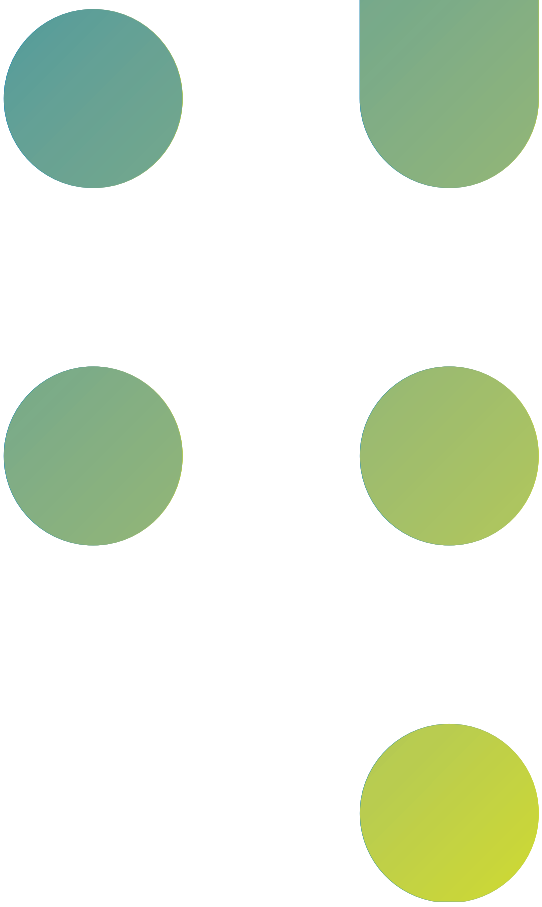


IOT SECURITY SOLUTIONS

White paper

vault

TABLE OF CONTENTS

EXECUTIVE SUMMARY

The most important security functions to secure the Internet of Things connected devices are :

- Authentication: confirming the identity of the communication peer,
- Secure communication: protecting data in transit,
- Secure execution of code : protecting data in process,
- Secure storage: protecting data at rest.

This document provides focus on security mechanisms and Inside Secure range of solutions, from software-only to hardware IP Core and up to a standalone embedded secure element.

BACKGROUND INFORMATION

Spread of computing resources

Computing history is a tale of distribution of computing resources, once the exclusive property of centralized and isolated computer centers have spread out from the mainframes to an ever widening grid of interconnected and computationally capable nodes.

The past few decades have witnessed wave after wave of further distribution of computing capability – first, in the 1980s, the rise of the personal computer, later followed by the explosive increase of networking as the Internet revolutionized the world at the turn of the millenium. This was eventually followed by the mobile computing of smartphones and tablets. At each development step the computing units have increased in numbers and decreased in physical size.

Today, advances in networking and semiconductor technologies have enabled the

next step: the emergence of the Internet of Things (IoT).

Internet of Things

The Internet of Things (IoT, sometimes "Internet of Everything") is a network of physical objects (or "things") embedded with electronics, software, sensors, and connectivity to enable those objects to exchange data with the manufacturer, operator and/or other connected devices. The IoT is based on the infrastructure of the International Telecommunication Union's Global Standards Initiative (IoT-GSI)[1]. It covers devices and objects connected over the Internet Protocol (IP) such as personal computing devices, laptops or desk computers, tablets, smartphones, and also devices that are connected to each other through non-IP protocols (e.g. Bluetooth, ZigBee...).

The IoT devices are typically thought of as "smart devices", such as networked home appliances remotely monitored or controlled; "smart home" components, such as lighting, heating, or ventilation units with remote management/monitoring access; sensor networks for industrial automation; networked vehicle telematics sensors, and a multitude of other embedded devices that are network-connected and computationally capable.

IoT devices (or nodes) which often operate without any screen or even without any user interface at all, may rely on battery power for operation, and are usually dedicated to a single task. We only mention a few examples here, the range of devices and use cases that are seen as inclusive to the Internet of Things being extremely wide. It stretches from the seemingly trivial such as toys or entertainment devices, to consumer devices such as wearable smartwatches and personal health devices; to the clearly mission critical, such as smart energy grid and medical technology; and leading to a large variety of architecture of edge devices connected to routers, gateways...

Proliferation of connected devices

Some noteworthy developments have made the roll-out of connected devices possible. The mobile Internet revolution introduced a number of wireless protocols that have now grown as the "over-the-air" (OTA) extensions of the traditional wired Internet. These wireless protocols – GPRS, UTMS, LTE, WiFi, and new low bandwidth network such as LoRA, etc. – have spread Internet coverage to an almost omnipresence.

For a while, it seemed that only the shortage of addresses could slow down the rapid expansion of the Internet. This critical shortage was soon overcome – an important enabling feature of IoT was the dramatic expansion of the IP addressing space brought on by the introduction of IP version 6 (IPv6) that can guarantee an address for each node.

A second critical development that paved the way for IoT is the readily and economically available computation power brought on by the present level of miniaturization of

[1] IoT-GSI "promotes a unified approach in ITU-T for development of technical standards (Recommendations) enabling the Internet of Things on a global scale".

integrated circuitry. Modern chip design combined with advanced semiconductor technology provides cost-effective and dense computational capability with sufficiently low-power requirements, which enables each addressed node to be capable of computing on a level that was in the past restricted to centralized specialized computers.

Once the addressability and computing power challenges are solved, the next biggest concern to address in IoT is security and privacy. The control of the connected devices[2] and the generated data is certainly one of the major challenges to solve alongside the expansion of the IoT. While the information processed by these IoT nodes seems innocuous in itself, the sheer volumes of the collected, processed and transferred information may have serious implications for both public safety and individual privacy. The security risks of IoT deployments vary a great deal; an attacker that gains illegitimate access to the location data of a certain wearable computing device certainly invades the privacy of the wearer, while an attacker that compromises the security of a smart energy network may be a threat on a national level.

SECURITY OF DEVICES AND COMMUNICATION BETWEEN DEVICES IN THE IOT - OVERVIEW

Risk Assessment

The enormous number of devices[3] connected through heterogeneous infrastructures increases the risk of attacks. The potential attack of an IoT infrastructure (network or end devices) generates risks, including: loss of control of the application, denial of service, switching off (e.g. smart grid application), user privacy loss, fraud, terrorist attacks... with heavy social consequences such as loss of revenue, liability issues, brand damage, people's health, job destruction...

When implementing an IoT infrastructure a security risk analysis must be conducted in order to evaluate the effect of a successful attack and to define the best security solution to be implemented.

[2] 70% of the most commonly used IoT devices contain vulnerabilities, including password security, encryption and general lack of granular user access permissions. See Hewlett Packard IoT Research Study (http://h20195.www2.hp.com/V2/GetDocument.aspx?docname=4AA5-4759ENW&cc=us&lc=en)

[3] New data from Juniper Research has revealed that the number of IoT (Internet of Things) devices will number 38,5 billion in 2020, as opposed to 13,4 billion in 2015: a raise of over 285%.

Hackers or malicious users are motivated by various considerations, such as money, having fun, technical challenge, terrorism, etc. To achieve their goals, they deploy solutions to abuse the functionalities of a device or extract information from it.

The hacker puts in perspective the reward of the hack vs. the "cost" and the "risk" of the attack: the time he spent to perform the attack, the cost of equipment needed to perform the attack (economical barrier). The expertise required to perform the attack is a good example of "cost". An example of "risk" includes legal penalty if caught (fine, prison, etc.). So security must be sized according to the consequence of a hack, not to the value of the device.

The device that the hacker targets may have different levels of accessibility: can the hacker have physical access to the device? What level of information on the system, i.e. level of collusion, can the hacker access ? Security must also be sized to the environment where the device is running and its access to the hacker.

In most case it is faster and cheaper to implement the right security level when designing a system rather then trying to increase the security of an existing system already deployed.

And remember: security is a chain only as strong as its weakest link.

Attacks

IoT IT architecture and IoT devices are potentially open to a huge number of attacks. Attacks may be performed at different levels :

- At network level, the hacker has only access to the device through the network or through the applications offered by the IoT solution provider (eavesdropping).
- At device level, the hacker has also access directly to the device, and can perform additional attacks, i.e. invasive or semi-invasive attacks.
- At chip level, the hacker can physically perform attacks on chips located in the device (e.g. reverse engineering).

Due to the diversity of IoT devices and to the emergence of the market, there is no one standard today describing the potential attacks or vulnerabilities for this market. Still some standards exist and can provide a good reference for IoT: Common Weakness Enumeration (CWE™) and the Common Attack Pattern Enumeration and Classification (CAPEC™) provide a presentation of software vulnerabilities and attack methods; FIPS 140-2 from US authorities; Common Criteria developed by the SmartCard industry.

Some examples of attacks follow :

- Software attacks: the attacker finds and exploits vulnerabilities in protocols, crypto-algorithms, or in their implementation to bypass security mechanisms. These attacks may be very efficient and usually do not require sophisticated equipment. They can sometimes be performed through the network without

direct physical access to the device.

- Exploitation of Test features: the attack path aims to enter the IoT device (or Integrated Circuit in the IoT device) in test mode to provide a basis for further attacks (e.g. Disclosure or corruption of memory content to retrieve user data like crypto-keys or device configuration.
- Attacks on Hardware as an example, attacks on RNG (Random Number Generator): random numbers are at the basis of cryptography. Reducing the entropy of a RNG significantly reduces the security level of the system.
- The importance of quality randomness generation cannot be over-stressed in the context of embedded device security. Some very prominent crypto-system failures have been traced back to poor random number generation implementation. True random number generation on embedded systems is a genuinely difficult task, yet true randomness is vital for high-quality generation of encryption keys. The strength and quality of the random number generation is in direct relation to the strength and quality of the cryptographic security system. For details, see the following white paper "The importance of true randomness in cryptography"[4].

Cryptography

Cryptography is the practice and study of hiding information. It is the science used to try to keep information secret and safe[5].

We cannot be exhaustive on this subject here. You will find some basic information about cryptography in this section. You will also find additional information spread throughout this document to explain how cryptography is used.

Cryptography can be divided into three main functions :

- Authentication: how to establish the true identity of a peer
- Confidentiality: how to keep information secret
- Integrity: how to detect if data was modified

These three functions can be used individually or combined to create a solution.

Several algorithms have been developed and deployed to perform such functions. They can be divided into three groups:

- Symmetric algorithms (sharing a single secret key between two communicating peers). If the key becomes public, the system collapses. The difficulty is to exchange the keys between the two peers. Such algorithms are quite efficient in terms of speed and computing power. DES and AES are typical standard algorithms, but AES provides a better resistance than DES. These algorithms are mainly used for Confidentiality purpose (encryption/decryption).
- Asymmetric algorithms (using a public/private key pair), only the public key

[4] The importance of True Random Number Generator in Cryptography (Inside Secure, 2015) Download on Inside Secure website.

[5] Source : Wikipedia.

needs to be exchanged. They are usually more computing power consuming than symmetric algorithms. Typical algorithms are based on RSA and ECC. By design, ECC requires lower key size than RSA for the same security level. The trend is to move from RSA to ECC as security requirements mandate key size increase. These algorithms are mainly used for Authentication purpose (signature/verification). Messages signed with the private key can be verified only by using the associated public key. Successful verification of the message using the peer's public key assures that only the peer can have sent the message.

- Hashing : These algorithms are mainly used for hashing purposes (signature/verification).

Cryptography is a robust and proven technology if it is used with standard algorithms and appropriate key sizes. Some national agencies provide recommendations for algorithms and key sizes[6]. It is highly recommended to use standard and proven algorithms as opposed to private ones. The reason is that the standard algorithms have been extensively analyzed and tested. History has shown that the use of private algorithms exposes the user to easy to exploit vulnerability (DVD...).

Finally, in cryptography, a crypto system should be secure even if everything about the system, except the key, is publicly known (Kerckhoffs's principle).

Therefore, the reading, writing of keys and execution of crypto-functions must be performed within a controlled environment to avoid any possible key extraction or modification within the IoT.

SECURITY FUNCTIONS OF INTERNET OF THINGS: THE FOUR PILLARS OF SECURITY

Regardless of the use case (energy/smart grid, industrial automation, connected home, wearable computing, etc.) the IoT nodes share common basic security needs which stand as pillars of security :

- Authentication: confirming the identity of the communication peer
- Secure Communication: protecting data in transit

[6] The NIST provides guidelines for selecting different crypto-algorithms and associated key sizes.

- Secure Execution of code: protecting data in process
- Secure storage: protecting data at rest

Cryptography is the foundation of these four pillars, as it is used to authenticate, encrypt/decrypt information, sign, and check data integrity.

Hardware and software technologies are available to authenticate and protect data at rest, data in process, and data in transit.

After a risk assessment, the selection of security solutions should be based on :

- 1. the right security functions.
- 2. the right architecture choice (hardware and/or software).
- 3. the most suited crypto algorithm.

FIRST PILLAR: AUTHENTICATION

Authentication is the process in which the communicating peers identify each other and assure each other of their identities. The IoT deployments imply a vast number of interconnected and distributed endpoints that need to communicate, which highlights the importance of the strength, reliability, and scalability of the authentication methods used. Each endpoint must be controlled in order to make sure each peer is guenuine and to avoid insertion of fake devices into the network.

The most common (and simple) form of authentication is based on sharing a common secret. AES or SHA-256 are typical examples of algorithms used for authentication with a symetric key. But remember, this methodology is highly vulnerable because if the shared secret is compromized, the entire security mechanism collapses.

A better authentication method relies on the use of asymmetric, public key cryptography. Public key authentication is commonly used in Internet connected servers and devices to provide strong authentication. ECDSA is a typical example of an asymetric algorithm used for authentication.

The use of public key cryptography requires access to a Public Key Infrastructure (PKI). PKI is applicable to a wide variety of secure communications protocols (IPsec,

TLS / SSL, DLTS) in a standards-compliant way. A secure implementation of PKI for authentication requires :

- On-chip key pair generation with a key generator using high quality TRNG, and a secure key storage inside the security module (please refer to the specific Secure Data Storage section in this document).
- Execution of cryptographic operations within a controlled environment (signing, signature verification, en/decryption).

Authentication and Root Keys

The challenge in communication is to create an initial trust to make sure that the received public key belongs to the intended communication peer and can be trusted.

A device needs to obtain the “trust” that the public key it uses to verify a signature indeed belongs to the device it “wants to talk to”, and/or belongs to a device it can trust.

This invariably requires the storage of some public key in the device. Even so the public key does not need to be kept secret, but it must be immutable – it must not be possible for an attacker to modify the key, or cause the device to use another key instead. Sometimes it is needed to combine the public key of a device/user with other information to identify the device (Unique device identifier, IP address, domain name, real name and address, etc).

This is what digital certificates are meant to do.

Provisioning and loading the keys in the device is critical and therefore requires particular attention.

Key Exchange mechanism

It is sometime necessary to create and share a common secret between two peers. For instance, a shared secret can be used as a root key to create a subsequent derivative keys thanks to Key Derivation Function (KDF). Such derivative keys could be used as an ephemeral key to communicate using asymmetric cryptographic algorithms. This function can be performed thanks to a key exchange (or key establishment) method. The key exchange main challenge is to exchange information so that no one else but the authorized peers can obtain a copy of it. The Diffie–Hellman key exchange protocol is an example of cryptographic protocol solving this issue, but this protocol has to be combined with other methods to address the authentication issue (being sure of the actual identity of the peer at the other end of the communication channel).

A well-known and probably original authenticated key exchange mechanism, is the ‘Station to Station’ protocol, even if many authenticated key exchange protocols exist, as described in specifications like the NIST SP 800-56 series, or in well-known security

protocols like IKE, TLS, SSH or MIME.

With the Station to Station protocol, the peers still need to be able to trust that the public key they have stored belongs to the entity they communicate with. The identity can be either built-in to the device, if all devices only need to talk to a single 'server', or it can be signed by a trusted third party, if devices need to be able to set up secure connections to multiple other devices. The difficulty in this is that no device can store all possible public keys. One solution is to only store a single 'root key' used by a 'trusted third party' that is used to sign all 'valid' public keys.

Digital Signatures for Device Authentication

Digital Signatures are always related to one or both parties showing proof that they have / know a particular secret.

A 'secret' can be a symmetric key (e.g. HMAC key) allowing both parties to sign requiring possession of the secret HMAC key to complete. A typical use case for symmetric keys is when public-key cryptography is considered too expensive. This provides a system with no 'individual' authentication of a device or entity: both or all peers share the same key. The shared key is used to prove 'membership' in some ecosystem. If the secret is ever compromised, all devices using the same key are compromised.

More often, the 'secret' is an asymmetric private key, which is used to sign a challenge. The associated public key is then used to verify the signature. This implies more complex arithmetic steps, hence more time, more power, but a more finely grained security framework. Each device has its own unique private key.

Digital Signatures can be used for offline authentication (i.e. by signing boot code) and for online authentication (i.e. by signing challenges).

SECOND PILLAR : SECURE COMMUNICATION – PROTECTING DATA IN TRANSIT

As their name implies, the IoT devices leverage existing Internet technologies and protocols. These protocols provide a selection of proven secure communication solutions. The actual use case of the node typically dictates the criteria for selecting the leveraged secure communication protocol – for example, in "Smart Grid" deployments where the communication model typically follows the client-server model, protocols such as SSL/TLS or DLTS are used, while in "full mesh"-type networks a network level security protocol such as IPsec is more applicable.

All three security protocols that are mentioned above can use the earlier introduced and suggested PKI authentication in a standardized way. This guarantees the IoT nodes' seamless integration into and interoperability with the existing standard network infrastructure.

While the data rates to and from an individual node are typically limited in IoT deployments, the constrained nature of the host devices may pose implementation challenges in the context of cryptographically secured communications. To limit the amount of CPU cycles used for cryptography on the general purpose processing unit of the the host, and to reduce power consumption of the device during crypto operations, it is likely beneficial to offload the symmetric and asymmetric cryptographic algorithms to specialized silicon. When executed on hardware the common symmetric encryption algorithms (such as AES and 3DES) and asymmetric algorithms (such as RSA and ECDSA) are faster and more power-efficient (which is important in power-constrained and battery-powered devices) compared to software implementations.

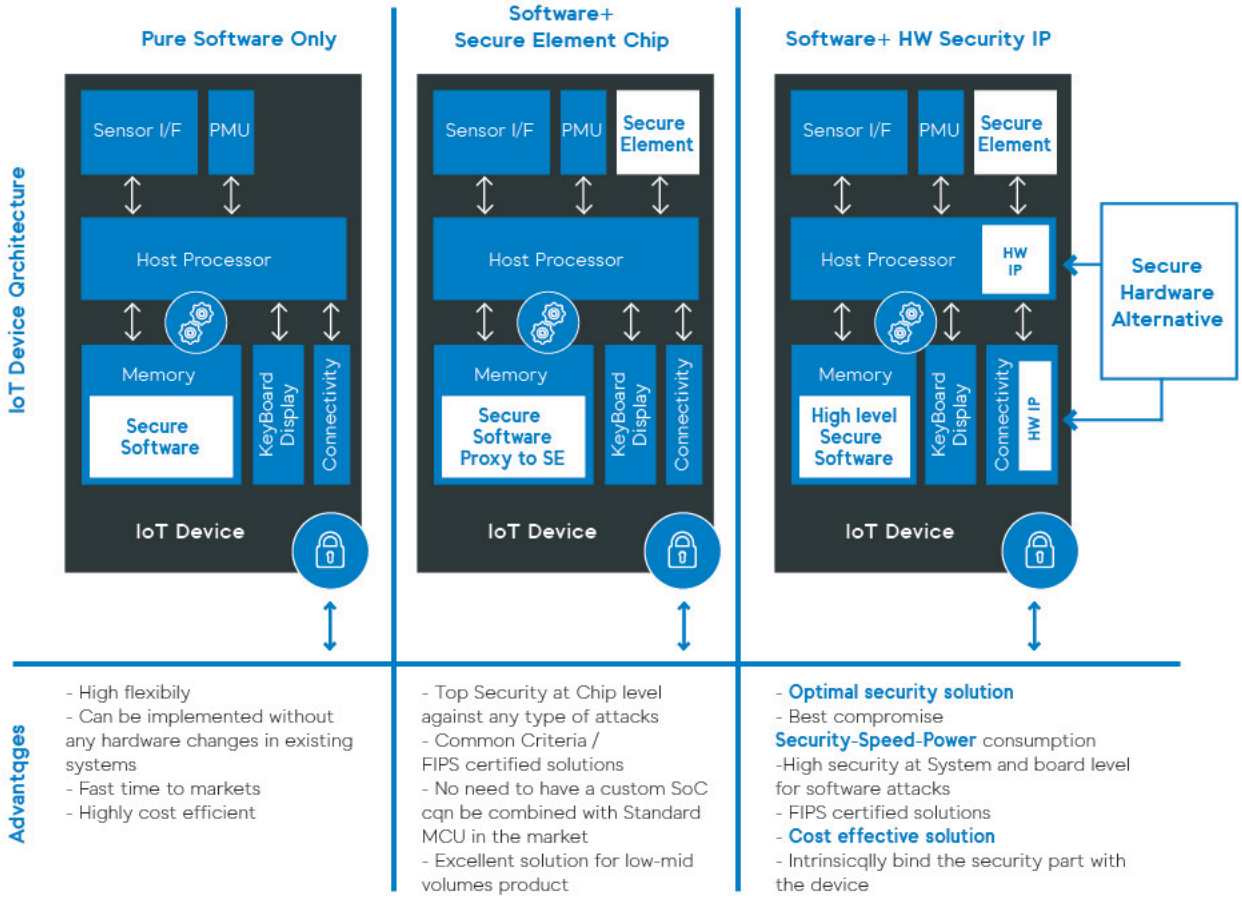
THIRD PILLAR : PROTECTING DATA IN PROCESS

The code executed by an IoT device must behave as expected: it should not be modified or corrupted and should not "leak" sensitive data. This is even more important when the code handles cryptographic keys or performs cryptographic algorithms. The key must remain secret and unchanged during the full cryptographic process, including loading of keys, and the code executing the crypto-function must behave as expected. Some sensitive applications such as payment applications in a wearable device may require some specific level of further protection.

The code can be protected in two ways: either execution happens within a secure or trusted environment or execution can be protected using software tools.

Secure or trusted execution environment can be provided by either Security chips or an isolated trusted or Secure environment built-in the IoT device host SoC/CPU.

The figure below provides a summary of different security implementations.



Solution-based on companion chip: Secure Element

Security Chips- also called "Secure Elements" or "Security Modules"- are usually derived from the SmartCard industry and can be inserted as a companion chip into IoT device architectures. They usually offer a very high level of security as they can be certified according to SmartCard industry standards like Common Criteria. This security implementation can also offer a good solution for medium size volume applications as it does not require any sepecfic Integrated Circuit development.

We can consider two types of security Module/Secure Element chips: The first type is only executing security functions (cryptography, data storage) within the secure environment, and are therefore optimized in terms of power consumption and cost. The second type offers additional computing power and memory and can execute a part of the device application within this secure environment. Financial transaction is a typical application that may require such implementation.

It has to be noted that as the crypto-functions are executed by an external component to the main CPU device, it is necessary to create a secure bridge between this secure element and the host CPU.

Solution based on hardware IP core

The second solution consists of briging security to the heart of the device by adding a security harware IP core into the SoC of the device. This solution offers an excellent option for high-volume devices in terms of cost, compared to Security Chip. It also offers the best solution when power constraint is a key issue in the IoT device and when crypto-functions are executed in a dedicated hardware block. In this case, the need to create a secure bridge between the host processor and the secure element automatically disappears.

High quality TRNG can only be obtained with an hardware solution[7].

Pure software solution

The third solution consists of protecting the code thanks to software mechanisms. The execution code can be processed through a software development tool to improve resistance against several types of attacks. For example, it provides protection against debuggers, memory dumpers, and reverse engineering which is referred to as "software obfuscation"; it also provides an additional layer of security by hiding, partitioning, and re-arranging data as it is handled by software.

Obviously these security mechanisms can be combined with the trusted or secure execution environment to offer additional protection.

Another important aspect of protecting data in execution, and to ensure that the device runs the intended software in the way its manufacturer or deploying organization

[7] The importance of True Random Number Generator in Cryptography (Inside Secure, 2015). Download on Inside Secure website

intended it to, is the “secure boot”.

A securely booted device is allowed to boot only software images or to accept data that is endorsed or signed by the manufacturer (or a trusted third party). In general terms, this can be realized by a hardware-assisted boot process in which the images are verified by hashes (data integrity check) and digital signatures prior to executing them at boot time. Implementing secure boot as a part of a device architecture is closely connected to other device management and maintenance tasks, and should be considered as a part of a larger security picture. The secure booting procedure needs to be compatible, for example, with remote and possibly “over the air” (OTA) device firmware upgrade mechanisms, to allow device software images to be altered post-deployment.

IoT devices will not be static. In most cases the software running on an IoT device wil need to upgrade (bug fixes, adding new features...). The management of software upgrade is an other important aspect and is critical when devices are installed in the field. Secure software update usually uses a combination of signature verification (software has been signed at the source by a trusted party, the device can verify the signature of the source thanks to the public key), and data integrity check to verify that the software code has not been modified between the source and the device. Hashing functions are in general used to perform data integity check. A hashing function, like SHA-256, generates a hash (a constant size “fingerprint”) value. A small change in the orginal data will change the hash value. Hashing functions work only in one way, meaning that the original data cannot be re-created from the Hash value. So the device can process a Hash on a message (or part of it) it has received and compare the Hash value sent along the hashed message by the issuer of the code.

FOURTH PILLAR: SECURE STORAGE - PROTECTING DATA AT REST

While data is protected during its manipulation, stored data must also be protected.

There are two main categories of data that must be protected during storage:

- Encryption key and unique device identifier. Such data is used as a trust anchor for a secure system. From this root key, derivative and session keys will be generated to authenticate and securely communicate between peers. If those

keys are known, a clone of the device can be created, and communication can be decrypted.

- Sensitive data that belongs to the application and must be protected to warranty its privacy.

In an IoT device this data is stored in variety of media, but essentially in memory devices offered by the semiconductor industry (OTP, ROM, RAM, Flash, EEPROM...). Data can be protected by encryption or by being stored in a tamper-resistant device, or a combination of both. In any case, the access to such data must be carefully controlled for authorized persons, machines or processes.

INSIDE SECURE PORTFOLIO OF SECURITY SOLUTIONS

Inside Secure's product portfolio contains security products such as software toolkits and hardware IP cores, but also provides services such as personalization and provisioning. The full range of Inside Secure's product and service portfolio is depicted in figure 3.

As a true veteran of the Internet security market and a leading security technology vendor, Inside Secure is in a unique position to offer a solution that can cover all security aspects of an IoT node. Inside Secure's solution components are mature, robust, and field-proven in some of the most demanding markets. This experience is readily applicable to the IoT market.

The methods, technologies, and architectures described in this overview document can be implemented with the offerings, expertise, and experiences of Inside Secure engineering. Products can be customized to fit specific customer use cases.

Security is the key business enabler and needs to be built into each IoT node from the start.

PURE SOFTWARE SOLUTIONS

Inside Secure's portfolio includes pure software solutions in cost-effective and easy to implement toolkits.

Software protection - Protecting data in process

Metaforic CoreTM

As an anti-tamper software toolkit, Metaforic CoreTM prevents code modification by linking code parts to each other and lettin an application defend itself if a change is detected.

Metaforic ConcealerTM

Metaforic ConcealerTM software toolkit provides obfuscation - a technique used to prevent reverse engineering by scrambling and confusing application code.

Metaforic WhiteBoxTM

Metaforic WhiteBoxTM software toolkit provides crypto-protection by always concealing crypto-functions or secrets in memory even while in use.

MatrixSSE

MatrixSSE combines Metaforic Core, Metaforic Concealer and Metaforic WhiteBox all in one to offer a powerful technique to detect hacked services: it detects a non-authorized version of an-OS on a specific device. Matrix SSE can be considered as a software Secure Element solution.

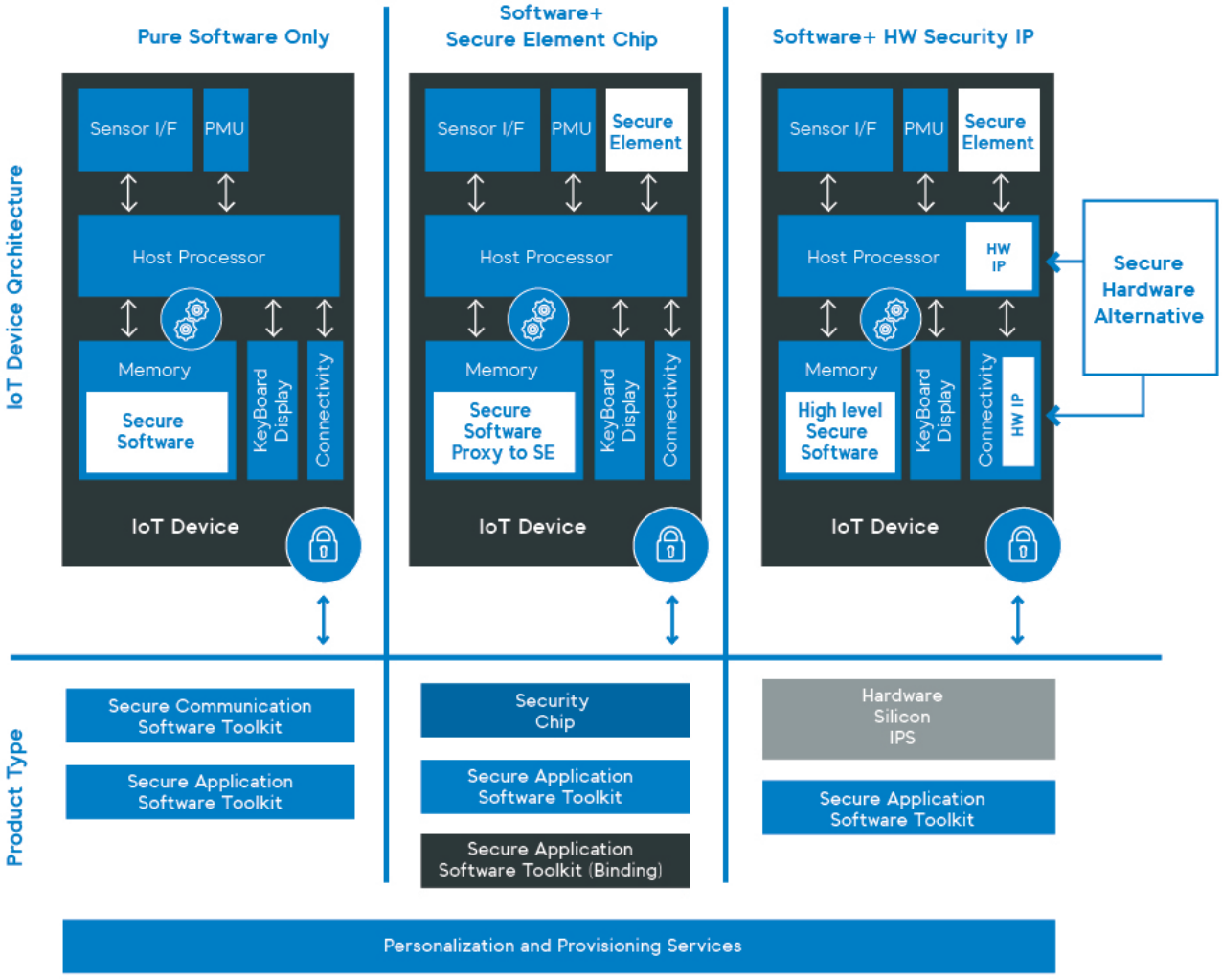
Secure communication

Inside Secure is a global leading secure communications protocol expert. It has been instrumental in specifying, standardizing, and implementing secure IP communications since the late 1990s, and has licensed secure protocols toolkits to over 100 licensees globally.

MatrixSSL

The Inside Secure MatrixSSL is a compact, low-footprint SSL library for embedded devices. For IoT nodes, MatrixSSL provides an extremely well optimized secure sockets layer (SSL) implementation that requires a fraction of the code and memory required by OpenSSL, the most common implementation of this protocol. MatrixSSL is the market leading SSL protocol implementation for resource-constrained environments, such as the IoT, MatrixSSL is an excellent fit.

MatrixSSL is interoperable with any standards-compliant SSL server, and has a modular design, so that customers have the freedom to deploy only the desired extensions to the base SSL standard. Since MatrixSSL is not based on OpenSSL, it is not vulnerable to the Heartbleed vulnerability.



QuickSec IPsec

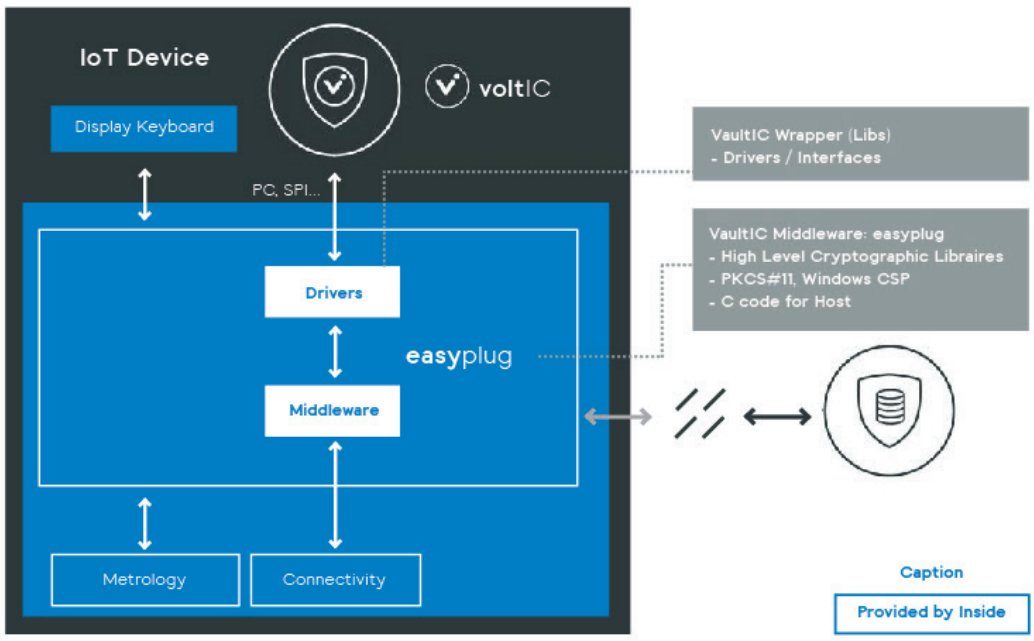
Inside Secure QuickSec IPsec toolkit is the leading IPsec and IKEv2 software implementation solution on the market today. This protocol is the foundation of a large number of network infrastructure vendor products. Versions of QuickSec IPsec and IKEv2 have been ported to embedded devices, firewalls, security gateways, and mobile devices.

In IoT devices that leverage the widely used Linux operating system, the QuickSec IPsec implementation provides a solid network security foundation to build on. Along with the IPsec protocol, the QuickSec toolkit provides industry-leading implementation of the Internet Key Exchange (version 2) protocol.

IPsec and IKE protocols are often thought of as being useful mostly in infrastructure-type deployments, and sometimes considered to be too heavy or complex for embedded deployments. This is somewhat of a misunderstanding, as the protocol standards themselves have no such restrictions. In fact, the standardization body (IETF®, Internet Engineering Task Force) has even published a guideline document on how to apply these standards in a minimal fashion for resource-constrained deployments.

VaultIC security module and VaultSEcure secure element

VaultIC, an integrated circuit with an embedded operating system and some security functions, is a secure element (SE) limited to cryptographic functions used as a companion chip in an IoT device. VaultIC provides a trusted, tamper-resistant and secure environment for executing security sensitive code, such as key generation, key handling, and en/decryption.



In an IoT node, VaultIC serves as the concrete hardware foundation upon which the security subsystem is built. A secure element is physically separated from the general purpose computing resources and peripherals, and exists only for the purpose of performing its pre-programmed security routines.

The VaultIC security module includes a rich set of pre-programmed cryptographic services and functions that are executed in a hardened environment.

VaultIC cryptographic services provide:

- The low-level cryptographic methods and algorithms needed for authentication and data encryption/decryption:
  - Symmetric encryption algorithms (DES, 3DES, AES)
  - Asymmetric encryption algorithms (RSA, ECC, DH)
  - High Quality True Random number generation
- The storage capabilities that allow hardware secured storage of essential data items (private keys, CA and user certificates, user credentials, configuration data).

Once the key material is generated, and possibly enrolled into a PKI with a protocol, the secure element protects the key material (secrets, private keys, entity and CA certificates) from compromise and modification. The access to keys is restricted, and the applications on the host node are provided access to the security functions over standardized APIs (for example the multi-platform PKCS#11) that provide applications with the functionality needed but never disclose the sensitive keys outside the secure confines of the Secure Element hardware environment.

Due to their nature as “mission critical”, some IoT use cases (automotive, medical, or smart energy) face stringent regulatory requirements. In these use cases, the quality of the implementations is typically demonstrated with public certifications. The VaultIC is highly tamper-resistant to avoid logical and physical attacks and cloning. It is certified to FIPS 140-2 Level 3 for the complete product, and the hardware alone is verified to Common Criteria EAL4+/5+. The VaultIC low-power consumption profile makes it a viable solution to meet the limited power budgets of the embedded IoT nodes.

The VaultSEcure secure element product inherits all functionalities from VaultIC but also allows user to embed customer-specific applications thanks to its JavaCard OS.

VaultIP in the SoC

VaultIP is a verilog RTL Hardware IP Core to be integrated into SoC design or an ASIC to provide secure, energy efficient and accelerated security functions. VaultIP provides a trusted and isolated (from main SoC CPU) environment for executing crypto-services, such as key generation, key handling, and en/decryption, as well as key storage services.

Combination of hardware crypto- and security functions

The VaultIP security module includes a rich set of cryptographic services and functions that are executed independently from the general purpose computing resources. For example:

- On-Chip Hardware Security to Support to Key management
  - Key storage in OTP, Key derivation,
- Symmetric and asymmetric crypto-algorithms (e.g. AES and ECC) for authentication, secure communication, secure boot...
- High Quality Entropy Source
  - High Quality (TRNG)
  - Required for secure key generation
- Access Control to Protected functionality
  - Secure debug activation
- Secure Timer
  - Locality checks / Protocol Timeouts
  - Login Timeouts

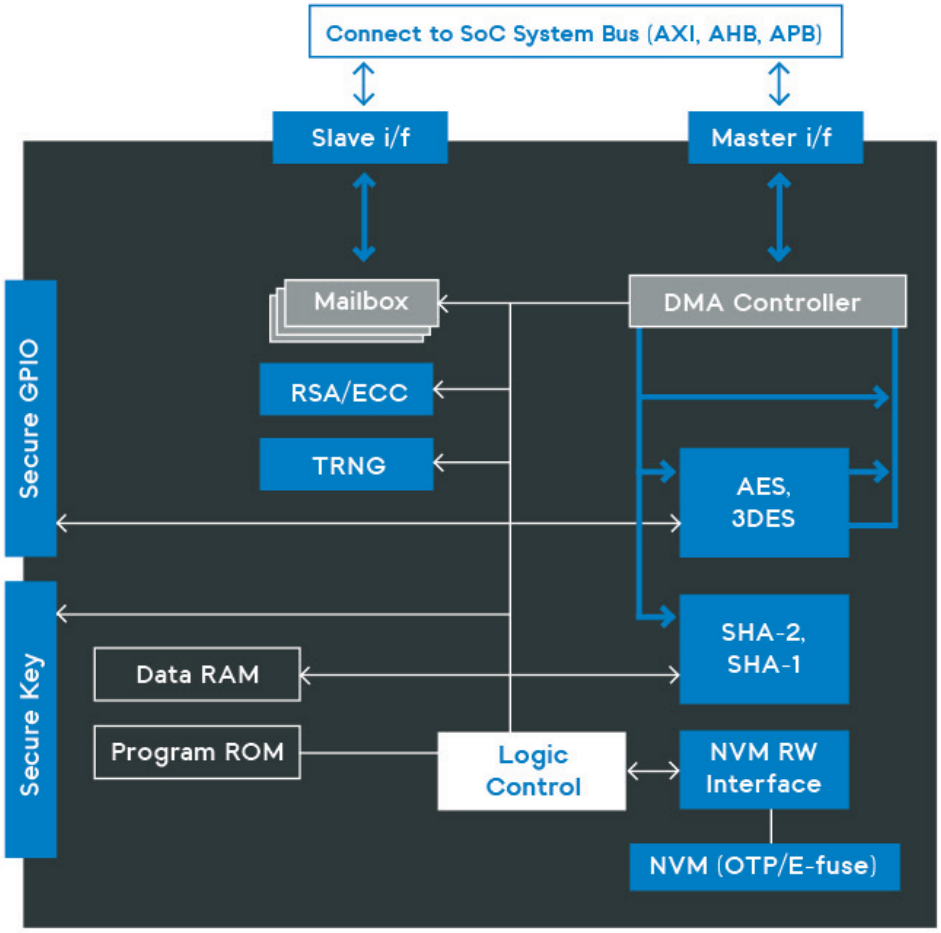
Family of VaultIP products

Please contact your nearest sales office to obtain additional information regarding VaultIP product family.

VaultIP-130 - Superset : Target devices based on ARM TrustZone that need authentication, confidentiality and integrity.

- Hardware device identity & authentication (symmetric or asymmetric)
- Hardware support for secure storage of (personalized) keys.
- ARM Trustzone support
- Full crypto-function to support, secure bootstrap, firmware update, secure storage
- Hardware supported secure communications (DMA/AES for SLL/TLS)
- FIPS140-2 L2 pre-certified (allows fast certification cycle)

As the security partner of a number of Tier 1 semiconductor manufacturers, Inside Secure has implemented secured boot architectures with some of the most demanding semiconductor manufacturers in the industry, and can leverage these experiences in implementing future architectures.



Personalization and provisioning

Please contact your nearest sales office to obtain additional information regarding VaultIP product family.

Summary of Inside Secure's IoT Solutions

With more than 25 years experience in digital security Inside Secure is the only player in the market that can offer the full spectrum of products meeting all security requirements. Inside Secure is a one-stop shop security solution provider, helping its customers to implement the right balance of security vs. cost and performance.

Inside Secure helps its customers to become more competitive: Inside Secure provides robust-proven software, security chip and silicon IPs products. This helps to reduce cost and save time for product launch, and to accelerate business growth. The fact that our prodcuts are proven reduces development time (e.g. Si Iteration for hardware), test, validation and security evaluation.

Inside Secure holds more than 1,000 patents, most of them related to digital security, and delivering solutions to numerous prestigious customers.

CONCLUSION

The Internet of Things is a new and emerging technology market, with a large scope of requirements and needs in terms of security. The security requirements are similar to those of earlier generations of networked devices, embedded devides or security tokenization but need to be combined together strategically to match each specific use case. The security design of an IoT node can greatly benefit from the experiences gathered in the fields of platform, network security, and highly sensitive applications such as payement or financial transaction, government related application, content protection.

The solution must be adapted to each specific use case.

REFERENCES

[1]	IPv6	http://www.rfc-editor.org/rfc/rfc2460.txt
[2]	VaultIC	http://www.insidesecure.com/Products/Embedded-Secure-Element
[3]	Sony RND compromise	http://www.exophase.com/20540/hackers-describe-ps3-security-as-epic-fail-gain-unrestricted-access/
[4]	PKCS#11	http://www.emc.com/emc-plus/rsa-labs/standards-initiatives/pkcs-11-cryptographic-token-interface-standard.htm
[5]	CMP	http://tools.ietf.org/html/rfc4210
[6]	MatrixSSL	http://www.peersec.com/matrixssl.html

[7]	OpenSSL Heartbleed	http://info.insidesecure.com/heartbleed
[8]	QuickSec IPSec	http://www.insidesecure.com/Products/Protocol-Security-Toolkits/QuickSec-IPsec-Client-Toolkit
[9]	Minimal IPsec and IKE – IETF	http://tools.ietf.org/html/draft-kivinen-ipsecme-ikev2-minimal-01

APPENDIX 1 – CRYPTOGRAPHY PRIMER

Applications of Cryptography

Confidentiality - Transform data into a form that hides and does not reveal any of the original form

Integrity - Ensure that data remains identical

Authentication - Identify the communication peer

Non-repudiation - Ensure that if an event happened between two parties, neither of the parties can deny that it happened.

Cryptographic Algorithms

- Symmetric Ciphers are typically used to provide Confidentiality.
  - e.g. AES, DES, 3DES
- Asymmetric Ciphers are typically used to provide Authentication and Key-wrapping.
  - e.g. RSA, ECC
- Stream Ciphers are typically used to provide Arbitrary Length Confidentiality.
  - e.g. RC4, Salsa20, SNOW
- Key Exchange algorithms are used to provide Authorization and for establishing common secrets.

  - e.g. Diffie-Hellman (DH), ECDH
- Cyptographic Digest is used to provide data Integrity
  - e.g. SHA, MD5
- Message Authentication Code or in other wors Keyed Cryptographic Digest is used for Authenticated Integrity.
  - e.g. CBC-MAC, HMAC, CMAC, GMAC

Cryptographic Key Sizes and Key Lifetime

Figure 5 - Asymmetric key strength relative to symmetric cryptography keys (source: NIST)

Symmetric	Asymmetric (RSA)	Asymmetric (ECC)
128 bits	3072 bits	256 bits
192 bits	7680 bits	384 bits
256 bits	15360 bits	521 bits

Figure 6 - What size key is required to protect information for its full lifecycle

Lifetime	Symmetric	Elliptic Curve	RSA	Example Application
Short	64 bits	128 bits	700 bits	Session key
< 10 years	80 bits	160 bits	1024 bits	Mobile phone
10+ years	128 bits	256 bits	3072 bits	Automotive

Elliptic Curve Diffie–Hellman (ECDH)

Anonymous key agreement protocol that allows two parties, each having an elliptic curve public–private key pair, to establish a shared secret over an insecure channel. Same principle applies as with DH, but using Elliptic Curve Cryptography. ECDH is a

NIST SP 800-56A recommended key establishment scheme

Authenticated Elliptic Curve Diffie–Hellman (ECDH) - Authentication of the key agreement provides protection against Man-in-the-Middle attack. Public / Private key pairs are used for signing (ECDSA - Elliptic Curve Digital Signature Algorithm) the key agreement messages.

True Random Number Generator (TRNG)

A TRNG that cannot be software or even hardware getting enough entropy (with acceptable die area) can be challenging. A truly random number stream is indistinguishable from noise, which means that all numbers in the value space have an equal probability of being generated. And that there is no correlation between the numbers generated by the TRNG.

Random numbers are used e.g. for Nonces (number used once), Key generation (symmetric and public), and Initialization vectors (IV) to initialize an encryption mode (e.g. AES-CBC) so that the first encrypted block is not identical for all identical plaintext.

Even the slightest linearity in RNG compromises its security.

«The generation of random numbers is too important to be left to chance» (Robert Coveyou, Oak Ridge National Laboratory)

APPENDIX 2 – GLOSSARY

3DES	Triple Data Encryption Standard
AES	Advanced Encryption Standard
AES-CBC	Cipher Block Chaining AES
APIs	Application Programming Interface
ASIC	Application Specific Integrated Circuit

CA	Certificate Authority
CBC-MAC	Cipher Block Chaining MAC
CMAC	cipher-based MAC
CPU	Central Processing Unit
DES	Data Encryption Standard
DH	Diffie-Hellman
DLTS	Datagram Transport Layer Security
DMA	Direct Memory Access
DSA	Digital Signature Algorithm
ECC	Elliptic curve cryptography
ECDH	Elliptic curve Diffie–Hellman
ECDSA	Elliptic curve digital signature algorithm
EEPROM	Electrically Erasable Programmable Read-Only Memory
FIPS	Federal Information Processing Standards
GMAC	Galois Message Authentication Code
HMAC	Keyed-Hash Message Authentication Code
IETF	Internet Engineering Task
IKE	Internet Key Exchange
IoT	Internet of Things
IP	Internet Protocol or Intellectual Property (if used in the Hardware IP Core)
IPsec	Internet Protocol Security
IT	Information technology
ITU-T	Telecommunication Standardization Sector of the International Telecommunication Union
KDF	Key Derivation Function

LTE	Long Term Evolution
MAC	Message Authentication Code
MD5	Message Digest 5
MIME	Multipurpose Internet Mail Extensions
OS	Operating System
OTA	Over-The-Air
OTP	One Time Programmable
RAM	Random Access Memory
RC4	Rivest Cipher 4
RNG	Random Number Generator
ROM	Read Only Memory
RSA	Rivest Shamir Adleman
RTL	Register Transfer Level
SHA	Secure Hash Algorithm
SNOW	Steganographic Nature Of Whitespace
SoC	System on Chip
SSH	Secure Shell
SSL	Secure Sockets Layer
TEE	Trusted Execution Environment
TRNG	True Random Number Generator
UTMS	Universal Mobile Telecommunications System
TLS	Transport Layer Security