RESEARCH ARTICLE OPEN ACCESS

An Overview of Cryptography Requirement in IOT Environment

T. Naga Lakshmi [1], Dr.A. Subramanyam [2]
Assistant Professor [1], Professor [2]
Department of CSE, AITS Rajampet
India

ABSTRACT
Now a days, information sharing has increased exponentially. The information is vulnerable to unauthorized access and interception, while in storage or transmission. Recently, the concept of the Internet of Things (IoT) has drawn considerable attention from both industry and academia. In the IoT, millions of objects with sensors collect data and send the data to servers that analyze, manage and use the data in order to construct some kinds of smart systems. To provide the security for IoT applications, mechanisms like cryptography must be designed to protect communications. This paper gives a brief idea on why cryptography is essential and cryptography primitives in IoT.

Keywords:- IoT, Cryptography -Symmetric and Asymmetric Cryptography, Elliptic curve algorithm

I. INTRODUCTION

Internet of Things (IoT) was first invented in 1998, it is a network of networks where large number of objects or sensors are connected through communications and information infrastructure to provide value-added services. It can be shown in Fig: 1. The Internet of Things (IoT) is the network of "things" embedded with sensors, where connections through the network will enable these objects to collect and exchange data, while each thing is uniquely identifiable through its embedded computing system.

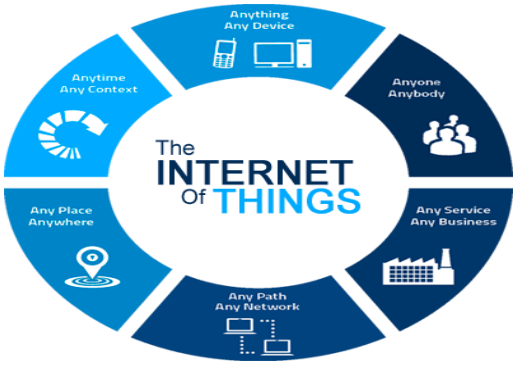

Fig: 1 Internet of Things

The concept of the Internet of Things (IoT) has drawn considerable attention from both industry and academia. In the IoT, millions of objects with sensors collect data and send the data to servers that analyze, manage and use the data in order to construct some kinds of smart systems. To provide the security for IoT applications, mechanisms like cryptography must be designed to protect communications. Cryptography converts data into a format that is unreadable for an unauthorized user. Cryptography can be categorized as Symmetric and Asymmetric.

II. WHY CRYPTOGRAPHY IS ESSENTIAL IN IOT ?

The IoT can find applications in many fields, from the earliest wireless sensor networks such as the smart grid, smart healthcare, smart agriculture, and smart logistics and so on. Current Internet security protocols rely on a well-known and widely trusted suite of cryptographic algorithms: the Advanced Encryption Standard (AES) block cipher for confidentiality; the Rivest-Shamir-Adelman (RSA) asymmetric algorithm for digital signatures and key transport; the Diffie-Hellman (DH) asymmetric key agreement algorithm; and the SHA- 1 and SHA-256 secure hash algorithms. This suite of algorithms is supplemented by a set of emerging asymmetric algorithms, known as Elliptic Curve Cryptography (ECC).

Implementing public-key cryptography for IoT devices requires significant expertise, but organizations can turn to IoT security partners who are capable of providing complete, robust security solutions, typically in a matter of weeks, which is mostly performed in parallel with other development efforts.

Any key can theoretically be broken using a brute-force attack with sufficient computing power. The practical approach of modern cryptography is to use a key of sufficient enough length that it can't be broken without an extraordinary amount of computing power that would be significantly in excess of the value of the contents that the cryptography protects. Modern cryptographic algorithms can make IoT security so affordable.

III. Types of cryptographic Algorithms in the IoT

Cryptographic primitive types fall into the following categories:

- Encryption and decryption
  - Symmetric
  - Asymmetric
- Hashing
- Digital signatures

3.1 Encryption and Decryption

It is used to protect the confidentiality of the information from hackers and only allow it to be deciphered by intended parties. Encryption algorithms can be symmetric or asymmetric. In both cases, a cryptographic key and the unprotected data are given to the encryption algorithm, which cipher encrypts it. Once in this state, it is protected from hackers. The receiving party uses a key to decrypt the data when it is needed. The unprotected data is called plaintext and the protected data is called **cipher text**. The basic encryption process is depicted in the following fig:2

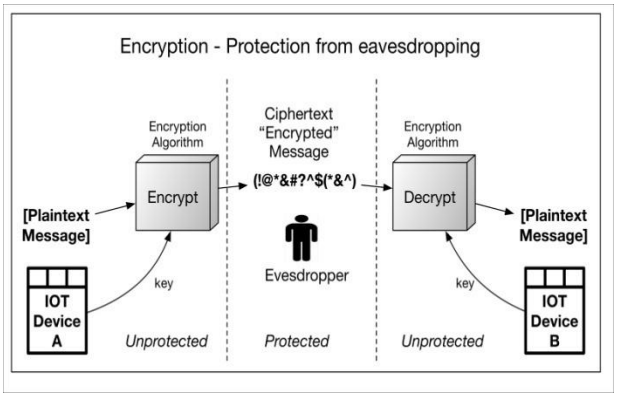

Fig: 2. Encryption and Decryption

In today's Internet threat environment as many encrypted protocols operate only on a point-to-point basis and must traverse a variety of gateways and other intermediate devices, the paths to which may be highly insecure.

3.1.1 Symmetric Encryption

Symmetric encryption simply means the sender and the receiver use an identical cryptographic key. The algorithm, which is able to both encrypt and decrypt depending on the mode is a reversible operation, as shown in the following diagram:

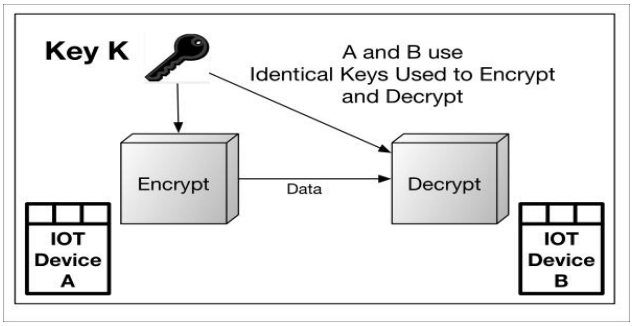

Fig: 3.Symmetric Encryption

In many protocols, a different symmetric key is used for each direction of travel. So, for example, Device A may encrypt to Device B using key X. Both parties have key X. The opposite direction (B to A) may use key Y which is also in the possession of both parties. This type of encryption leads to insecure for data transmission.

3.1.2 Asymmetric Encryption

Asymmetric encryption simply means there are two different, pair wise keys, one public and the other private, used to encrypt and decrypt, respectively.

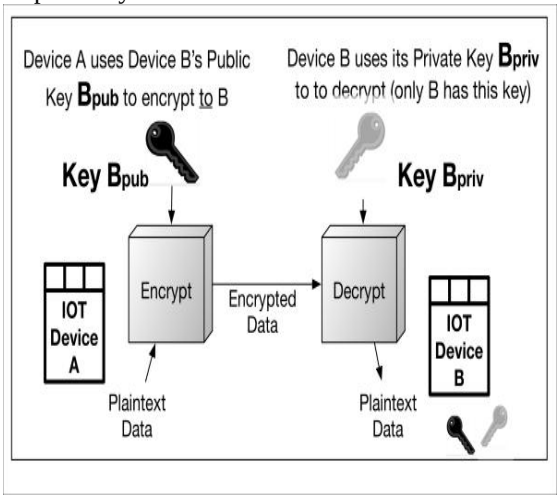

Fig: 4. Asymmetric-Encryption

In the following diagram, IoT device A uses IoT device B's public key to encrypt to device B. Conversely, device B uses device A's public key to encrypt information to device A. Each device's private keys are kept secret, otherwise anyone or anything possessing them will be able to decrypt and view the information.

The only asymmetric encryption algorithm in use today is that of **RSA** (**Rivest, Shamir, Adelman**), an **integer factorization cryptography** (**IFC**) algorithm that is practical for encrypting and decrypting small amounts of data. The advantage of this encryption technique is that only one party possessing the pair wise RSA private key can decrypt the traffic. Typically, private key material is not shared with more than one entity. The disadvantage of asymmetric encryption (RSA), as stated earlier, is the fact that it is limited to encrypting up to the modulus size in question (1024 bits, 2048 bits, and so on). Given this disadvantage, the most common use of RSA public key encryption is to encrypt and transport other small keys frequently symmetric or random values used as precursors to cryptographic keys.

3.2 Hashing

Cryptographic hashing are used in a variety of security functions for their ability to represent an arbitrarily large message with a small sized, unique thumbprint (the hash). They have the following properties:

- They are designed not to disclose any information about the original data that was hashed (this is called resistance to first pre-image attacks)
- They are designed to not allow two different messages to have the same hash (this is called resistance to second pre-image attacks and collisions)
- They produce a very random-looking value (the hash).

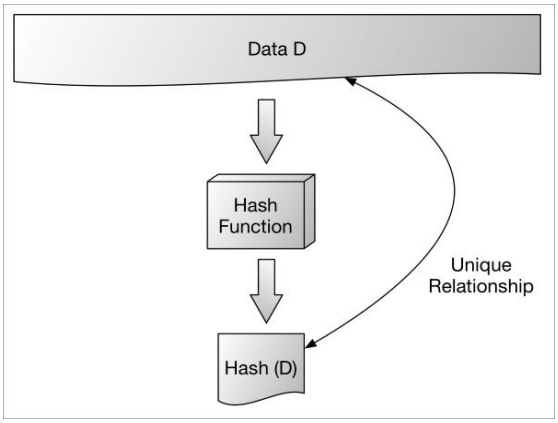

Fig: 5. Hash-functions

The fig: 5 denotes an arbitrary chunk of data D being hashed into H(D). H(D) is a small, fixed size (depending on the algorithm in use); from it, one cannot (or should not be able to) discern what the original data D was.

3.3 Digital signatures

A **digital signature** is a cryptographic function that provides integrity, authentication, data origin, and in some cases, non-repudiation protections. Just like a hand-written signature, they are designed to be unique to the *signer*, the individual or device responsible for signing the message and who possesses the signing key. Digital signatures come in two flavours, representing the type of cryptography in use: symmetric (secret, shared key) or asymmetric (private key is unshared). It can be shown as fig:6.

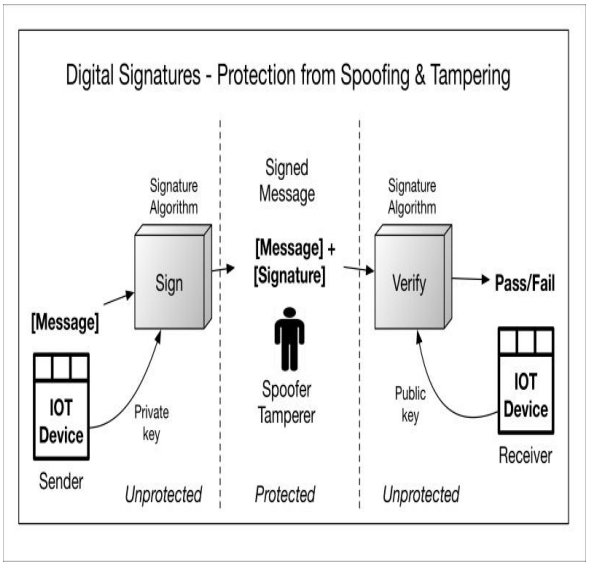

Fig: 6 Digital Signature

Cryptographic algorithms used in IoT are:

- The Advanced Encryption Standard (AES) block cipher for confidentiality.
- The Rivest-Shamir-Adelman (RSA) asymmetric algorithm for digital signatures and key transport.
- the Diffie-Hellman (DH) asymmetric key agreement algorithm.
- SHA- 1 and SHA-256 secure hash algorithms

This suite of algorithms is supplemented by a set of emerging asymmetric algorithms, known as Elliptic Curve Cryptography (ECC).

3.4 ECDSA - Elliptic Curve Digital Signature Algorithm

ECDSA is a variant of the Digital Signature Algorithm (DSA) that operates on elliptic curve groups. For sending a signed message from A to B, both have to agree up on Elliptic Curve domain parameters.

Elliptic Curve Domain parameters:

Apart from the curve parameters a and b, there are other parameters that must be agreed by both parties involved in secured and trusted communication using ECC. These are domain parameters.

The domain parameters for prime fields and binary fields are described below. The generation of domain parameters is out of scope of this paper. There are several standard domain parameters defined. Generally the protocols implementing the ECC specify the domain parameters to be used.

Domain parameters for EC over field Fp:

The domain parameters for Elliptic curve over Fp are p, a, b, G, n and h. p is the prime number defined for finite field Fp . a and b are the parameters defining the curve y2 mod p= x3 + ax + b mod p. G is the generator point (xG, yG), a point on the elliptic curve chosen for cryptographic operations. n is the order of the elliptic curve. The scalar for point multiplication is chosen as a number between 0 and n – 1. h is the cofactor where h = #E(Fp)/n. #E(Fp) is the number of points on an elliptic curve. 9.2. Domain parameters for EC over field F2 m The domain parameters for elliptic curve over F2 m are m, f(x), a, b, G, n and h. m is an integer defined for finite field F2 m . The elements of the finite field F2 m are integers of length at most m bits. f(x) is the irreducible polynomial of degree m used for elliptic curve operations. a and b are the parameters defining the curve

$$y2 + xy = x3 + ax2 + b.$$

G is the generator point (xG, yG), a point on the elliptic curve chosen for cryptographic operations. n is the order of the elliptic curve. The scalar for point multiplication is chosen as a number between 0 and n – 1. h is the cofactor where h = #E(F2 m)/n. #E(F2 m) is the number of points on an elliptic curve.. Sender 'A' have a key pair consisting of a private key dA (a randomly selected integer less than n, where n is the order of the curve, an elliptic curve domain parameter) and a public key QA = dA * G (G is the generator point, an elliptic curve domain parameter). An overview of ECDSA process is defined below.:

Signature Generation:

For signing a message m by sender A, using A's private key dA

- Calculate e = HASH (m), where HASH is a cryptographic hash function, such as SHA-1
- Select a random integer k from [1,n − 1]
- Calculate r = x1 (mod n), where (x1, y1) = k * G. If r = 0, go to step 2
- Calculate s = k − 1(e + dAr)(mod n). If s = 0, go to step 2
- The signature is the pair (r, s)

Signature Verification:

For B to authenticate A's signature, B must have A's public key QA

- Verify that r and s are integers in [1,n − 1]. If not, the signature is invalid

- Calculate e = HASH (m), where HASH is the same function used in the signature generation
- Calculate w = s −1 (mod n)
- Calculate u1 = ew (mod n) and u2 = rw (mod n)
- Calculate (x1, y1) = u1G + u2QA 6. The signature is valid if x1 = r(mod n), invalid otherwise

As we discussed earlier the point multiplication is the main operation in elliptic curve cryptography. Point multiplication involves plenty of point addition and point doubling. Each point addition and doubling involves a multiplicative inverse operation.

Finding multiplicative inverse is a costly operation in both finite fields, Fp and F2 m. For using the projective coordinate in elliptic curve one has to convert the given point in affine coordinate to projective coordinate before point multiplication then convert it back to affine coordinate after point multiplication. The entire process requires only one multiplicative inverse operation. The operation in projective coordinate involves more scalar multiplication than in affine coordinate. ECC on projective coordinate will be efficient only when the implementation of scalar multiplication is much faster than multiplicative inverse operation.

3.5 ECDH – Elliptic Curve Diffie Hellman:

ECDH is a key agreement protocol that allows two parties to establish a shared secret key that can be used for private key algorithms. Both parties exchange some public information to each other. Using this public data and their own private data these parties calculates the shared secret. Any third party, who doesn't have access to the private details of each device, will not be able to calculate the shared secret from the availabl information. An overview of ECDH p defined below.

For generating a shared secret betwe B using ECDH, both have to agree up o Curve domain parameters. The domain p are defined in section 3.4. Both end have consisting of a private key d (a randomly integer less than n, where n is the ord curve, an elliptic curve domain paramet public key Q = d * G (G is the generator elliptic curve domain parameter).

Let (dA, QA) be the private key - public key pair of A and (dB, QB) be the private key - public key pair of B.

- The end A computes K = (xK, yK) = dA * QB
- The end B computes L = (xL, yL) = dB * QA
- Since dAQB = dAdBG = dBdAG = dBQA. Therefore K = L and hence xK = xL
- Hence the shared secret is xK.

Since it is practically impossible to find the private key dA or dB from the public key K or L, its not possible to obtain the shared secret for a third party.

IV. CONCLUSION

Finally in this paper, concluded that cryptography plays a major role for transmission of data in IoT environment. The cryptographic algorithms AES, RSA,DH, SHA-1, SHA-256 are using but now a days using asymmetric algorithm like Elliptic Curve Cryptography with modifications in the algorithm is implemented. Some type of cryptographic algorithms may be needed in cognitive systems.

V. FUTURE DIRECTIONS OF CRYPTOGRAPHY IN IOT

The cryptography used in the IoT today comprises the same cryptographic trust mechanisms used in the broader Internet. Like the Internet, however, the IoT is scaling to unprecedented levels that require far more distributed and decentralized trust mechanisms.

Digital conversion of brain-sensed signals allows the cognition-ready data to be conveyed over data buses, IP networks, and yes, even the Internet. In terms of the IoT, this type of cognitive research implies a future in which some types of smart devices will be smart because there is a human or other type of brain controlling or receiving signals from it. Now imagine what type of IoT security may be needed in such cognitive systems where the things are human brains and dynamic physical systems.

VI. REFERENCES

[1] K. Ashton, That ―Internet of Things‖ Thing, RFiD Journal. (2009).

[2] L. Atzori, A. Iera, G. Morabito, The Internet of Things: A survey, Computer Networks 54 (2010) 2787–2805.

[3] RFC 5091 Identity-Based Cryptography Standard (IBCS) #1:Supersingular Curve Implementations of the BF and BB1 Cryptosystems

[4] RFC 6090 Fundamental Elliptic Curve Cryptography Algorithms [14]Lightweight Cryptography for the Internet of Things, Masanobu Katagi and Shiho Moriai,Sony Corporation

[5] Cryptographic hash Algorithm Competition.http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/index.html

[6] The XTR public key system. Lenstra and Verheul. Crypto 2000

[7] Certificateless Public Key Cryptography Sattam S. Al-Riyami and Kenneth G. Patersony

[8] Darrel Hankerson, Julio Lopez Hernandez, Alfred Menezes, Software Implementation of Elliptic Curve Cryptography over Binary Fields, 2000, Available at http://citeseer. ist.psu.edu/hankerson00software.html