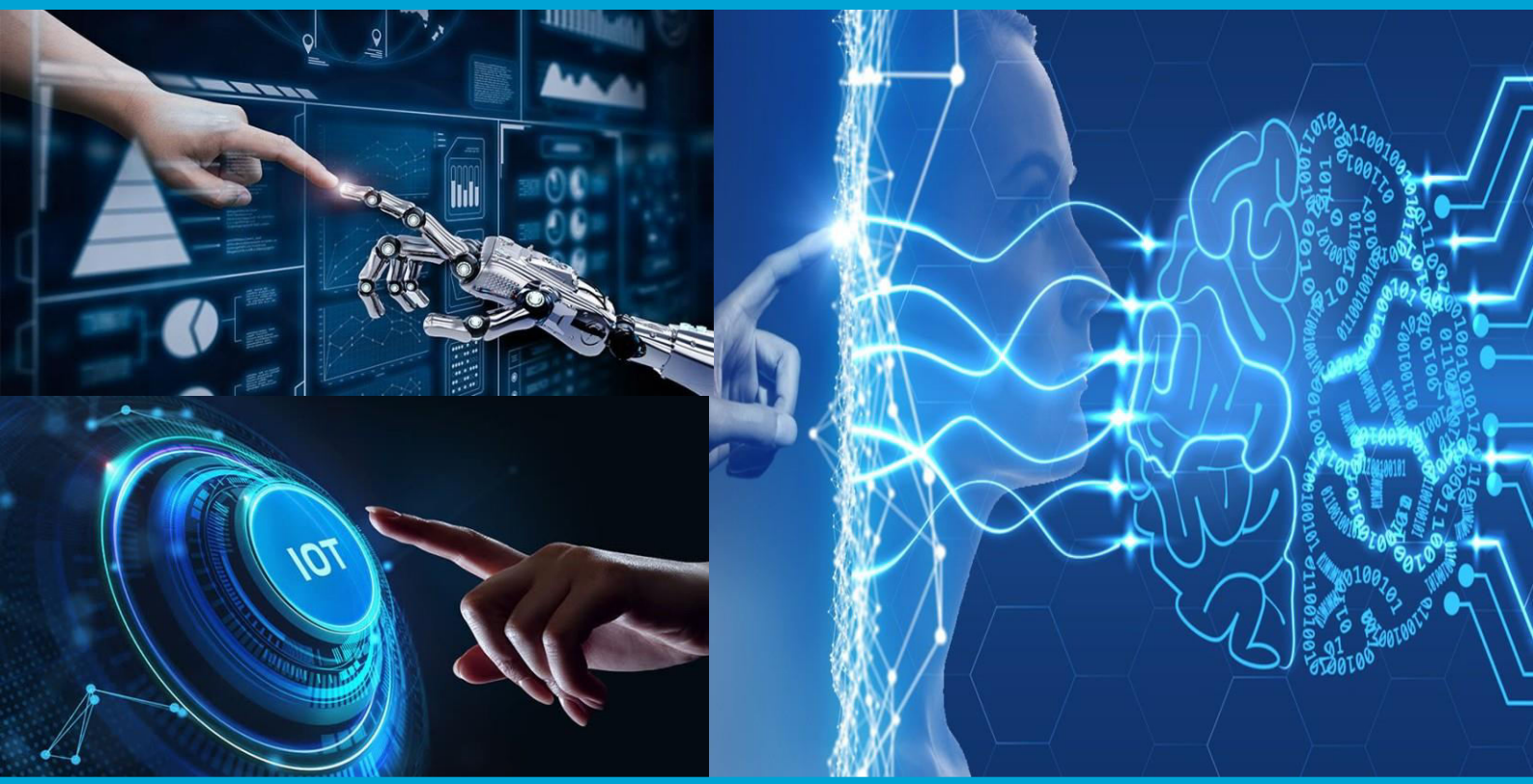
IJMRSETM

e-ISSN: 2395 - 7639



INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH

IN SCIENCE, ENGINEERING, TECHNOLOGY AND MANAGEMENT

Volume 8, Issue 6, June 2021

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.580

| ISSN: 2395-7639| www.ijmrset.com| Impact Factor: 7.580 | A Monthly Double-Blind Peer Reviewed Journal |

|| Volume 8, Issue 6, June 2021 ||

|DOI:10.15680/IJMRSETM.2021.0806011|

Cryptography as A Tool To Keep IOT Devices Secure

Dr. Rajendra Kumar Bharti

Associate Professor, Computer Science & Engineering, Bipin Tripathi Kumaon Institute of Technology, Dwarahat, Uttarakhand, India

ABSTRACT : - Cryptography is a way of communication that is secure. The prefix "crypt" means "hidden" and the suffix "graphy" means "writing". In cryptography, plain text is converted to encrypted text before it is sent, and it is converted to plain text after communication on the other side. The algorithms are utilized to create cryptographic keys, for digital signatures, for verification to secure the confidentiality of information, to browse the Internet and to ensure confidential transactions such as credit and debit card transactions. The Internet of Things (IoT) is a vastly growing area in computer science as it helps exchange data by interconnecting devices over the internet, therefore, it should be secured. IoT's are making smart devices smarter and of higher quality to enhance the user experience. IoT's security challenges are more vulnerable because the devices are openly accessible to all in the network. In this paper, cryptographic methods are proposed to secure the IoT devices i.e. four of the most used encryption algorithms namely: AES (Rijndael), DES, Triple DES and Blowfish

KEYWORDS: cryptography, IoT, secure, tool, digital, vulnerable, algorithms, keys, smarter, blowfish, network

I. INTRODUCTION

The rapidly growing reliance on IoT applications in industrial, medical, transportation, and other critical applications has dramatically altered the security landscape. Unlike earlier enterprise applications with readily available resources for processing security algorithms, enterprise-level IoT applications find themselves vulnerable to a growing array of threats that target expanding networks of resource-constrained IoT devices. In their rush to respond to rapidly emerging IoT opportunities, organizations too often deploy IoT devices functionally incapable of supporting basic security measures for protecting stored data and protecting the exchange of data and commands across vulnerable networks.[1]

How far developers need to go to secure a design of course depends on multiple factors. Each application faces its own set of threats and requires an appropriate assessment of risks from those threats. Because connected devices face an uncommon number of threats, any IoT device will require at least some minimal security measures.

To some, implementing robust security measures for a simple IoT device might seem like a case of overengineering, but even simple temperature sensor devices that lack sufficient protection can provide hackers with a point of entry into corporate networks [1]. Indeed, IoT security faces constant challenges due to the combination of pervasive connectivity provided by IoT applications and the resource limited devices underlying these applications. In fact, even in IoT device designs with sufficient resources to run cryptography algorithms in software, applications can remain vulnerable due to subtle errors in implementation of those algorithms.[2]

This article describes the basic classes of cryptography algorithms and explores their role in security. It then shows how developers can take advantage of processors and specialized devices—from Maxim Integrated, Microchip Technology, and Texas Instruments—that are designed to accelerate these algorithms, enhancing different aspects of security while simplifying its implementation.

The various types of cryptographic algorithms and their roles

Cryptography algorithms fall into three broad categories that address fundamental security principles of confidentiality, authentication (verify the source of the message), non-repudiation (prove that the sender created an encrypted or signed message), and integrity:

| ISSN: 2395-7639| www.ijmrset.com| Impact Factor: 7.580 | A Monthly Double-Blind Peer Reviewed Journal |

|| Volume 8, Issue 6, June 2021 ||

|DOI:10.15680/IJMRSETM.2021.0806011|

- Symmetric key algorithms, where the algorithm or cipher, uses the same secret key to encrypt a human-readable (plaintext) message to a protected version (ciphertext), and later to decrypt the ciphertext back to plaintext. Symmetric key ciphers are typically used to ensure confidentiality. Common symmetric cryptography algorithms include:
    - Triple DES (Data Encryption Standard), also known as 3DES, or officially by the U.S. National Institute of Standards and Technology (NIST) as the Triple Data Encryption Algorithm (TDEA).
    - Advanced Encryption Standard (AES)[2] algorithms such as AES-256, which uses 256-bit keys.
- Asymmetric key algorithms, where the cipher uses a paired set of private and public keys to encrypt and decrypt a message, typically as part of more extensive security protocols for key agreement and digital signatures. Asymmetric ciphers are typically used to ensure confidentiality, authentication, or non-repudiation. Public key cryptography algorithms include:
    - Algorithms using finite field cryptography (FFC), including:
        - Federal Information Processing Standard (FIPS) Digital Signature Algorithm (DSA)
        - Internet Engineering Task Force (IETF) Diffie-Hellman (DH)[3] key exchange
    - Algorithms using integer factorization cryptography (IFC), including:
        - RSA (Rivest–Shamir–Adleman)[4] algorithm
    - Algorithms using elliptic-curve cryptography (ECC), including:
        - Elliptic Curve Diffie-Hellman (ECDH)[5] key exchange
        - Elliptic Curve Digital Signature Algorithm (ECDSA)[6]
- Hash algorithms, where the algorithm reduces an original message to a much smaller, unique fixed length value, variously called a hash, digest, or signature. These one-way conversion functions play a critical role in verifying that the message has not been altered (integrity) and find application in multiple protocols involving a message authentication code (MAC), keyed-hash message authentication code (HMAC), or key derivation functions (KDF), among others. Cryptographic hash algorithms include:
    - MD5 (message digest 5)
    - Secure Hash Algorithms (SHA)[7] such as SHA-256, which generates a 256-bit hash value for a message.[3]

As with any effective crypto algorithm, the algorithms mentioned above are designed with multiple critical requirements that lie well beyond the scope of this brief article. From a broad perspective, however, key-based algorithms need to generate ciphertext that is virtually impossible (or at least economically unfeasible) to decrypt without the key. For their part, hash algorithms need to generate hashes quickly, producing the same hash for the same input message but generating a markedly different hash for even a slight change in the input message, while never producing the same hash value for two different messages, and never producing the original message given a particular hash value.

II. DISCUSSION

The object lesson of this admittedly superficial look at these algorithms is that cryptographic algorithms rely on a sequence of mathematical operations designed to make attempts to compromise the results so computationally expensive that it is impossible—or impractical—to complete them fast enough to be useful to the perpetrator. In addition, even a cursory inspection of each algorithm suggests that a resource-constrained IoT device will likely have little chance of executing a software implementation of the algorithm without potentially compromising the device's primary functional requirements. Finally, the precise details of the algorithm beyond the steps shown here mean that even a subtle coding error or minor misinterpretation of the standard can result in security vulnerabilities, or even outright failure of the cryptographic process.[4]

Errors in algorithm implementations can occur even in the largest development organizations, and in applications that critically depend on those algorithms. For example, a well-known vulnerability in a game console arose because the company-built implementation of ECDSA used a constant value of k, rather than a random value for the type of calculation shown in Equation 3. As a result, hackers could derive the secret key d. Use of a faulty random generator for creating k led to a similar exploit resulting in a major Bitcoin loss.

Hardware-based cryptography capabilities built into processors and dedicated security ICs let developers largely ignore the complex details of cryptography algorithm execution and focus instead on the benefits of using those capabilities to secure their applications. These devices add an extra layer of security by integrating the data flows and operations

| ISSN: 2395-7639| www.ijmrset.com| Impact Factor: 7.580 | A Monthly Double-Blind Peer Reviewed Journal |

|| Volume 8, Issue 6, June 2021 ||

|DOI:10.15680/IJMRSETM.2021.0806011|

within the device, removing a common form of attack that monitors external buses for signs of privileged information. Besides providing a trusted implementation of a particular algorithm, a hardware-based solution allows developers to build security into designs without compromising fundamental requirements such as response latency and overall performance.[5]

Crypto accelerators built into these devices offload cryptography execution from the main processor, freeing it to handle the design's primary function. In fact, hardware-based crypto support has become an increasingly common feature of processors. At the same time, not every application requires the full force of security measures supported by the algorithms described above. Indeed, developers can find a broad range of accelerated crypto algorithms and combinations of algorithms in processors such as the following:

- Maxim Integrated's MAX32631 32-bit microcontroller, which supports AES and DSA ciphers
- Maxim Integrated's MAX32520 32-bit MCU, which supports AES, SHA, and ECDSA algorithms
- Microchip Technology's PIC 24F XLP 16-bit microcontroller family members, where devices such as the PIC24FJ256GA406 support AES and 3DES ciphers
- Microchip Technology's 32-bit PIC 32MZ MCU and 32-bit SAM9X60 MPU family members, where devices such as the PIC32MZ2048EFM144 and SAM9X60T support AES and 3DES ciphers as well as SHA and HMAC hash functions
- Texas Instruments' SimpleLink MCU family members such as the CC1312R and CC2652R wireless MCUs, which support AES, ECDH, and ECDSA ciphers as well as SHA hash functions[6]

Other security devices such as the Maxim Integrated DS28E38 and Microchip Technology ATECC608A integrate crypto accelerators and related functionality needed to speed authentication protocols. Among their broad crypto capabilities, these devices support the kind of ECDSA operation described earlier. In an IoT device or smart peripheral, a host processor can use authentication ICs such as these to quickly create ECDSA P-256 digital signatures to send to another device or authenticate the ECDSA P-256 signatures from other devices.

Security-enabled processors and specialized devices are typically built with a broad hardware-based security framework that provides additional security functionality such as high quality random number generators. Many devices that provide this level of capability take advantage of random noise sources inherent in semiconductor designs to maximize the entropy needed in true random number generators (TRNGs). As suggested in the Bitcoin example mentioned earlier, these kinds TRNGs are an essential factor in the proper operation of cryptographic algorithms.[7]

Integrated support for secure storage of private keys and other secret data provide one of the most important capabilities of secure designs. Other architectural features available with these and similar processors provide an even deeper level of security support.

For all their capabilities, processors with integrated crypto accelerators and related features help simplify development of secure designs through use of straightforward application programming interface (API) libraries. Intuitive API function calls let developers abstract security implementations, relying on the API to access the underlying hardware functionality. [8]

III. RESULTS

Authentication is ensured by establishing the identity of the server (and optionally the client) by verification of security certificates, which contain the respective public key for each participant. Here, each participant sends a message encrypted with its private key. Because the received public key can only decrypt a message encrypted with its associated private key, each participant can confirm that the provider of a certificate actually owns that certificate.

In the next TLS stage, the participants execute a series of transactions to create a shared session key. This shared session key is then used to encrypt the actual message traffic, ensuring the confidentiality of message exchanges for that session.[9]

This move to higher security strength continues to drive the evolution in ciphers and recommended cipher suites. For example, the U.S. National Security Agency (NSA) Commercial National Security Algorithm (CNSA) Suite displaced

| ISSN: 2395-7639| www.ijmrset.com| Impact Factor: 7.580 | A Monthly Double-Blind Peer Reviewed Journal |

|| Volume 8, Issue 6, June 2021 ||

|DOI:10.15680/IJMRSETM.2021.0806011|

the earlier NSA Suite B with recommendations for use of more robust parameters needed to protect information classified as top secret (Table 1).

Table 1: The NSA recommended CNSA Suite includes cryptography algorithms with recommendations for minimum levels of security strength required to protect highly sensitive information. (Image source: Digi-Key, from NSA data)

Algorithm	Function	Specification	Parameters
Advanced Encryption Standard (AES)	Confidentiality	FIPS Pub 197	Use 256 bit keys to protect up to TOP SECRET
Secure Hash Algorithm (SHA)	Integrity	FIPS Pub 180-4	Use SHA-384 to protect up to TOP SECRET.
Diffie-Hellman (DH) Key Exchange	Key establishment	IETF RFC 3526	Minimum 3072 bit modulus to protect up to TOP SECRET
Elliptic Curve Diffie-Hellman (ECDH) Key Exchange	Key establishment	NIST SP 800-56A	Use Curve P-384 to protect up to TOP SECRET.
Elliptic Curve Digital Signature Algorithm (ECDSA)	Digital signatures	FIPS Pub 186-4	Use Curve P-384 to protect up to TOP SECRET.
RSA	Key establishment	NIST SP 800-56B rev 1	Minimum 3072 bit modulus to protect up to TOP SECRET
RSA	Digital signatures	FIPS PUB 186-4	Minimum 3072 bit-modulus to protect up to TOP SECRET.

Multiple protocol options allow developers to refine this generic TLS session creation process, sometimes to the detriment of overall security. In addition, during the parameter exchange process, developers can use a different cipher suite by selecting a suitable combination of TLS 1.2 supported algorithms for each phase of the protocol, including:

- Key establishment: RSA, DH, ECDH
- Authentication: RSA, DSA, ECDSA
- Ciphers: 3DES, AES
- Message authentication: SHA

In cryptography, algorithm security strength is defined in terms of x bits and the expectation that an attack would require about 2^x operations to derive the private key underlying the algorithm. Defined in these terms, the different classes of algorithms can require significantly different key lengths to achieve comparable levels of security

IV. CONCLUSIONS

IoT devices and other connected designs face a growing number of threats that require increasingly robust security methods based on a wide range of cryptography algorithms. Designed to make cracking security an impractical proposition, these algorithms rely on a series of transformations and mathematical operations to encrypt plaintext to ciphertext and decrypt ciphertext back to plaintext. As shown, it's possible to use hardware-based implementations of these algorithms to let developers more easily build strong security into a design without compromising its primary requirements for functionality and performance.[9]

| ISSN: 2395-7639| www.ijmrset.com| Impact Factor: 7.580 | A Monthly Double-Blind Peer Reviewed Journal |

|| Volume 8, Issue 6, June 2021 ||

|DOI:10.15680/IJMRSETM.2021.0806011|

REFERENCES

1. How a fish tank helped hack a casino
2. FIPS PUB 197: the official AES standard
3. IETF RFC 3526 (DH)
4. NIST SP 800-56B Rev. 1 (DOI) (RSA)
5. NIST SP 800-56A (ECDH)
6. FIPS Pub 186-4 (digital signature)
7. FIPS PUB 180-4: Secure Hash Standard (SHS)
8. IETF RFC 5246 (TLS 1.2)
9. IETF RFC 8446 (TLS 1.3)

Impact Factor: 7.580

INTERNATIONAL JOURNAL
OF MULTIDISCIPLINARY RESEARCH
IN SCIENCE, ENGINEERING, TECHNOLOGY AND MANAGEMENT

+91 99405 72462 +91 63819 07438 ijmrsetm@gmail.com