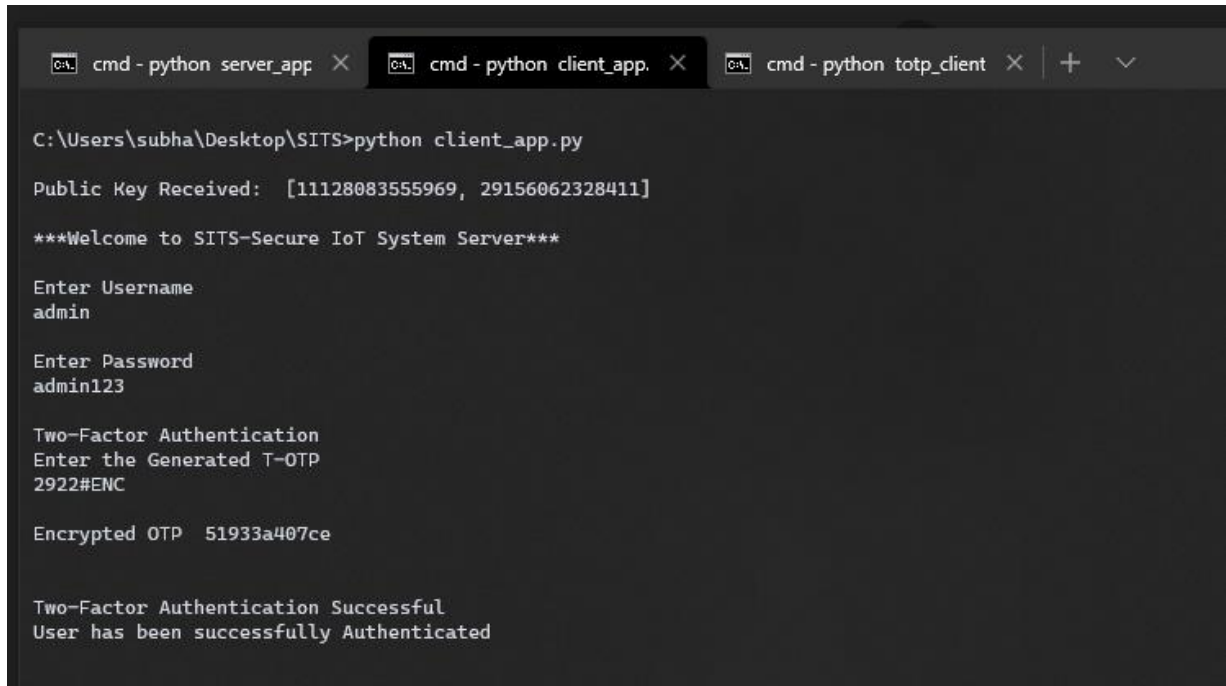
OTP generation on client side



A screenshot of a terminal window with three tabs: 'cmd - python server_app', 'cmd - python client_app', and 'cmd - python totp_client'. The active tab is 'cmd - python totp_client'. The terminal shows the command 'python totp_client.py' being executed, followed by the output 'Generating T-OTP,', and finally 'T-OTP is 2922 valid for 28 Seconds'.

```
C:\Users\subha\Desktop\SITS\Auth0>python totp_client.py
Generating T-OTP .....
T-OTP is 2922 valid for 28 Seconds
```

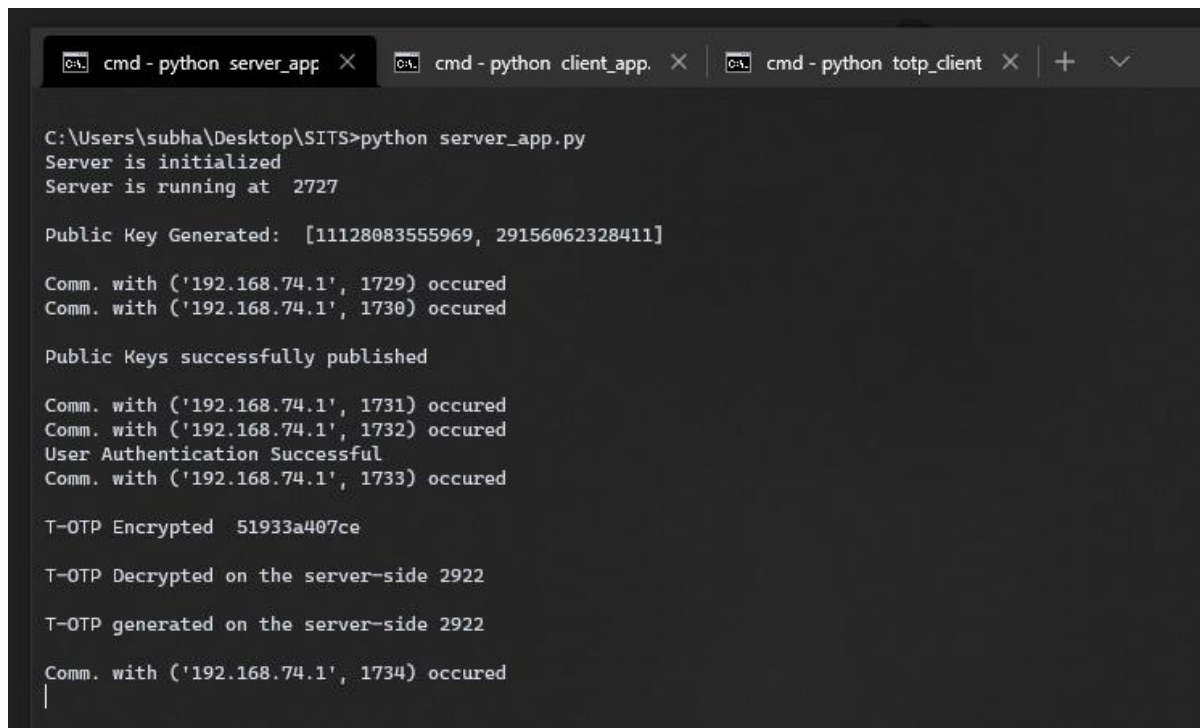
Entering credentials and valid T-OTP leading to successful authentication if client



A screenshot of a terminal window with three tabs: 'cmd - python server_app', 'cmd - python client_app', and 'cmd - python totp_client'. The active tab is 'cmd - python client_app'. The terminal shows the command 'python client_app.py' being executed, followed by several lines of output including public key information, a welcome message, prompts for username and password, two-factor authentication prompts, and a final success message.

```
C:\Users\subha\Desktop\SITS>python client_app.py
Public Key Received: [11128083555969, 29156062328411]
***Welcome to SITS-Secure IoT System Server***
Enter Username
admin
Enter Password
admin123
Two-Factor Authentication
Enter the Generated T-OTP
2922#ENC
Encrypted OTP 51933a407ce
Two-Factor Authentication Successful
User has been successfully Authenticated
```

Validating client credentials on the server side

A terminal window with three tabs: 'cmd - python server_app', 'cmd - python client_app', and 'cmd - python totp_client'. The 'server_app' tab is active. The output shows the server being initialized and running on port 2727. It displays a generated public key, receives two connection attempts from 192.168.74.1, publishes the public keys, receives three more connection attempts, and successfully authenticates a user. It then shows T-OTP encryption and decryption, and generation on the server side.

```
C:\Users\subha\Desktop\SITS>python server_app.py
Server is initialized
Server is running at 2727

Public Key Generated: [11128083555969, 29156062328411]

Comm. with ('192.168.74.1', 1729) occurred
Comm. with ('192.168.74.1', 1730) occurred

Public Keys successfully published

Comm. with ('192.168.74.1', 1731) occurred
Comm. with ('192.168.74.1', 1732) occurred
User Authentication Successful
Comm. with ('192.168.74.1', 1733) occurred

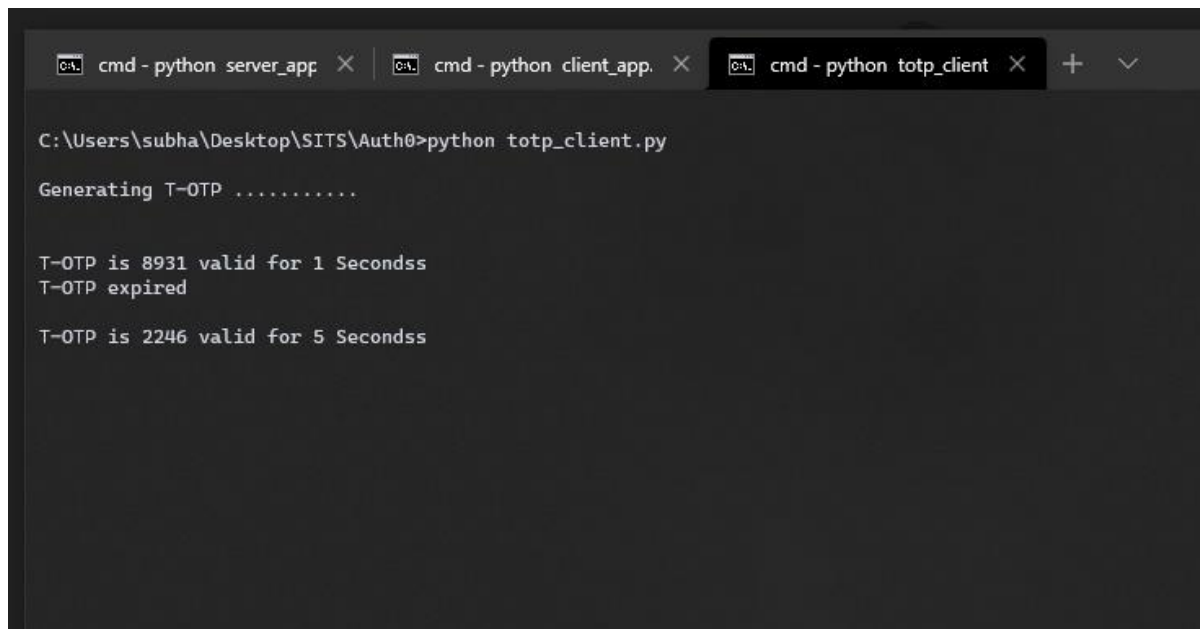
T-OTP Encrypted 51933a407ce

T-OTP Decrypted on the server-side 2922

T-OTP generated on the server-side 2922

Comm. with ('192.168.74.1', 1734) occurred
|
```

Expired TOTP

A terminal window with three tabs: 'cmd - python server_app', 'cmd - python client_app', and 'cmd - python totp_client'. The 'totp_client' tab is active. The output shows the client generating a T-OTP, which is valid for 1 second and then expires. It then generates another T-OTP, which is valid for 5 seconds.

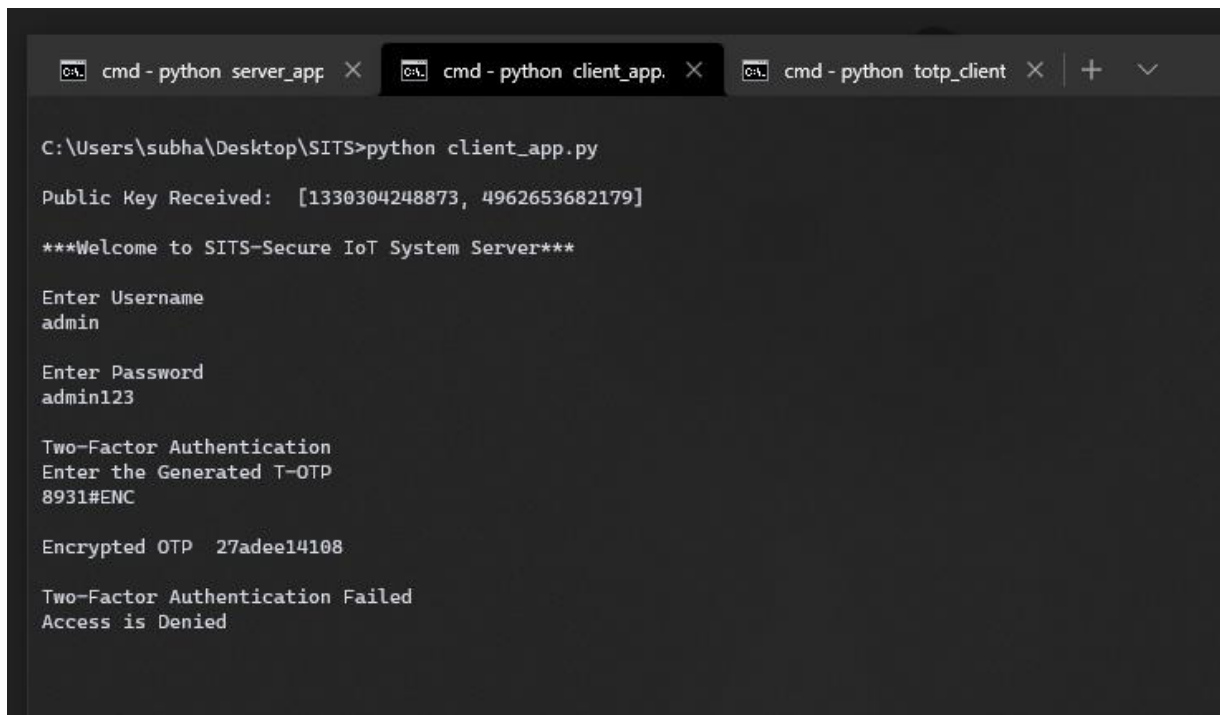
```
C:\Users\subha\Desktop\SITS\Auth0>python totp_client.py

Generating T-OTP .....

T-OTP is 8931 valid for 1 Secondss
T-OTP expired

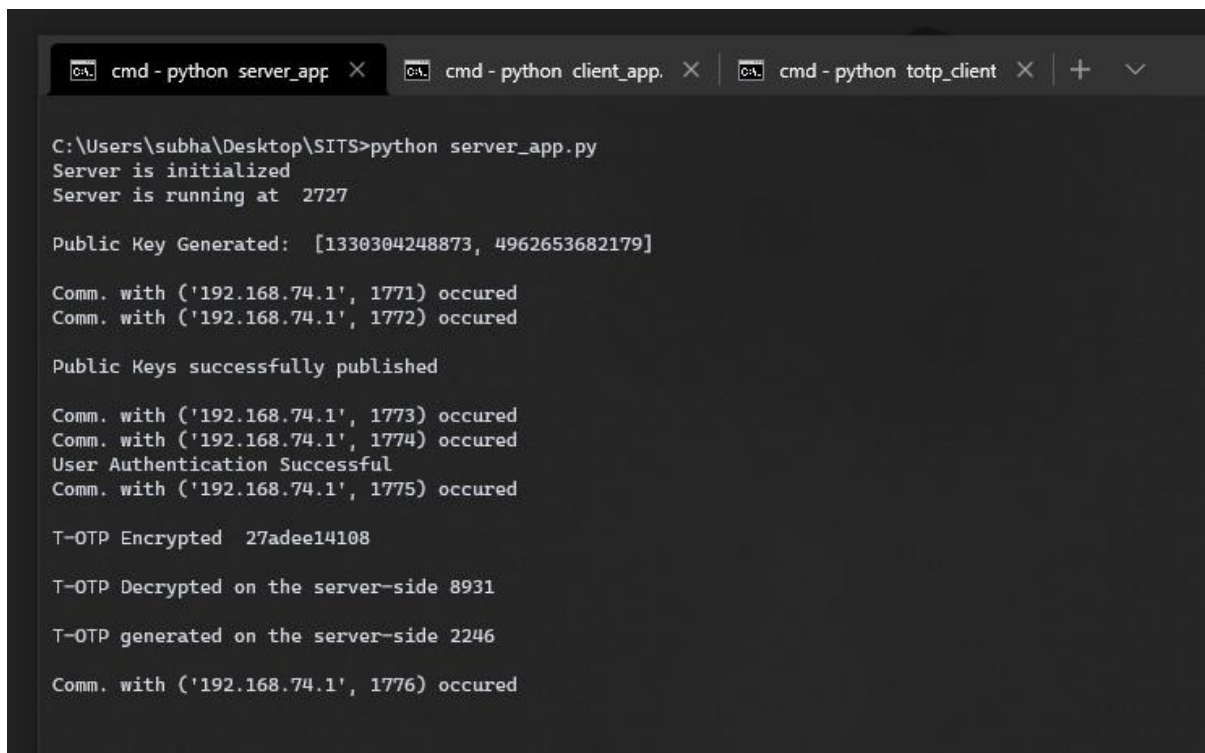
T-OTP is 2246 valid for 5 Secondss
```

Using expired TOTP resulting in failure to login



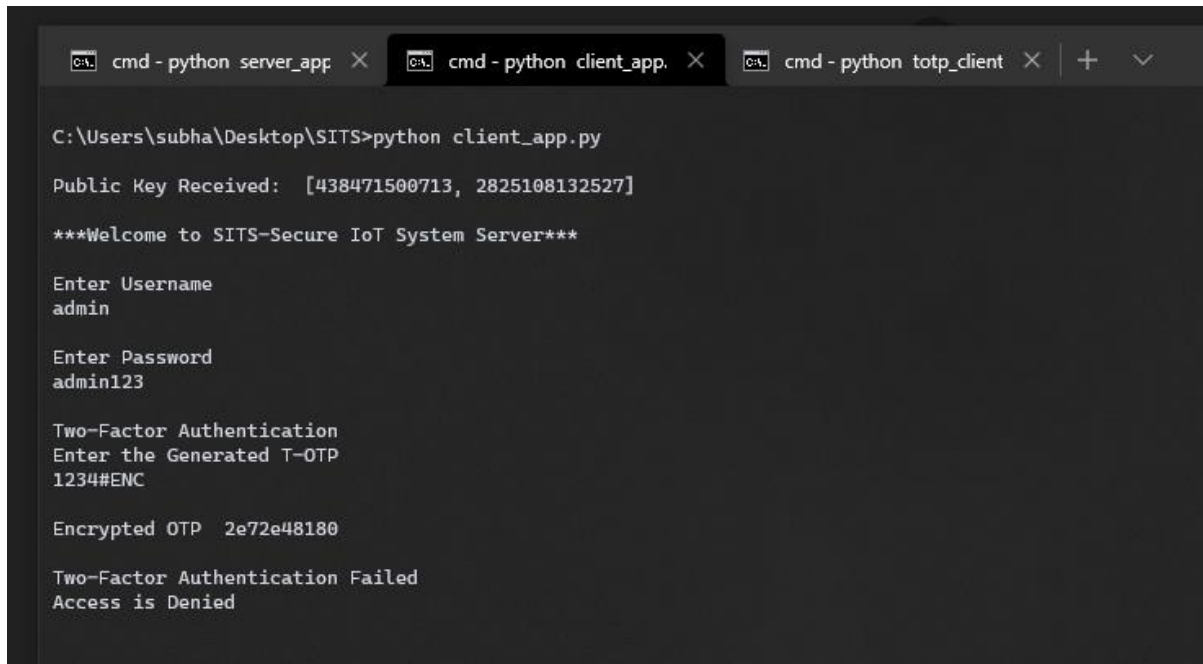
```
cmd - python server_app x cmd - python client_app. x cmd - python totp_client x + v
C:\Users\subha\Desktop\SITS>python client_app.py
Public Key Received: [1330304248873, 4962653682179]
***Welcome to SITS-Secure IoT System Server***
Enter Username
admin
Enter Password
admin123
Two-Factor Authentication
Enter the Generated T-OTP
8931#ENC
Encrypted OTP 27adee14108
Two-Factor Authentication Failed
Access is Denied
```

Server denies access to client using expired TOTP



```
cmd - python server_app x cmd - python client_app. x cmd - python totp_client x + v
C:\Users\subha\Desktop\SITS>python server_app.py
Server is initialized
Server is running at 2727
Public Key Generated: [1330304248873, 4962653682179]
Comm. with ('192.168.74.1', 1771) occurred
Comm. with ('192.168.74.1', 1772) occurred
Public Keys successfully published
Comm. with ('192.168.74.1', 1773) occurred
Comm. with ('192.168.74.1', 1774) occurred
User Authentication Successful
Comm. with ('192.168.74.1', 1775) occurred
T-OTP Encrypted 27adee14108
T-OTP Decrypted on the server-side 8931
T-OTP generated on the server-side 2246
Comm. with ('192.168.74.1', 1776) occurred
```

Submitting invalid TOTP resulting in denial of access



```
C:\Users\subha\Desktop\SITS>python client_app.py

Public Key Received: [438471500713, 2825108132527]

***Welcome to SITS-Secure IoT System Server***

Enter Username
admin

Enter Password
admin123

Two-Factor Authentication
Enter the Generated T-OTP
1234#ENC

Encrypted OTP 2e72e48180

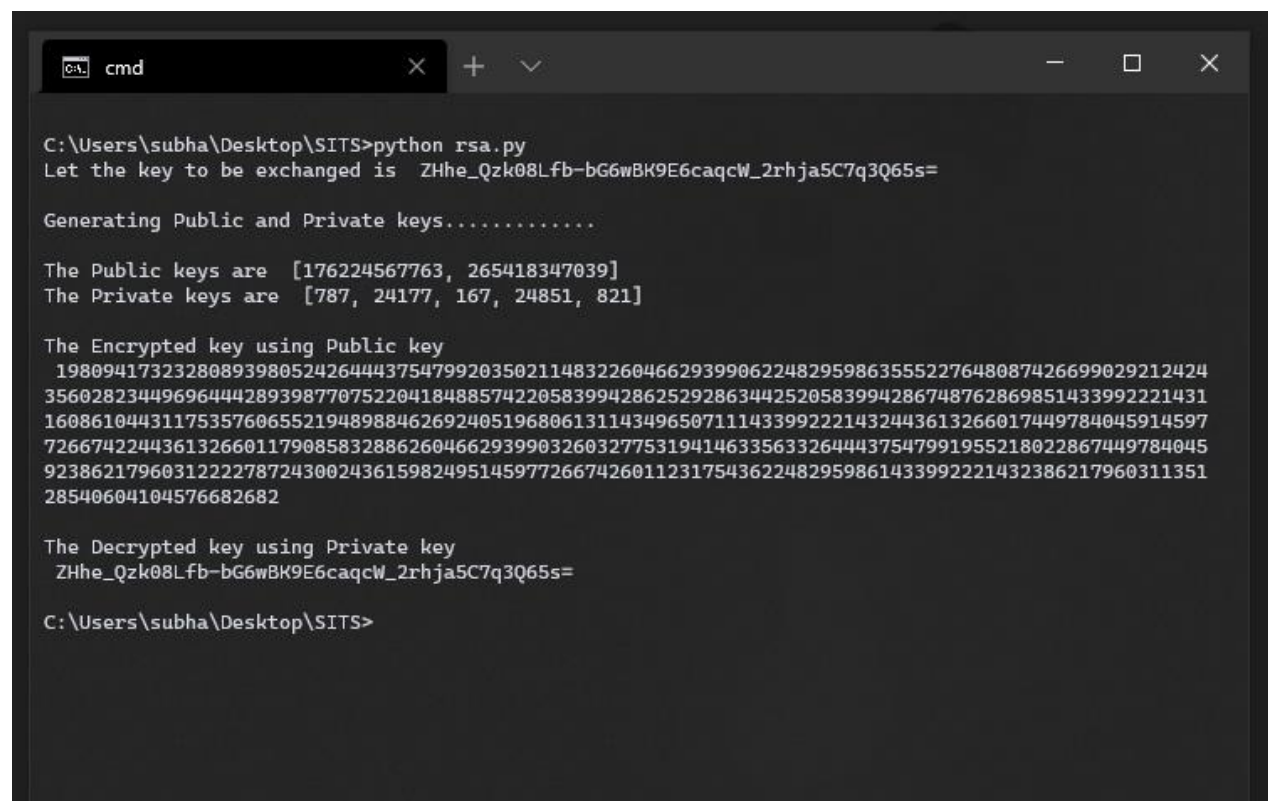
Two-Factor Authentication Failed
Access is Denied
```

Passwords are weak and can be cracked or can be obtained by some or other attacks like phishing attack. One-time-password based systems can increase the security by acting as a two-factor authentication system which validates the OTP user enters with the one the server generated after validating the password. The added advantage of time-based OTP is that they are not dependent on any external infrastructure like SMS or email which requires internet connectivity which might compromise the integrity of the system due to the fault of the service on which the OTP is reliant on like sim spoofing etc. Since T-OTP are based on time stamp and expire after a certain time period, putting time limitation on the intruder in case of any network breach, making it almost impossible for unauthorized access. The time interval can be narrowed down to increase the security of the network.

B. Key Exchange in unsecure channel using Lightweight RSA using Three Prime numbers

This enhanced key exchange security mechanism using RSA increases the speed of key generation, decryption of cipher text and very marginal improvements in encryption of plain text. This algorithm provides the same security mechanism of key exchange but for a low computational cost making it ideal for low powered IoT devices. Even though, being a lightweight algorithm, it doesn't compromise the security in any way allowing for passing keys over unsecured communication channel. With the utilization of three integral primes and Chinese remainder theorem, more speed of key generation and decryption than standard RSA is achieved. Also, another advantage is prevention from attacks involving mathematical factorization, since the addition of third prime adds complexity to the algorithm making to much more difficult to factorize and get back the originally considered primes. Also prevents from various other attacks namely timing attack, modulus attack, plaintext attack, cipher-text attack.

Public key cryptography using Lightweight RSA



```
C:\Users\subha\Desktop\SITS>python rsa.py
Let the key to be exchanged is  ZHhe_Qzk08Lfb-bG6wBK9E6caqcW_2rhja5C7q3Q65s=

Generating Public and Private keys.....

The Public keys are  [176224567763, 265418347039]
The Private keys are  [787, 24177, 167, 24851, 821]

The Encrypted key using Public key
19809417323280893980524264443754799203502114832260466293990622482959863555227648087426699029212424
356028234496964442893987707522041848857422058399428625292863442520583994286748762869851433992221431
160861044311753576065521948988462692405196806131143496507111433992221432443613266017449784045914597
726674224436132660117908583288626046629399032603277531941463356332644437547991955218022867449784045
923862179603122227872430024361598249514597726674260112317543622482959861433992221432386217960311351
28540604104576682682

The Decrypted key using Private key
ZHhe_Qzk08Lfb-bG6wBK9E6caqcW_2rhja5C7q3Q65s=

C:\Users\subha\Desktop\SITS>
```


C. Data encryption SIT: A Lightweight Encryption Algorithm for Secure Internet of Things

This encryption algorithm is well suited for the data security of IoT devices as the proposed algorithm relates to the category of lightweight cryptography; it ensures minimum memory and energy consumption. This algorithm provides the same security level as other encryption algorithms but with low computational cost suited for IoT devices. Even though the algorithm only consists of only five rounds of encryption, it ensures sufficient confusion and diffusion of Shannon's properties for ideal security. SIT algorithm overcomes the problem of weak keys as the algorithm does not use the actual key in the encryption process. Instead, it is first XOR-ed and then passed into an f-function. In the f-function, sufficient confusion and diffusion are created to diminish the bias in selecting a key. It also overcomes the problem of related keys. The given algorithm's key expansion is fast and non-linear diffusion of cipher key difference of that rounds. The presented algorithm can also stand against some well-known attacks known as Interpolation and square attack.

Data encryption using Lightweight SIT algorithm

```
cmd

SIT Algorithm for Data Encryption

Enter a secret key : aezpk1212

Key Generation

Round 1 key is 1110000110110001
Round 2 key is 0110011110110011
Round 3 key is 0111101101111110
Round 4 key is 1011110111001101
Round 5 key is 0100000010110001

Data Encryption
The cipher text of the data is

1ece0338d1b4d1afc2eaf1c5712a71a76348181f95335803f709d3ccd400d78e5e14ccf4a9fb3849cfa4d624f2713bd2d63
f6249ec89a21f56ca35a86ab114064424909314127d77b1e5e33f4233d9c9d9a3d6071a645bbb7a99bf4e74734f5c3231bb
17d199d0d232690db51e324d5c7602cca9f006f21b87602eed35a7153b7900e4f0cb672a8b5da30a8c97e5e4566cf70e195
b6f2f41a7403407db3618762ae2a775222e61c6f08f7fead74bb8e79f9cfc0af4c25934c97733b5c3ee1164572cc92cfcf6d
cef3100bdbc65a10f8f61f62aa0cbf8e42832df76b54fd7303824351e7427566611d9f1e9d25ad04bdfd648aec2169426641
a6c138aa81f529e79f3fba20e125fdc27a2832f25661c46d28649315bfa8e06b4cbc34ee03e799340a318e1072d07cd76ed
29f5468bb35485b409acc9115f7a984ccd1a7f77ad57e173c54cef0921d0b3e640b8ef4f7a5c6b12ac3fa28756a81450b41
c4a3c33d33a117a34bdf0f407b0975daa8587535f18dca66421443b997c8d00396c1ea96c8ec70cafc3775200ce138c420f
18c99d5ef40a3b9b75689c4cec41976c74e079f34404c847d386dfe73feee4ba88413f8daeb87c62ff153b10caacb59f90e
1c3c6c3aa3e3df1b9cf71d4acc5e9b728ca50d058e1798a78910c876fa31b03b3d81835c60cead629524c70a31d435744
80a1aebddf1fabf57d52214fae83111c4eb3ae7788cb0186f9b8f81f24f40667eaff71bf26a52d4e8f9f0b6c34b544e664
f04c0a4a66bb17c146b21205bb6738c6d5580d3d33388d6994a13385c456b0f85ba656c90c4c6ce1199914cc8f6d28c1ad1
258a469c6570878dac5fc68c6d0ed97210e51ea2feb2162aec0c62230710cb955f5068c3766c630562e6d12a4f7eb9e9698
9d45e7ec99a85e91f468b5a5825e5364b60bf047f37fa3b8c16c47cb590e10b8831bb5ceafadf477bde4161f4c6c54e3a15
75334c0f16ed9077134da9f400dd591cf13ca6ba86308233cd830997591fcc51a4fbc9f3f53d2bc0e24d248b5ac8726bc11
0b839a9e03ea58e3a6150d058e1798a789115a08db6a4387f14

Data Decryption
The plain text of data is

Lightweight cryptography performs strong security mechanism, encryption or decryption with low power applications and other functionalities for the pervasive computing. Here we apply the lightweight symmetry key algorithm in order to achieve end-to-end security and with lower power consumption in these devices. These lightweight algorithms are designed for IoT to deal with the security and resource utilization challenges. This is done by implementing various encryption and security algorithms so that the data being transferred between these IoT devices is very secure and is not vulnerable to breaches. The main focus of this paper is to implement these algorithms into one shared platform for all IoT devices from secure authentication to data encryption.
```

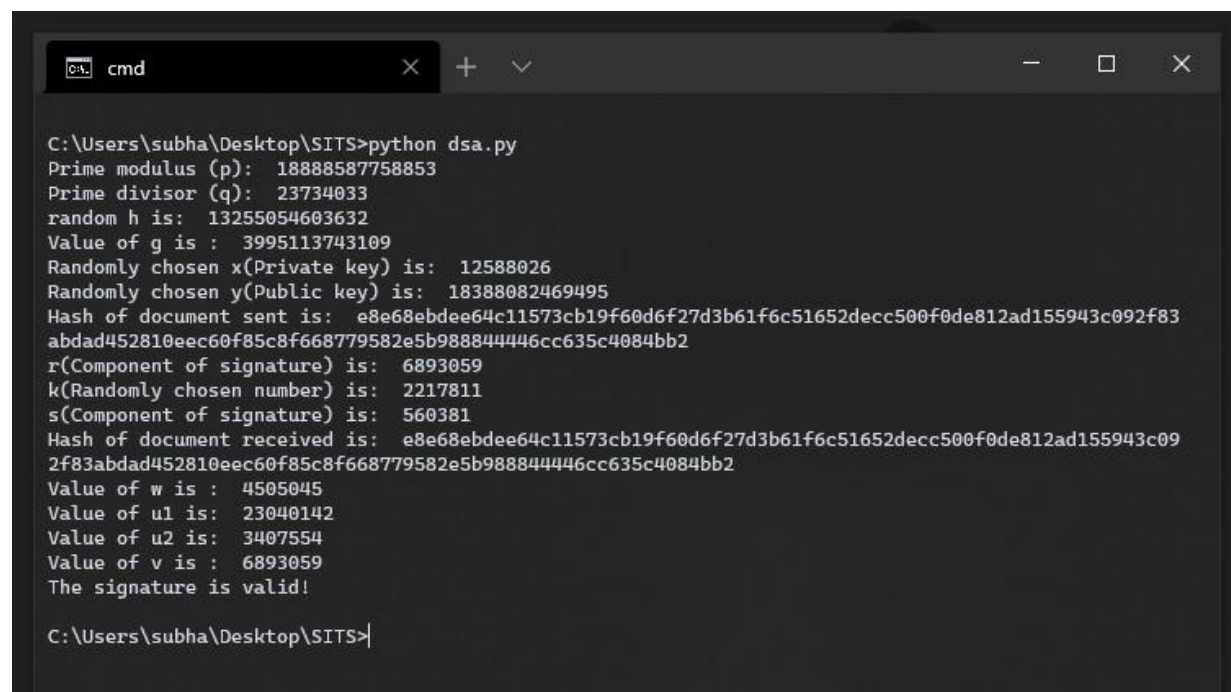
D. Data Integrity verification using Digital Signatures

The length of the signature will be short. Computation speed is fast. It is based on modular exponentiation, along with the discrete logarithm which make it more secure. Takes less space. At every stage we are generating some random number that's increases the level of security.

E. Blake 2b

Fast in both software and hardware. It has a natural instruction parallelism of 4 instructions inside the G function. Padding is minimal. Supports Tree Hashing for verification of large files. Takes lesser RAM than normal blake hash function. Blake2b depends cha-cha. Cha-cha is highly trusted to ever be broken.

Message integrity check using digital signature



```
C:\Users\subha\Desktop\SITS>python dsa.py
Prime modulus (p): 18888587758853
Prime divisor (q): 23734033
random h is: 13255054603632
Value of g is : 3995113743109
Randomly chosen x(Private key) is: 12588026
Randomly chosen y(Public key) is: 18388082469495
Hash of document sent is: e8e68ebdee64c11573cb19f60d6f27d3b61f6c51652decc500f0de812ad155943c092f83
abdad452810eec60f85c8f668779582e5b988844446cc635c4084bb2
r(Component of signature) is: 6893059
k(Randomly chosen number) is: 2217811
s(Component of signature) is: 560381
Hash of document received is: e8e68ebdee64c11573cb19f60d6f27d3b61f6c51652decc500f0de812ad155943c09
2f83abdad452810eec60f85c8f668779582e5b988844446cc635c4084bb2
Value of w is : 4505045
Value of u1 is: 23040142
Value of u2 is: 3407554
Value of v is : 6893059
The signature is valid!

C:\Users\subha\Desktop\SITS>
```


V. Conclusions

The proposed three-module system that implements user authentication using time-based OTP, exchanging keys in unsecure communication channel using lightweight RSA algorithm using three prime numbers, data encryption using SIT algorithm and finally data integrity verification using digital signature algorithm. All of these modules form an end-to-end secured IoT system which ensures the security of the system is never compromised using lightweight cryptographic algorithms. This end-to-end security solution can be adopted to any low power IoT devices in order to securely communicate over the internet and thus the proposed lightweight algorithms are suitable solutions to the current security challenges faced by IoT devices. The proposed system combines multiple lightweight cryptographic solutions into one shared platform for IoT devices from secure authentication to encryption of data for transmission and storage.

VI. References

1. M. L. T. Uymatiao and W. E. S. Yu, "Time-based OTP authentication via secure tunnel (TOAST): A mobile TOTP scheme using TLS seed exchange and encrypted offline keystore," 2014 4th IEEE International Conference on Information Science and Technology, Shenzhen, 2014, pp. 225-229, doi: 10.1109/ICIST.2014.6920371.
2. V. L. Shivraj, M. A. Rajan, M. Singh and P. Balamuralidhar, "One time password authentication scheme based on elliptic curves for Internet of Things (IoT)," 2015 5th National Symposium on Information Technology: Towards New Smart World (NSITNSW), Riyadh, 2015, pp. 1-6, doi: 10.1109/NSITNSW.2015.7176384.
3. Zaid, Mustafa & Hassan, Soukaena. (2018). Lightweight Rsa Algorithm Using Three Prime Numbers. International Journal of Engineering and Technology(UAE). 7. 293-295. 10.14419/ijet.v7i4.36.23790.
4. T. K. Goyal and V. Sahula, "Lightweight security algorithm for low power IoT devices," 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Jaipur, 2016, pp. 1725-1729, doi: 10.1109/ICACCI.2016.7732296.
5. V.G, Kiran. (2015). Design And Implementation Of Tiny Encryption Algorithm. Int. Journal of Engineering Research and Applications. 5. 94-97.
6. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., & Wingers, L. (2015). SIMON and SPECK: Block Ciphers for the Internet of Things. IACR Cryptol. ePrint Arch., 2015, 585.
7. Usman, Muhammad & Khan, Shujaat. (2017). SIT: A Lightweight Encryption Algorithm for Secure Internet of Things. International Journal of Advanced Computer Science and Applications. 8. 10.14569/IJACSA.2017.080151.
8. Baraa Tareq Hammad, Norziana Jamil, Mohd Ezanee Rusli and Muhammad Reza Z`aba "A survey of Lightweight Cryptographic Hash Function"Message authentication using quark and photon,Volume 8, Issue 7, July-2017
9. Jean-Philippe Aumasson ,Samuel Neves, Zooko Wilcox-O'Hearn, Christian Winnerlein "BLAKE2: simpler, smaller, fast as MD5", 2013.01.29
10. Tobias Meuser¹, Larissa Schmidt¹, and Alex Wiesmaier "Lightweight Hash Functions PHOTON & Quark"