Design of Security Chips in the IoT Ecosystem

Jiayi Zhang*

School of Electronic and Electrical Engineering, University of Leeds, Leeds, the United Kindom

*Corresponding author: el23j2z@leeds.ac.uk

Abstract. With the rapid growth of IoT devices, security issues are becoming increasingly prominent. The main research object of this article is the Internet of Things security chip. In response to the difficulties faced by the current Internet of Things security and the importance of Internet of Things security, this article mainly introduces the basic design principles of Internet of Things security chips, and provides a detailed introduction to the structures and applications of three classic models: TEE, HSM, and TPM. In addition, comparative analysis was conducted on these models. IoT security is a highly challenging field that requires comprehensive consideration of security issues at different levels and stages. The IoT security chips play a crucial role in ensuring the security and data privacy of IoT devices. In the future, IoT security chips may be combined with emerging technologies such as blockchain, artificial intelligence, and quantum encryption to enhance security and credibility.

Keywords: Security chips in the IoT, TEE, HSM, TPM.

1. Introduction

The development of IoT security chips can be traced back to the beginning of computer security in the last century. In its development process, a series of important scientists have laid the foundation for the development of IoT security. Among them, Whitfield Diffie and Martin Hellman proposed the famous "public key cryptosystem" in 1977, which laid the foundation of modern encryption technology and provided important support for later Internet of Things security [1]. By introducing the concepts of public and private keys, the RSA encryption algorithm proposed by Rivest, Shamir, and Adleman in 1978 provided a secure means for IoT communication [2], laying the foundation for modern communication security. In terms of hardware security,

Paul has proposed many methods for side channel attacks and defense, making outstanding contributions to the hardware protection of IoT security chips [3]. In 2012, Nigel emphasized the importance of protecting IoT devices, communication, and user privacy. In his paper, he detailed the challenges that may be encountered in IoT environments and proposed corresponding solutions [4], providing comprehensive guidance for the application of cryptography in IoT and providing important information on IoT security for future researchers. In addition, Gene studied the security issues of home automation protocols in 2016, providing guidance for the practical application of IoT security [5]. Contemporarily, with the comprehensive popularization of IoT technology, IoT security has exposed various problems, including but not limited to cloud platform access control defects, communication protocol vulnerabilities, communication traffic side channel information leakage, and device firmware vulnerabilities. The emergence of these issues directly reflects the urgency of IoT security, and researchers have also provided specific solutions to address these issues. For the issue of access rights management on the SmartThings platform, ContextIoT proposes to extract information such as execution paths, data dependencies, real-time variable values, and environmental parameters within the SmartApp to represent the context information during application execution [7]. Before the operation is executed, it actively solicits user authorization permission. Only operations authorized by the user can continue to be executed. In terms of secure communication protocols, Alshahrani et al. proposed a model for mutual authentication and key exchange between devices based on ZigBee communication, which can enhance the robustness of the ZigBee protocol when applied in adversarial environments [8]. In terms of building a trusted firmware runtime environment, some studies have divided the firmware into different components to implement minimum permission isolation, e.g., EPOXY [9], MINION [10].

This study analyzes the complexity and challenges of IoT security research compared to traditional computer information security.Sec. 2 introduces the design and composition of IoT security chips, which integrate hardware security functions such as encryption engine, random number generator, secure storage, and identity authentication. It mainly introduces the roles of IoT security chips in the perception layer, network layer, and application layer of IoT, Implement different levels of security protection. Subsequently, several modules proposed to address security issues in the IoT security field were introduced, including Trusted Platform Module (TPM), Hardware Security Module (HSM), and Trusted Execution Environment (TEE). Sec. 3 will introduce these models separately, including their construction principles, basic functions, design focus, and specific application scenarios. At the end of Sec. 3, a comparative analysis will be conducted on the performance, applicability, and other aspects of these three models to more intuitively represent their characteristics. Sec. 4 will summarize the entire article, mentioning the current popular applications and future development prospects of IoT security chips.

2. Basic Principles

With the rapid growth of IoT devices, security issues have become increasingly prominent. The deployment environment and network structure of IoT are extremely complex, involving a large number of devices, data, and communication, and security needs to be ensured at different levels and links. Compared with traditional computer information system security, IoT security has the following difficulties. IoT involves various types of devices, protocols, and applications, These heterogeneity make unified security policies and mechanisms complex. Devices such as sensors, actuators, and embedded devices may have differences in computing power, storage capacity, and communication methods. The communication technologies supported between devices and systems in the Internet of Things are also different, e.g., WiFi, Bluetooth, LoRa, NB IoT. When formulating security policies, these heterogeneity should be taken into account and appropriate solutions should be taken for different situations. Due to the fact that IoT systems typically contain a large number of devices, which may involve billions or even more nodes, resulting in a large amount of communication and data sharing. However, computing and storage resources are very limited, which can cause many problems such as management inconvenience and data privacy leakage. Therefore, developing powerful IoT security chips is becoming increasingly important. IoT security chips are a specially designed hardware component specifically designed to protect system security. They integrate various hardware security functions such as encryption engines, random number generators, secure storage, and identity authentication to protect devices from physical attacks, data leaks, and unauthorized access. According to different requirements, it can be used at three levels in the IoT architecture to achieve different levels of security protection. The structure of the Internet of Things is generally divided into three layers, namely the perception layer, network layer, and application layer. Building a security architecture for the Internet of Things based on these three layers of structure. Fig. 1 shows the security architecture of the Internet of Things.

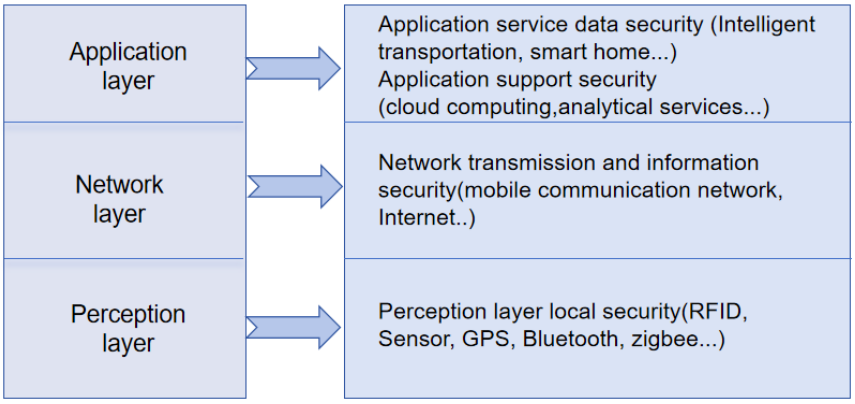

Fig. 1 loT security architecture

The main goals of application layer security policies are twofold: protecting the integrity of application service data and ensuring the security supported by the application. Firstly, the security measures at the application layer perform user authentication and authorization, allowing only authorized users to access the application, effectively improving the credibility of the entire system. Secondly, implementing a firewall at the application layer can be used to resist malicious attacks, which effectively limits the entry of potential threats. At this layer, sensitive data can also be encrypted during data processing, such as anonymization technology. In the application layer, IoT security chips mainly achieve multiple key functions such as identity authentication, encryption and decryption, and digital signature. They can also provide security authentication for communication between devices and application programs, further consolidating the overall security of the system.

The security policies of the network layer are applied in multiple important aspects. The primary goal is to ensure the integrity of information during data transmission. By using strict security policies, the transmission of information is tightly protected to prevent any form of tampering or damage to the data during transmission. Another aspect focuses on identity authentication when connecting devices to the network. By verifying whether the devices connected to the network have legal identities, the system can effectively prevent unauthorized device access. In the network layer, IoT security chips mainly use security protocols (e.g., SSL and DTLS) to encrypt data transmission, and sensitive data and critical information can be reliably protected during transmission.

The core responsibility of the security strategy of the perception layer is to ensure the security during the data collection process of IoT devices and sensors. By embedding IoT security chips, hardware level security guarantees are provided for these devices. This chip can generate high-quality random numbers to implement important functions such as data encryption and key management, and this hardware level protection mechanism lays a solid foundation for device operation and data collection. In addition, IoT security chips also play a role in preventing physical attacks at the perception layer, including physical dismantling and side channel attacks. By embedding a security chip in the device, it can effectively prevent attackers from illegally invading the device and physical attacks, and provide a foundation for the entire system security at the hardware level. In summary, IoT security chips play a crucial role in the entire IoT structure, and can be applied at various levels to provide rich and powerful protection at every stage of task execution.

The following are typical algorithms that may be used in various modules of IoT security chips. As for encryption module, classic algorithm including AES (Advanced Encryption Standard), which is a symmetric encryption algorithm that uses the same key to encrypt plaintext and generate ciphertext for system use. The formula is Ciphertext=AES_Encrypt(Key,Plaintext). As for authentication module, HMAC (Hash based Message Authentication Code), which combines a key and a hash function to perform hash operations on the message and encrypt the hash results using the key. Formula: Token=HMAC(SecretKey,Data). Regarding to key management module, RSA (Rivest Shamir Adleman), which is an asymmetric encryption algorithm that uses public key encryption keys for secure storage or transmission of data. Formula: EncryptedKey=RSA_Encrypt(PublicKey,Key). There are also AES-GCM algorithms for secure storage modules and ECDSA algorithms for digital signature modules. The specific chip design will select different algorithms based on design concepts, performance requirements, and other factors to meet different security needs.

In order to achieve the appeal security maintenance function and isolate untrusted components, multiple models have been proposed to address security issues, each with unique applications in different fields. Outstanding examples include Trusted Platform Module (TPM), Hardware Security Module (HSM), Security Element (SE), and Trusted Execution Environment (TEE) [11]. According to different system requirements and security requirements, the specific model of IoT security chips can vary according to their design and functional characteristics. The following is a detailed introduction to the principles and characteristics of three classic modules.

3. Classic Models

3.1. TPM

TPM module: Trusted Platform Module, which is a chip installed inside a computer. This chip is a chip planted inside a computer to provide a trusted root for the computer. The specifications of this chip are developed by the Trusted Computing Group [12].TPM enhances the security and credibility of the system by providing functions such as encryption, authentication, key management, and data protection. TPM guarantees protective computing in all environments such that TPM provides integrity because it measures the platform (computing device) status and ensures that the platform is trustable. TPM provides authentication which guarantees that the platform can prove that it is the intended entity [13]. This makes TPM an important component of many security applications and fields, including digital signatures, data privacy, secure boot, etc.

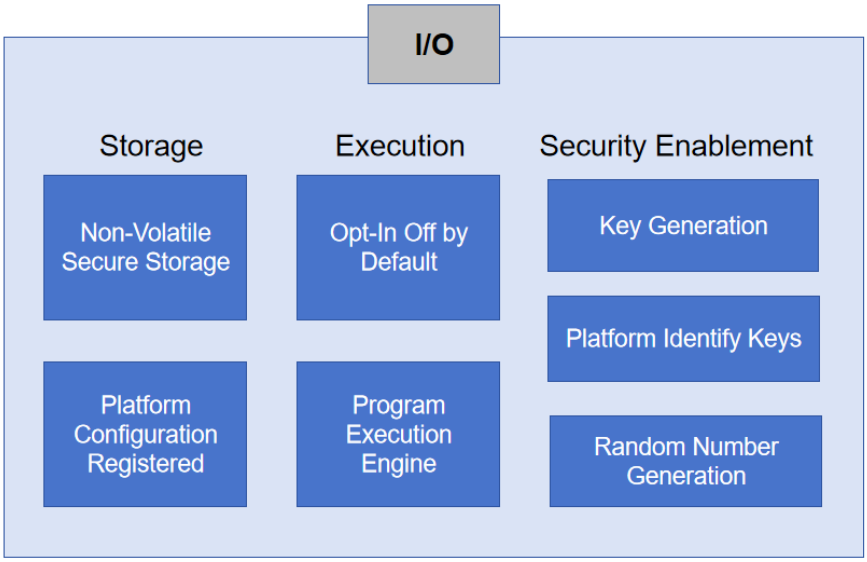

Fig. 2 Components in TPM.

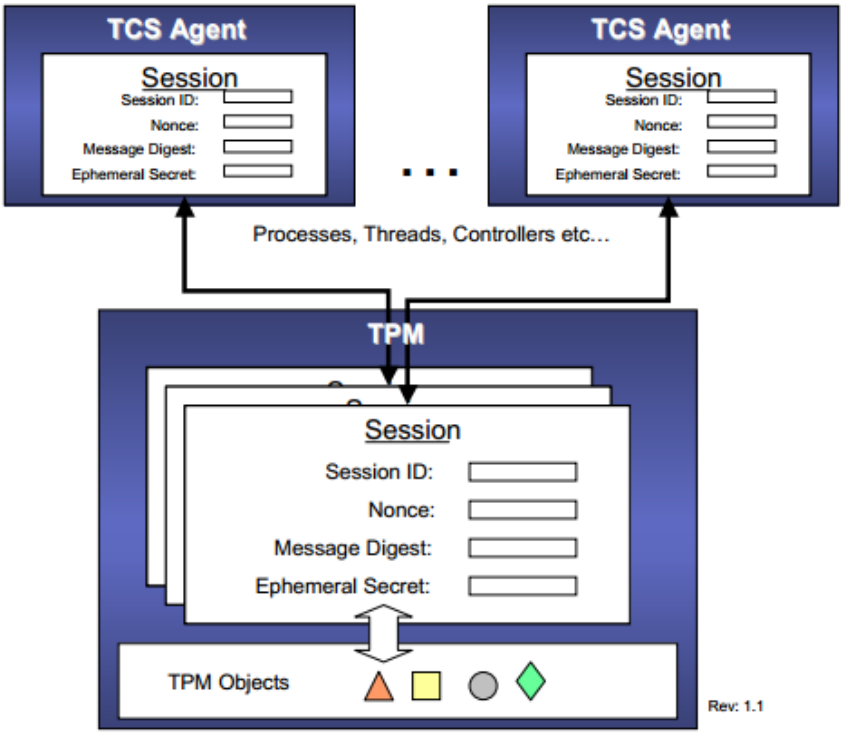

Fig. 3 TPM for managing multiple Command Validation sessions.

Fig. 2 shows the relevant components in TPM, and the main components are introduced. Non-Volatile Secure Storage: refers to a special area inside the TPM chip used to store sensitive data and keys. These data and keys are protected and encrypted by hardware, ensuring that they are not stolen or modified during storage and transmission. Opt In Off by Default: Ensure that the TPM can only be started with user permission, while in other cases, the TPM function is turned off by default. Secure Platform Configuration Registered: It is used in TPM to store information about computing devices and system configurations, such as unique identifiers related to devices, hardware parameters, operating system settings. The TCG specification defines 16 PCRs [14], of which 8 are reserved for internal TPM use and the remaining 8 are for operating systems and user applications. Secure Program Execution Engine: It mainly completes hardware isolation, confirms a trustworthy environment, ensures secure startup, and other mechanisms to protect critical operations in the environment from interference.Key Generation: It is used to generate TPM encryption keys. The Platform Identity Keys are Use public and private keys to verify the identity of computing devices, ensure communication security, and generate digital signatures [15].

During the process of submitting commands from entities (Processes, Threads, Controllers) to the TPM, a secure communication channel is formed between the entity and the TPM, which can be a physical connection (such as a hardware interface) or a software interface (such as a TPM API). And establish a session, Fig. 3 illustrates a TPM that uses external entities to manage multiple Command Validation sessions. The purpose of establishing a session is to ensure that access to TPM objects is authorized [16]. The session includes a unique session identifier, random number, command code (representing the operation to be performed), etc. The TPM will parse the command data and perform corresponding operations. After the operation is completed, the TPM will generate a response code and transmit it back to the entity, which can then proceed with subsequent operations.

The following is an introduction to the algorithms and corresponding formulas that may be involved in TPM: 1 RSA encryption and decryption: Public key encryption: $C = M \times e \bmod n$, Private key decryption: $M = C \times d \bmod n$, C is ciphertext, M is plaintext, e is the exponent of the public key, n is the modulus of the public key, and d is the exponent of the private key. RSA (Rivest Shamir Adleman) is an asymmetric encryption algorithm used for encrypted communication, digital signatures, key exchange, and establishing trust systems to protect data confidentiality, integrity, and identity verification. 2. HMAC algorithm: HMAC generation: $\mathrm{HMAC}(\mathrm{Key}, \mathrm{Message}) = \mathrm{H}\big[(\mathrm{K} \times \mathrm{opad}) || \mathrm{H}[(\mathrm{K} \times \mathrm{ipad}) \,||\, \mathrm{Message}]\big]$. Among them, Key is the key, Message is the message, opad and ipad are specific constants, and H is the hash function. HMAC is a method of generating message authentication codes by applying hash functions and keys to messages, used to verify the integrity and authenticity of messages. 3. Digital signature generation and verification: TPM can generate and verify digital signatures, using algorithms such as RSA and ECDSA. The formula for generating digital signatures: $\mathrm{Sig} = \mathrm{Sign}\big(\mathrm{PrivKey}, \mathrm{Hash}(\mathrm{Message})\big)$. Sig is the digital signature, Sign is the signature algorithm, PrivKey is the private key, and Hash is the hash value of the message. 4. Key Derivation Algorithm: Generate derived key using hash function: $\mathrm{DerivedKey} = \mathrm{H}(\mathrm{Secret} \,||\, \mathrm{Salt})$. Secret is the primary key, Salt is the value used to increase randomness, and DerivedKey is the generated derived key.

3.2. HSM

Hardware security module (HSM) is usually interpreted as a hardware device that can be used for secure management or storage of keys, and can provide password calculation operations, including a security centric operating system with protected memory, program code, data flash memory, accelerator, and true random number generator. HSM is based on the principles of physical isolation and strict access control, used to perform key generation and management, digital signature, and data encryption and decryption. HSM is usually used as a part of critical infrastructure such as public infrastructure or online banking, and can also be used for the development of automotive components. Generally, multiple HSMs are used simultaneously, achieving dual hardware and software isolation. Fig. 4 shows the multiple key modules contained within the HSM OS.

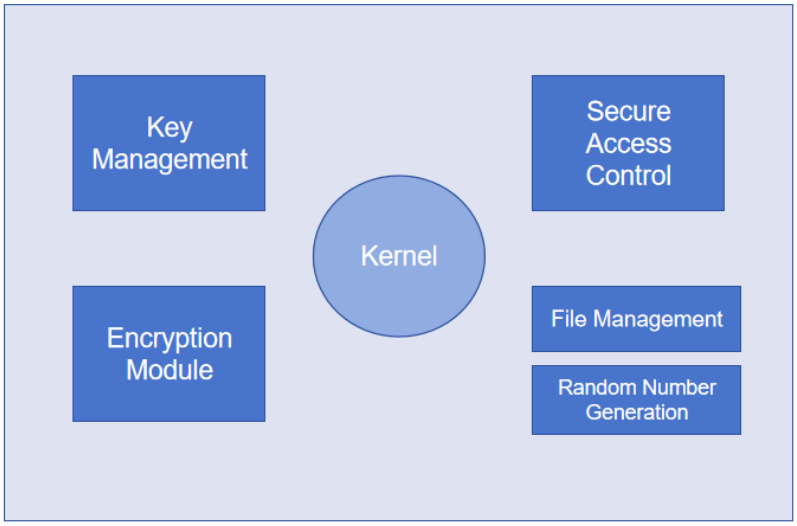

Fig. 4 Key modules contained within the HSM OS

Kernel is the core module of the operating system, used to schedule operational tasks, determine which task runs first, allocate CPU time slots, and ensure isolation between different tasks to prevent interference between them. Key Management Module is responsible for generating, storing, and managing keys. These keys are the core of security operations such as encryption, decryption, and digital signature, and will be used for data storage and transmission processes. Encryption Module: This module contains an encryption engine for performing encryption and decryption operations. It supports multiple encryption algorithms and modes to ensure the confidentiality of data during storage and transmission. Secure Access Control Module implements functions such as identity authentication, access control, and permission management. It ensures that only authorized users can access sensitive data and resources. There are also digital signature modules, random number generation modules, and file management modules, which work together to complete the core functions of the HSM operating system. Different HSM manufacturers may implement different modules and functions in their operating systems to meet different security requirements.

The following is an introduction to the algorithms and formulas that may be used in HSM: 1 Symmetric encryption: Use the same key for encryption and decryption. Encryption: Ciphertext=Encrypt(Key, Plain text), Decrypt: Plaintext=Decrypt(Key, Cipher text). Asymmetric encryption operation: Asymmetric encryption is used for digital signatures, key negotiations, etc. Digital signature generation: Signature=Sign(PrivateKey, Data), Digital signature verification: Valid=Verify(PublicKey, Data, Signature), Key negotiation: SharedSecret=DH(PrivateKey, OtherPublicKey). 3. Digital certificate operation: HSM is typically used to process digital certificates for authentication and secure communication. Generate CSR: CSR=GenerateCSR(SubjectInfo, Public Key), Verify digital certificate: Valid=VerifyCertificate(Certificate, IssuerPublicKey). EVITA (Embedded Virtually Integrated Trusted Architecture) is a hardware security standard aimed at defining a trusted security architecture in embedded systems. The EVITA HSM standard provides unified hardware safety features and functional requirements for HSM, aiming to provide highly secure solutions for automotive electronics and other embedded systems. EVITA divides HSM into three levels of specifications: Full, Medium, and Light. These levels represent the functional levels of HSM, and users can choose the appropriate level based on the application's needs. Table 1 lists the requirements that each level can provide.

Table 1. Requirements That Each Level Can Provide

	Full HSM	Medium HSM	Light HSM
RAM(random-access memory)	√	√	Optional
NVM(non-volatile memory)	√	√	Optional
Symmetric cryptographic engine	√	√	√
Asymmetric cryptographic engine	√		
Hash engine	√		
Counters	√	√	Optional
Random-number generator	√	√	Optional
Secure CPU	√	√	
I/O component	√	√	√

HSM must meet a series of strict safety and functional requirements. This includes physical isolation, resistance to side channel attacks, identity verification mechanisms, and trusted startup processes. Full level HSM is suitable for applications with high security requirements, such as the financial, military, and automotive industries. HSM needs to complete some hardware protection measures, authentication and authentication mechanisms, as well as a certain level of secure startup. This level is suitable for enterprise level applications and some in vehicle electronic systems. HSM only needs to complete some basic security protection tasks, such as simple authentication and identity verification. The Light level HSM is suitable for applications with low security requirements, such as some smart home devices and relatively simple embedded systems. This classification system enables the EVITA standard to be applicable to various types of applications, providing a unified hardware security module solution for various scenarios.

3.3. TEE

Rich Execution Environment (REE) refers to an execution environment that provides rich functionality and resources in a computing system. It covers a variety of hardware and software components to support the operation of various complex tasks and applications. REE typically provides diverse functions and services to meet different types of computing needs. Trusted Execution Environment (TEE) is a tamperresistant processing environment that runs on a separation kernel. It guarantees the authenticity of the executed code, the integrity of the runtime states (e.g., CPU registers, memory and sensitive I/O), and the confidentiality of its code, data and runtime states stored on a persistent memory [17]. REE covers a variety of hardware and software components to support the operation of various complex tasks and applications. REE typically provides diverse functions and services to meet different types of computing needs. There are four primary components in TEE: two execution environments which are the Rich Execution Environment (REE) and the Trusted Execution Environment (TEE), and two storage areas which are External Volatile Memory and External Non-Volatile Memory [18]. Fig. 5 shows the software architecture of TEE, and the main modules will be introduced below.

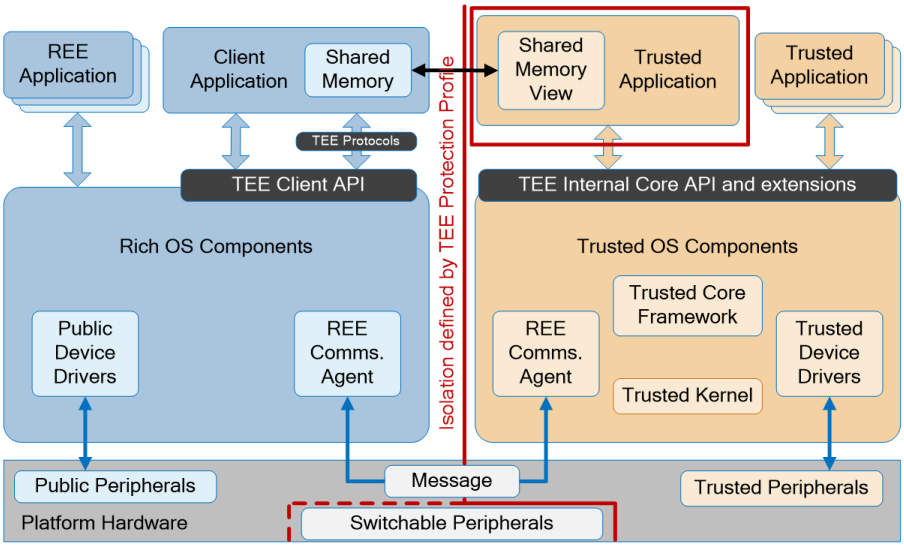

Fig. 5 TEE software architecture

In the Rich Execution Environment (REE) system, Client application mainly corresponds to upper-level applications and serves as a bridge between users and system resources to provide various services, such as completing payment applications and fingerprint collection functions. Its main function is to achieve data protection and perform sensitive tasks, such as data authentication, digital signature, etc. The core part of TEE that manages resources and functions, provides protection for sensitive data, and provides a secure execution environment. The core components are responsible for managing resources and implementing security isolation between TEE applications. The underlying hardware platform of TEE, including processors, memory, security modules, etc. It provides hardware level security protection, ensuring isolation between different execution environments, and is a key foundation for forming a trusted execution environment.

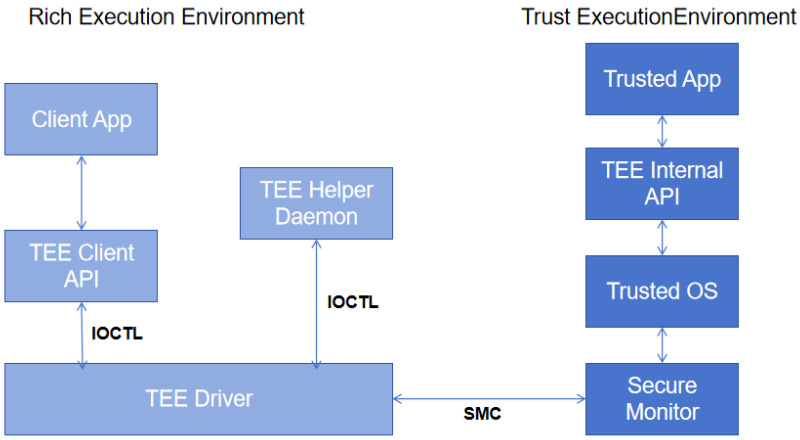

Fig. 6 the interaction between Client APP and Trusted APP

Fig. 6 shows the interaction flowchart between the Client APP and the Trusted APP. Next, This study will introduce the basic interaction process in detail. Firstly, the Client APP starts the execution environment through the TEE Client API, which communicates with the TEE Driver through ioctl system calls. Next, the system entered the kernel state of the Rich Execution Environment and searched for and called the corresponding REE driver. After finding the driver, the system will call the SMC (Secure Monitor Call) assembly instruction to enter Secure Monitor mode. The Secure Monitor mode is mainly used to manage key tasks such as environment switching and request forwarding between REE and TEE. After switching to the Trusted Execution Environment environment, TEE OS will call the corresponding Trusted APP through the TEE Internal API based on the Client APP's service request. This process includes ensuring the security and legality of the request, as well as preparing to run Trusted APP in TEE. Trusted APP will perform tasks in a

protected environment, which may involve sensitive operations such as data protection, computation operations, encryption and decryption. After completing the task, the Trusted APP will generate running results and data, and return them to the Client APP. After the interaction is completed, the Client APP will close the connection with the Trusted APP, release resources, and end the interaction task. The process described above describes a basic interaction process between Client APP and Trusted APP. In reality, this process may vary due to factors such as differences in choosing TEE platforms and adopting different security strategies.

3.4. Comparison of Three Models

Table 2 gives the comparison of the three models. TPM mainly completes key management and data encryption functions, ensuring system security, but its functions are relatively limited; HSM has hardware isolation and can be used for security protection of advanced cryptographic operations; TEE provides application level hardware isolation, providing a secure execution environment for application data and code. TPM is commonly used on platforms such as PCs and servers, and common standards include TPM 1.2 and 2.0; HSM is commonly used for data protection in the financial and automotive fields, with common standards being PCI HSM Security Requirements and Assessment Procedures; TEE is mainly used for mobile devices and IoT devices, with the common standard being the GlobalPlatform TEE Standard. The advantage is that it provides root security functionality and hardware isolation, while the disadvantage is that it is not suitable for mobile devices and can only be integrated into specific hardware chips, requiring some system support. The advantage of HSM is that it can prevent physical attacks and side channel attacks, and is suitable for environments with higher security requirements. The disadvantage is that it is a separate hardware device, which can cause unnecessary costs. TEE: The advantage lies in its low power consumption and lightweight characteristics, which can be used for mobile devices. The disadvantage is that it is limited by the support of the underlying hardware platform, and its running functions may be limited.

Table 2. Comparison of Three Models

0	TPM	HSM	TEE
Isolation and Security	System level isolation	Hardware isolation	Application level isolation
major function	Key management, data encryption	Advanced Cryptographic Operations	Provide a secure execution environment
Common fields	Computer Platform	Financial, automotive sectors	Mobile devices, IoT devices
Common standards	TPM 1.2 and 2.0	PCI HSM Security Requirements and Assessment Procedures	GlobalPlatform TEE Standard

4. Conclusion

In response to the current difficulties faced by IoT security and the importance of IoT security, this article mainly introduces the basic design principles of IoT security chips, including the specific applications of security chips in the perception layer, network layer, and application layer. Three classic models, TEE, HSM, and TPM, were introduced in detail, including their basic structures and classic applications. And a comparative analysis was conducted on these models, listing the advantages and disadvantages of each model. With the continuous development and growth of the Internet of Things, the development of stronger security chips is crucial. IoT security is a highly challenging field that requires comprehensive consideration of security issues at different levels and stages. IoT security chips play a crucial role in ensuring the security and data privacy of IoT devices. For example, IoT security chips are embedded in smart home devices, such as smart door locks and smart cameras, to provide user authentication functions. Embedding sensors and cameras into the transportation system can achieve secure communication to prevent hackers from tampering with data; Used to protect the privacy of medical data and prevent unauthorized access to patient data. In the

future, with the rapid increase in the number of IoT devices, the development of IoT security chips is full of potential, One will continue to integrate with emerging technologies such as blockchain, artificial intelligence, quantum encryption, etc. to enhance security and credibility. With the diversification of IoT application scenarios, security chips may develop towards more personalized directions to meet the specific needs of different industries and applications. Overall, IoT security chips will become an important support for ensuring the security of the IoT ecosystem, and are expected to achieve greater breakthroughs in security, privacy protection, and credibility.

References

[1] Diffie W, Hellman M E. Special Feature Exhaustive Cryptanalysis of the NBS Data Encryption Standard. Computer, 1977, 10(6): 74-84.

[2] Rivest R L, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM, 1978, 21(2): 120–126.

[3] Kocher P C. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: Koblitz, N. (eds) Advances in Cryptology — CRYPTO '96. CRYPTO 1996. Lecture Notes in Computer Science, vol 1109. Springer, Berlin, Heidelberg.

[4] Katagi M, Moriai S. Lightweight cryptography for the internet of things. Sony Corporation, 2008, 2008: 7-10.

[5] Tsudik G. The Security of Home Automation Protocols. IEEE Computer Society, 2016.

[6] Yang W, Zhou W, Zhao S et al. Survey of IoT security research: threats, detection and defense. Journal on Communications, 2021, 42(8): 188-205.

[7] Mrabet H, Belguith S, Alhomoud A, et al. A Survey of IoT Security Based on a Layered Architecture of Sensing and Data Analysis. Sensors (Basel), 2020, 20: 13.

[8] Mohammed A, Issa T, Isaac W. Anonymous mutual IoT interdevice authentication and key agreement scheme based on the ZigBee technique. Internet of Things, 2019, 7.

[9] Clements A A, Almakhdhub N S, Saab K S, et al. Protecting bare-metal embedded systems with privilege overlays. 2017 IEEE Symposium on Security and Privacy (SP). IEEE, 2017: 289-303.

[10] Kim C H, Kim T, Choi H, et al. Securing Real-Time Microcontroller Systems through Customized Memory View Switching. NDSS, 2018.

[11] Jauernig P, Sadeghi A R, Stapf E. Trusted Execution Environments: Properties, Applications, and Challenges. IEEE Security & Privacy, 2020, 18(2): 56-60.

[12] Goyal S, Mathew R. Security issues in IOT computing. Proceedings of the International Conference on Computer Networks, Big Data and IoT, Springer, Madurai, India, 2020.

[13] Mohammad F, Ikram A, Muhammad S K, Su M K, Junsu K. Establishment of Trust in Internet of Things by Integrating Trusted Platform Module: To Counter Cybersecurity Challenges. Complexity, 2020, 2020: 6612919.

[14] Deepthi S L M, Yamini G, Srinivasarao P, Sivaramakrishna K. A novel and cost-effective approach for privacy. International Journal of Advanced Computer Communications and Control, 2014, 2: 3.

[15] Canedo E D, Junior R T, Albuquerque R O. Trust model for reliable file exchange in IOT computing. International Journal of Computer Science and Information Technology (IJCSIT), 2012, 4: 1.

[16] TCG. TCG Specification Architecture Overview. TCG Specification Revision 1.4, The Trusted Computing Group, Portland, OR, USA, 2007.

[17] Sabt M, Achemlal M, Bouabdallah A. Trusted Execution Environment: What It is, and What It is Not. 2015 IEEE Trustcom/BigDataSE/ISPA, Helsinki, Finland, 2015: 57-64,

[18] Arfaoui G, Gharout S, Traoré J. Trusted Execution Environments: A Look under the Hood. 2014 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, Oxford, UK, 2014: 259-266.