Secure Memory Architectures in Emerging Electronic Systems

Introduction – Trends Driving a Need for Greater Security

Modern society is currently experiencing exponential growth of connected devices on public IP networks. In August 2018, IoT Analytics, a well-known market research firm, put the overall number of connected devices at 17 billion, and IoT (Internet of Things) devices at 7 billion. These numbers don't even include smartphones, tablets, laptops, or fixed line phones[1]. The range of targeted IoT devices could be anything from atmospheric sensors, remote payment systems, IP cameras, smart lighting systems, home routers, to connected vehicles. There are many forms of attacks that could be carried out on these devices. To give just one common example is the use of botnets to carry out DDOS (Distributed Denial of Service) attacks. The effects of this threat alone have been enormous in disabling major websites in the past. As more devices, vehicles, homes, and "things" become connected, the threat of attacks is inevitable. The challenge of protecting valuable assets from theft or maliciousness has forced many organizations to devise secure system architectures, some of which are described below.

This paper will discuss these architectures from the standpoint of memory because it is what is commonly targeted in attacks, either directly or indirectly. It could be by installing malware by modifying software stored in memory, or installing a sniffer device that modifies software or data in transit between memory and a host. The attack could also be the stealing of sensitive data ranging from personal financial data, to data that could be used to bring down the defenses of a device.

What follows is a top down description to familiarize the reader with high level security concepts, standards, and common threats along with matching remedies. Current security frameworks and features being implemented to protect data at rest and in transit will be described. The paper will end with application use cases to help the reader translate all these concepts and information into actual applications.

General Attack Types and Protection Approaches

There are three major objectives to achieving secure information in connected devices:

- Integrity -- ensuring that information is authentic and hasn't compromised. In other words, providing a security feature called non-repudiation that provides proof of authenticity and origin.
- Confidentiality -- securing information so that it is private and cannot be accessed by or made available to unauthorized users.
- Availability – making sure that information is accessible when it is needed.

The best place to start in understanding device security would be to take a high-level view of it. There are general categories of attacks that are useful in understanding the threat landscape, as well as common approaches to protecting against these threats.

When designing a product, the potential attacker's motivations need to be considered. Is it IP theft, Denial of Service (DOS), financial theft, or some other type of data theft? Next, the would-be attackers need to be understood. Are they government-funded, competitors, or just weekend hackers? After analyzing the attacks that seem likely, are they invasive, semi-invasive or non-invasive? Lastly what kind of attack methods might be used and how can a secure design thwart the attack or make it too expensive to hack. In short, a thorough security analysis needs to be conducted up front.

Table 1: Major Attack Surfaces, Vulnerabilities, Protection Approaches summarizes some of the major attack surfaces, their vulnerabilities, and protection approaches against likely attacks, which an advanced secure Flash device like Macronix ArmorFlash protects against.

Table 1: Major Attack Surfaces, Vulnerabilities, Protection Approaches

Attack Surface	Vulnerabilities	Protection Approaches
Communication Channels	• Man in the middle • Snooping, injection • Weak entropy sources • Clear text	• Use of sequenced frames • Strong random number generators • Use of encryption and authentication
Physical	• Side-channel analysis o Timing, power, EM emissions, acoustics • Fault injection (power and clock glitches) • Data remanence in memory • Brute force: probing, de-capsulation, reverse engineering, uncovering special debug modes	• Fault Injection - redundancy and fault tolerant computing • Hiding and scrambling the bus • Data encryption and de-encryption in a trusted zone • Sensor mesh in top metal layer • Fault detection -memory erase • Other anti-tampering designs
Software	• Fault injection • Version rollback • Interrupts • Buffer overflows • Malware injection • Code in clear text form over non-encrypted channels	• Fault injection o Execution Redundancy o Checksums on data transfers o Randomized execution • Encryption/Decryption • Creation of trusted execution zones • Designs based on compartmentalization and isolation, and access control • Root of Trust and Chain of Trust implementations
Lifecycle	• Unsecure memory provisioning environments • Code and data programmed and stored in clear text • Provisioning and re-provisioning in the field • Root-key discovery and exploitation	• Secure provisioning environments or services • Root of Trust and Chain of Trust implementations • Encryption and authentication • PKI- public/private keys, certificates, authentication • In the field secure secret key sharing

Standards, Frameworks & Platforms Driving Today's Security Implementations

Security is a broad subject with many competing specifications and organizations pushing standard implementations. There are a wide range of security standards that differ based on particular technologies, applications, markets, and level of required security. For a designer that is given requirements to build a secure device, taking into consideration all these factors are a daunting task. Where does one start and make sense of it all?

There are specifications, guidelines and recommendations from governmental agencies. The best known are NIST (National Institute for Standards and Technology), FIPS 140 (Federal Information Processing Standards) in the U.S., and BSI (Federal Office for Information Security) in Germany. But there are many others, like IETF (Internet Engineering Task Force) and ISO (International Organization for Standardization). All of these agencies and organizations are focused on general implementations or are targeted towards a particular technology, application, or market implementation.

There are organizations and industry consortiums that are focused on platforms. They are usually targeted towards a general area or operating environment. Some well-known organizations are the Linux Foundation, TCG (Trusted Computing Group) for secure modules, AUTOSAR (Automotive Open System Architecture), SIG (Automotive Special Interest Group)/ Automotive SPICE, and GlobalPlatform for secure-chip technologies. But there are many others and which ones that are leveraged will depend on what markets and applications the product is targeted toward.

Then there are security frameworks that are driven by IP and component suppliers. The most well-known and influential processor IP provider is ARM Holdings. ARM has produced a security framework called the ARM Platform Security Architecture (PSA), along with associated IP, design tools and software systems that are rapidly being adopted. Terms like TrustZone are well-known by most embedded designers. They are single-handily driving de-facto standard security implementations for the embedded industry. But it is important to note that their security implementations are based on recommendations from NIST, BSI, ISO, IETF, and others. Another important security implementation to mention is the Unified Extensible Firmware Interface (UEFI) for which the Secure Boot standard is defined. This has been driven for years by Intel, Microsoft and PC makers for securing BIOS firmware and had now found its way into embedded designs.

Lastly, there are secure frameworks and secure provisioning processes that target distributed networked computing systems, which is one of the foundations for securing IoT devices. These frameworks and processes are being driven by Cloud vendors supplying web management services, such as Amazon (AWS), Microsoft (Azure) and Google (Google Cloud Platform).

Modern Security Frameworks and Architectures

Several security features, frameworks, and architectures are worth mentioning because they are good examples that have been successfully implemented in todays embedded systems that are commonly found. What follows in this section is an expansion on these areas and where secure memory plays a part.

TPM – Trusted Platform Module

The foundation of security is to develop the entire system so that it is a trustworthy entity. Making a device safe from malicious attacks has consequences for system design. The hardware and software present in the device must work closely to achieve comprehensive, layered, security features. TPM (Trusted Platform Module) was conceived by the TCG and is a well-known international standard. TPM has been addressing the trust issue and related security benefits for PCs, servers, networking gear and even embedded systems for decades. For

resources constrained IoT devices, TCG developed DICE architecture (Device Identifier Composition Engine) with minimal silicon requirements to achieve enhanced security and privacy. Even simple silicon capabilities combined with software techniques can realize enhanced security by implementing a strong device identity, attesting software and safely deploying updates.

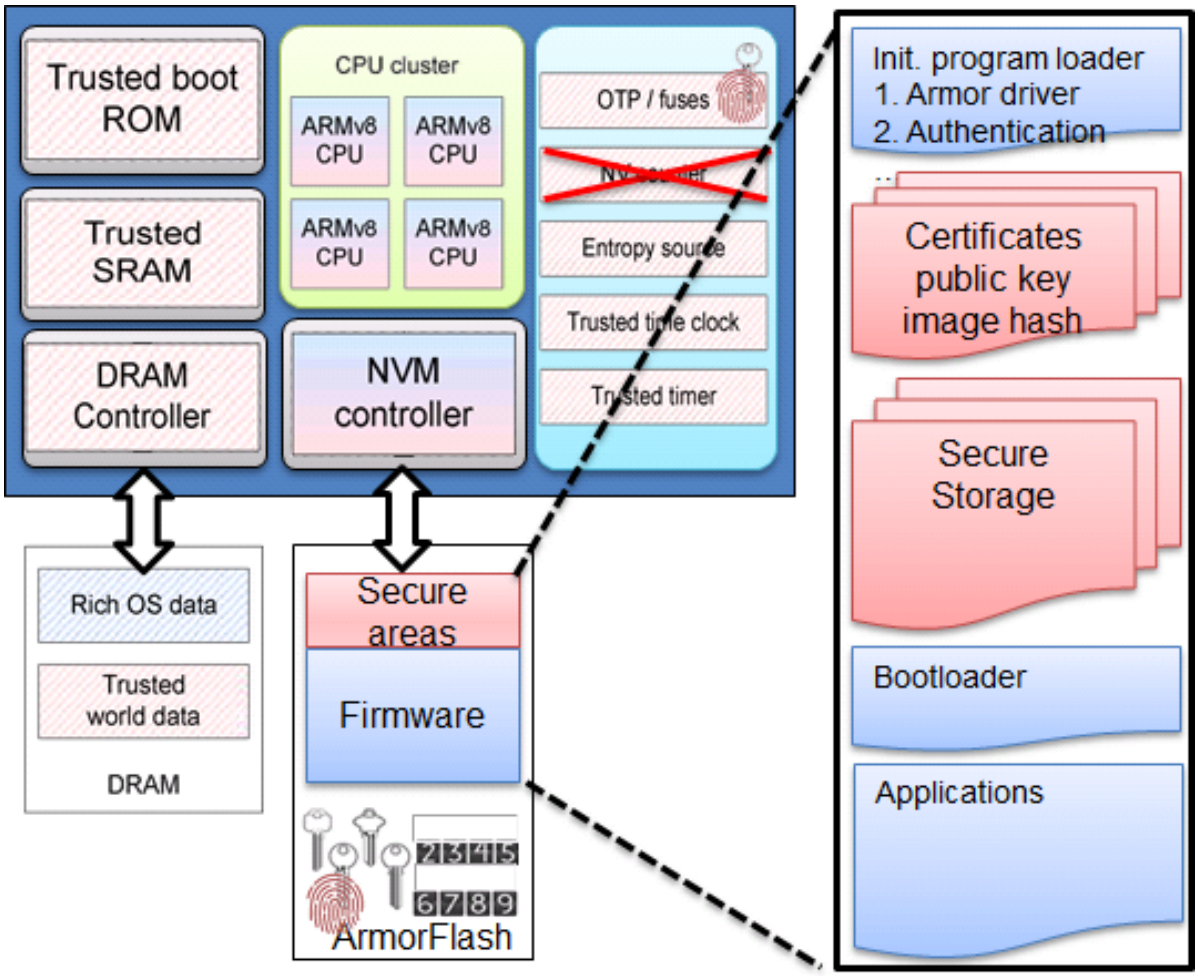
ARM TrustZone

ARM unveiled the TrustZone technology aimed at establishing trust in ARM-based architectures during the same period TPMs were being implemented on x86 platforms. Unlike the fixed-function TPM device, TrustZone privileges certain portions of hardware and software to construct two orthogonal worlds; a secure “trusted” zone for the security subsystem and a “non-trusted” zone for everything else. The trusted zone provides a secure environment called a Trusted Execution Environment (TEE). The TrustZone design extends from software to all hardware access points. For example, a TrustZone-enabled AMBA AXI bus fabric ensures that trusted resources cannot be accessed by non-trusted access requests. These two zones share the same physical processor core in a time-sliced fashion and do context switching and arbitration either through a software-only implementation for microprocessors or through a combination of software and hardware for smaller microcontrollers. TrustZone removes the need for a dedicated security processor core and further saves system cost and power by allowing high-performance security software to run alongside a non-trusted operating environment.

Secure Boot

Along with TrustZone, trusted software is required. During the boot sequence, a series of software images are digitally signed with keys generated based on the value of the previous image. This mechanism creates a "Chain of Trust” (CoT) during this booting process on the assumption of a known ROM starting point. This starting point is code that is booted from an on-chip tamper-proof ROM, accompanied with a root cryptographic key. This is used to verify the next-stage software fetched from a separate non-volatile memory before handing over the control to it. The key components in the CoT are either a key or ROM images. Certificates following the X.509 standard are created and issued for the authentication of all software images in a sequenced fashion. This includes trusted and non-trusted applications. The complete ROM image is guaranteed by the vendor by using secure provisioning facilities and processes. Refer to *Figure 1: Secure boot*.

Figure 1: Secure boot



Trusted Execution Environment: Secure Storage in TEE

ARM TrustZone and GlobalPlatform (*Figure 2: GlobalPlatform Architecture*) use the TEE (Trusted Execution Environment) concept in their architectural specifications. The TEE is a hardware-isolated or compartmentalized secure environment where the trusted applications can be executed. The TEE offers isolated, safe execution of authorized software and provides end-to-end security. GlobalPlatform's TEE Internal Core API specification mandates that it be possible to store general-purpose data and key material that guarantees confidentiality, integrity, atomic operations when modifying memory contents. Secure memory may also be partitioned to support multi-tenancy architectures popular in cloud computing. Multi-tenancy supports different users within a single application and operating system. Each one of these users has credentials defining their degree of access to the system along with application specific data. These credentials and sensitive application data can only be read, modified or deleted from the TEE.

Secure Storage Support for TEEs and TPMs

In both TEE and TPM, there is a requirement for securely storing data that is partitioned by applications and users. This is particularly true if the OS architecture is comprised of virtual machines or is a multi-tenet implementation. GlobalPlatform has specified requirements for trusted/secure storage in the TEE Internal Core API Specification. What follows is an excerpt from the specification that relates to secure storage[2]:

- The Trusted Storage may be backed by non-secure resources as long as suitable cryptographic protection is applied, which MUST be as strong as the means used to protect the TEE code and data itself.
- Trusted Storage MUST be bound to a particular device, which means that it MUST be accessible or modifiable only by authorized TAs (Trusted Applications) running in the same TEE and on the same device as when the data was created.
- Ability to hide sensitive key material from the TA itself.
- Each TA has access to its own storage space that is shared among all the instances of that TA but separated from the other TAs.
- The Trusted Storage must provide a minimum level of protection against rollback attacks.

Discrete non-volatile devices such as eMMC, UFI, and PCIe SSD devices are commonly leveraged as secure storage for TA's. They utilize the Replay Protected Memory Block (RPMB)[3] for secure transmissions between host and device. RPMB requires authenticated reads and writes for accessing data in a RPMB partition. A monotonic "write counter" and generated "random number" is embedded in the authenticate algorithm and used to protect devices from replay attacks for write and read operations. *Figure 2: GlobalPlatform Architecture* below shows the hardware architecture. The REE (Rich Execution Environment has restricted access to untrusted resources but not the trusted ones. The REE is only allowed to access the trusted resource through services exposed by the TEE client interface.

Figure 2: GlobalPlatform Architecture

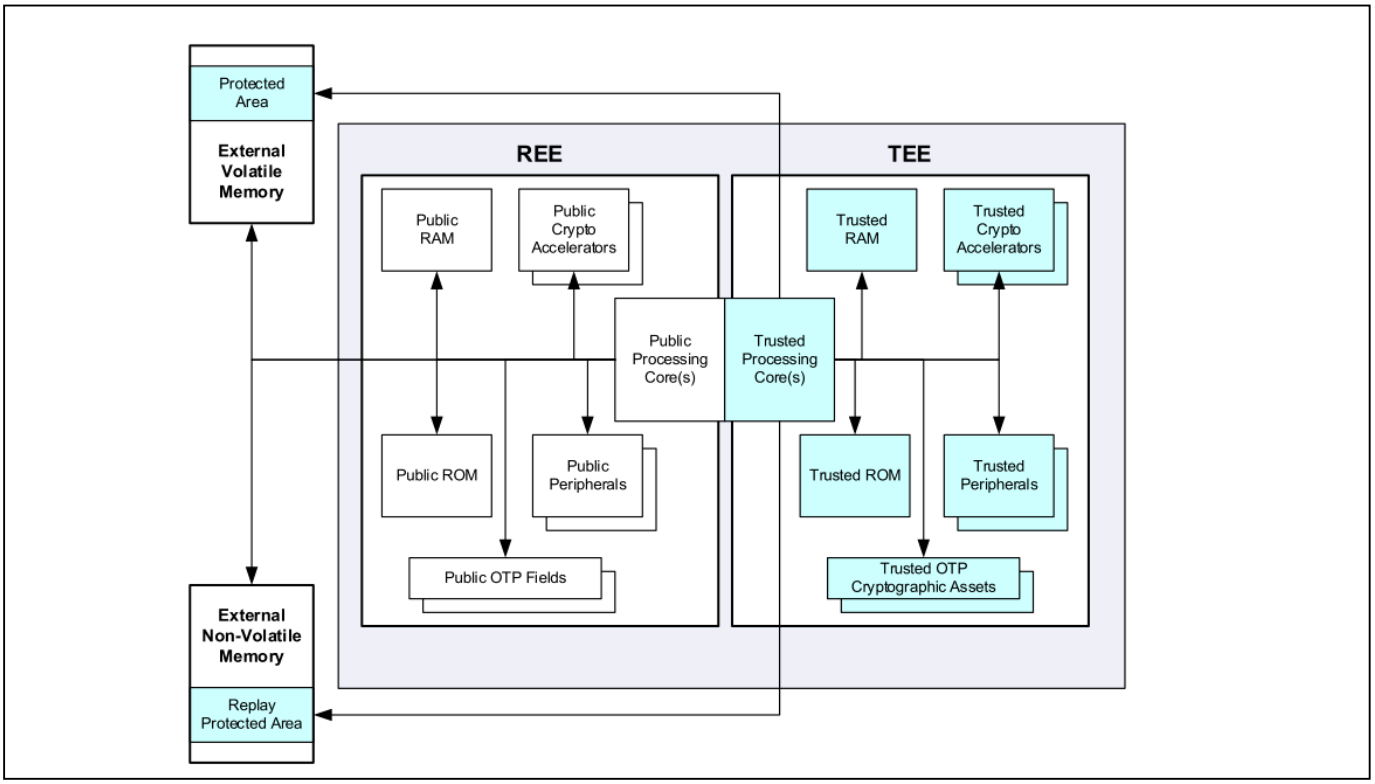
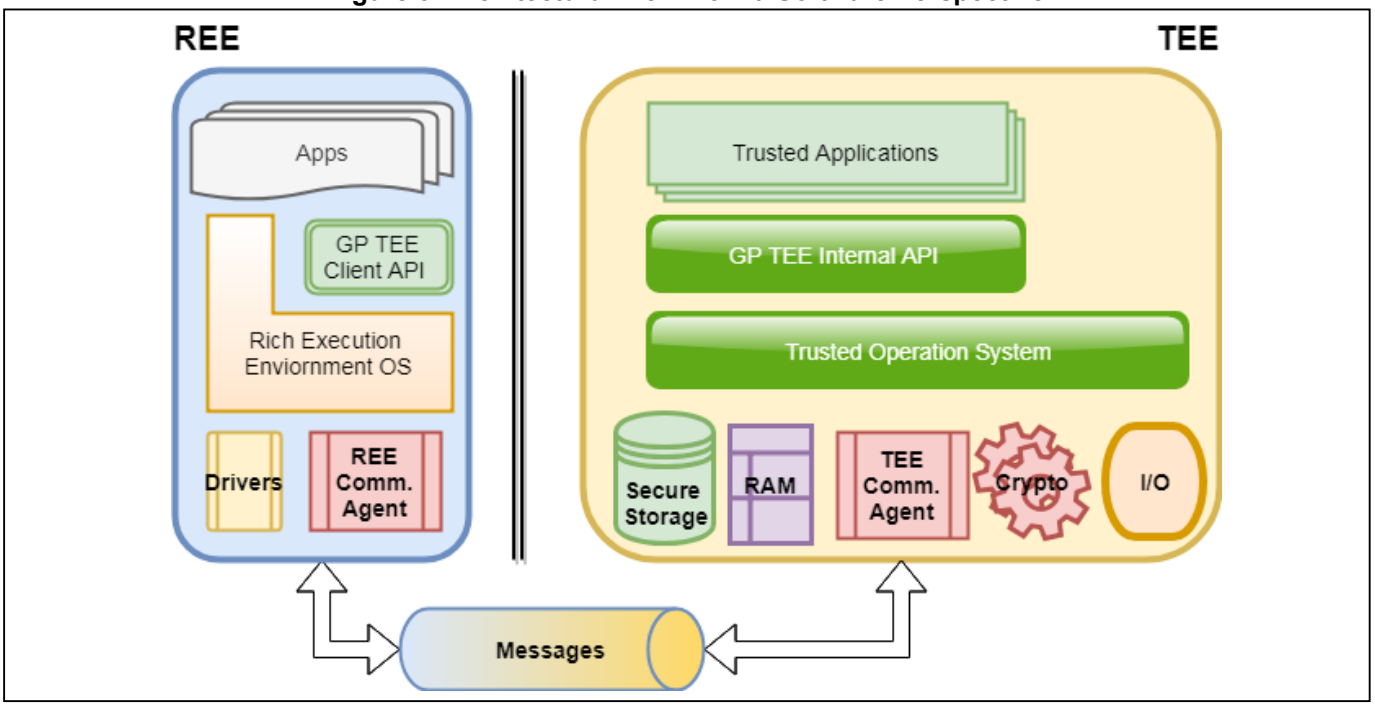


Figure 3: Architectural View from a Software Perspective shows architectural view from a software perspective. The TEE Client API is a low-level communications interface designed to enable applications in the REE to access and exchange data with a TA running inside TEE through communication agents.

Figure 3: Architectural View from a Software Perspective



Secure Flash devices like Macronix ArmorFlash provide all the security that is needed by a TEE or TPM. For example, in addition to the replay and authentication related security features that RPMB provides, ArmorFlash offers data-encryption capability to further protect the confidentiality of communications. By enabling the Physically Unclonable Function (PUF) feature, it makes it possible to uniquely bind a memory device to a particular host. Devices like ArmorFlash also provide 16 secure data regions with four independent keys providing 16 combinations of key pairs. This can be used to control access to each secure data region or user space to support hypervisors or multi-tenant implementations.

Secure Communication Channels and Data Buses

Communication channels and data buses provide prime opportunities for sniffing or man-in-middle attacks. Fortunately there are well known and secure protocols to thwart these sorts of attacks. For communications between a host and server, there are several worth mentioning that provide authentication and encryption features. TLS (Transport Layer Security) is a widely used cryptographic communications protocol standard. It secures traffic among devices with encryption, based on a public key exchange (PKE) framework. TLS is a layer on top of the TCP/IP software stack. A full TLS implementation requires memory and processing resources that most resource constrained devices lack. Therefore, for smaller IoT devices, a lightweight machine-to-machine connectivity protocol called MQTT is becoming popular. It minimizes the resource requirements to enable secure communication. The MQTT-SN variant can be implemented for the most resource constrained nodes while providing a form of TLS support.

Authentication, encryption, data scrambling, and frame sequencing with monotonic counters are the primary means of protecting data buses between a host and memory device. This will be discussed in more detail further down.

To conclude, Macronix ArmorFlash, is an advanced secure Flash that has been designed to interoperate and support these security features, frameworks, and architectures to effectively thwart attacks.

Security Features in Non-Volatile Memory

It is always assumed that the details of the cryptographic algorithms are well known by an attacker so it is only the secrecy of the key that ultimately provides security. Trying to keep keys secret is one of the toughest challenges in deploying security. Beside the keys, other user-sensitive or private data are usually stored at the same safety level as cryptographic keys. In some secure execution environments, a dedicated secure storage is required not only for secret key information, but also for application-specific data. Also, for modern OS (operating system) architectures that support virtualization or multi-tenancy, it is a requirement to support and store the security credentials of multiple users for multiple applications. For complex systems, this secret information will be stored in external secure flash memories due to the storage size requirements.

Table 2: Standard Security Features in Non-Volatile Memory below shows a wide range of standard security features that are common in non-volatile memory being offered in the market. They are effective in protecting various threats to its contents. Why would a product need to implement memory with advanced security features? To start with, there is an overabundance of existing designs based on older microprocessors. So if there is a requirement to make the product more secure, it is expensive to do major redesigns of existing products, particularly with complex applications in a hard, real-time environment that have been field-proven. If a requirement has come from an organization's marketing department that it must now be connected to an open network, where will the required keys and credentials be securely stored to support secure access?

On-chip embedded non-volatile (Flash) memory is considered relatively safe if tamper resistance has been implemented. But increasingly, general microprocessor suppliers are shrinking the embedded non-volatile memory footprint to only store Phase 1 boot code because as technology nodes shrink from, say, 55nm to 28nm, the footprint for embedded flash is much larger compared to the CMOS-based processor and other peripheral logic that has be shrunk. This is because embedded flash doesn't scale like CMOS-based logic (40nm stopping point) and so expected overall die size and costs reductions fall short. Also, customers have varying density requirements, so betting on any given customer's memory footprint requirement is risky. Lastly, embedded flash adds cost to the product that could be shifted to the customer. Lastly other forms of emerging embedded flash technologies, such as MRAM, are more expensive than discrete Flash memory.

There are 3 major advanced security memory products available today; RPMC, Authentication Flash, and a more full featured secure flash like Macronix ArmorFlash. RPMC is a memory device that is sole purpose is to provide non-volatile monotonic counters for the host to support sequenced frames in secure communication protocols like TLS/SSL to thwart man in the middle attacks. Several memory providers currently offer this type of flash as a dual purpose memory. This means there is a region in Flash memory reserved for RPMC functionality, with the rest reserved for normal NOR Flash. Authentication Flash devices only perform authentication with the host before a secure operation. A good example of this would be the RPMB feature in eMMC5.1. Full featured secure flash devices perform authentication and encryption along with a full range of additional security features.

Table 2: Standard Security Features in Non-Volatile Memory

	Security Features	NOR Flash	NAND Flash	e.MMC	ArmorFlash
Hardware	BGA package: The ball grid array under the chip protects against probing.	v	v	v	v
	Protection pin: The block protect operation using the protection pin can protect the whole chip or selected blocks from erasing or programming.		v		
	Hardware write-protect pin: There are 2 versions depends on the Flash type: NOR: It protects the register settings that configure the program/erase protection of blocks and sectors. NAND: The memory will not accept the program/erase operation. It is recommended to keep "write protect" active during power on/off sequence. It can avoid noisy environment makes data damaged.	v	v		v
Software Protection	Temporary block protection: This avoids accidental program/erase to specific blocks.	v	v	v	v
	Solid protection: This permanent block forbids malicious modification to block-protection configurations	v	v	v	v
	Unique ID: This is a value that is unique to each non-volatile memory device	v	v	v	v
	Password protect block locking: This feature uses an advance method to protect block locking configuration from modification.	v		v	
	Read protection: Protecting against data corruption	v			
	Sanitize: All data are physically erased to prevent data reuse.	v	v	v	v

A good example of a secure flash with advanced security features is the Macronix ArmorFlash. It is a NOR Flash device and has all the standard security features listed in *Table 2: Standard Security Features in Non-Volatile Memory* as well as advanced security features listed in *Table 3: Advanced Security Features*. ArmorFlash has a standard SPI interface and leverages the SPI command infrastructure. It introduces special packet operations that provide secure read and write operations to its secure memory region for data storage. The memory layout shown in *Figure 4: Example of how ArmorFlash could be configured* below is one example of how ArmorFlash could be configured. It is has a secure memory region that it is further partitioned into 16 independent 2 Mbit units for data storage. Another standard memory area of 224 Mbits for code storage is accessed directly with regular SPI Flash commands after an initial gated authenticated exchange.

Table 3: Advanced Security Features

Security Features	NOR Flash	NAND Flash	e.MMC	ArmorFlash
One-time programming: Protect configuration setting from others	v	v	v	v
OTP space: A space for OTP data	v	v		v
RPMB: Authenticated Access			v	
Replay Protected Monotonic Counter (RPMC): monotonic counter support	v			v
Secure region: Authenticated access				v
Independent areas for encryption/decryption: Support for multiple users				v
AES for encryption/decryption				v
True Random Number Generator (TRNG)				v
Key Generation				v
PUF Code– Hardware feature that provides a unique sequence of values based on intrinsic process variations during the manufacturing				v

ArmorFlash has the following security elements:

Non-volatile monotonic counter
The use of monotonic counters is the best weapon against replay attacks, even if power outages or glitches occur. This is because monotonic counter values are stored in non-volatile memory. ArmorFlash has 4 independent 32-bit monotonic counters, which only respond to authenticated operations. Counter configurations can be programmed to change counter behavior depending on different application scenarios.

TRNG
High entropy random numbers generators (RNG) are fundamental to almost all secure systems because cryptographic systems depend on secret data that is only known to authorized users. There are two main uses for RNG. The first use is in producing a random value called a nonce. It is used in calculating a unique MAC challenge-response authentication value. In this case, the client initially receives a random challenge, usually a message embedded with a nonce value from the host. The device uses this value as one of the inputs into a MAC calculation. The nonce value could also be provided by the memory device.

The main use is in generating encryption keys. In some implementations, a dynamic/session symmetric key is generated from random number. The TRNG in ArmorFlash has spatial and temporal dependence that produces a TRNG with high levels of entropy.

Key storage and management
There is a specific memory region in ArmorFlash capable of storing four 256-bit keys. Each key is setup through an independent and lockable process. Several key management commands are available to set up keys according to their intended use cases. The configuration allows for separate keys for authentication and encryption operations.

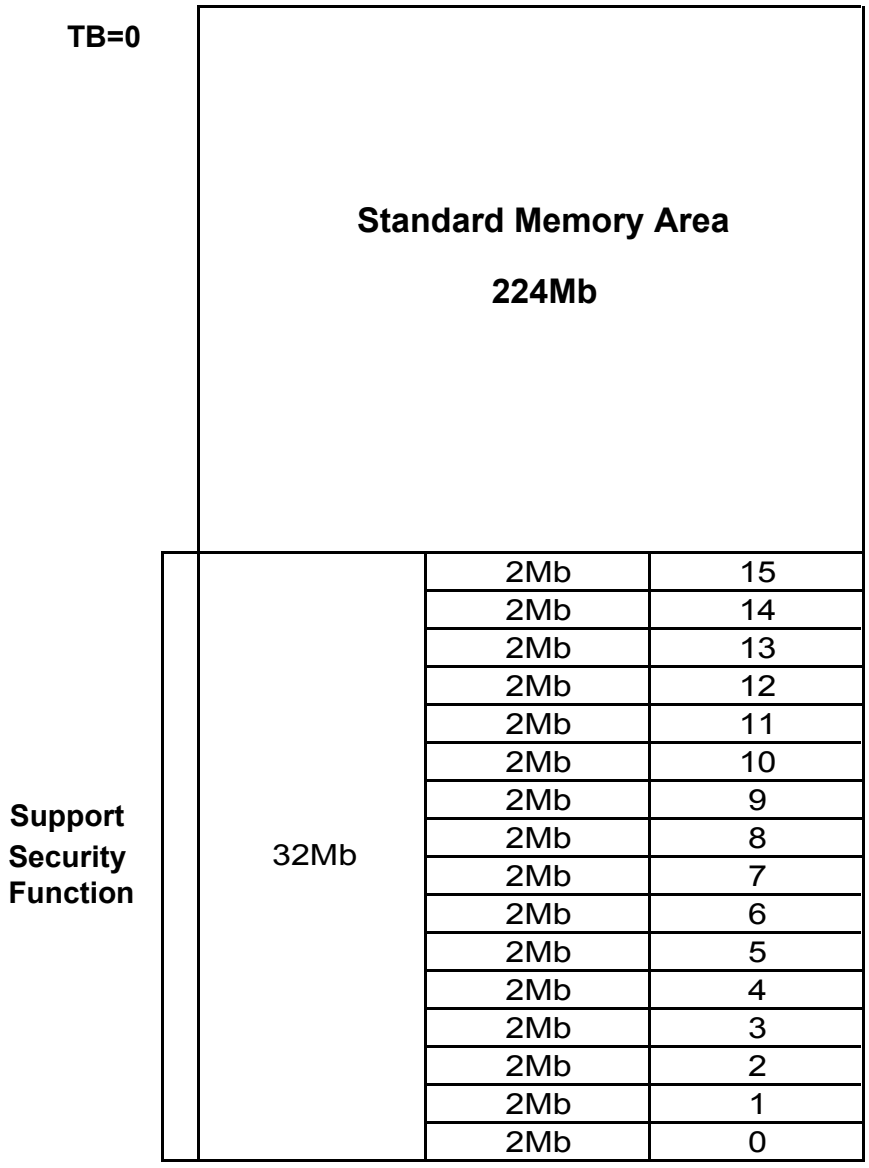
Authentication
ArmorFlash introduces a packet read/write command for various security operations. The packet command is a proprietary link layer packet protocol that is similar to the RPMB protocol. It has its own sub-command operation codes and data structures. The design for the layer above the link layer involves a shared symmetric key used for the authentication process and a Cipher Block Chaining Message Authentication Code (CBC-MAC), which is used to produce the required MAC values. CBC-MAC is implemented as part of a block cipher algorithm called CCM (Counter with Cipher Block Chaining-Message Authentication Code). It is a mode of operation of a block cipher algorithm and it integrates both authentication of the host and device, and encryption of the payload. It is a NIST standard (NIST Publication 800-38C).

Data Encryption
As part of the AES-CCM algorithm, the data field in the packet contains the encrypted payload based on a symmetric key block cipher algorithm with a minimum block size of 128 bits.

Unique ID and extra serial number
An additional 8K-bit secure one-time program (OTP) area is added for storing unique identifiers and static data according to system applications' demand. The 8K-bit secured OTP area further decomposes into two rows of 4K-bit configurations. ArmorFlash also incorporate an extra 8-byte serial number, which is factory-coded, and used in cryptographic calculations to uniquely bond the host and secure flash.

Non-volatile PUF
Although programmed unique ID or serial numbers are okay for identification, a PUF code can be used that is truly unique to each memory device. PUF is becoming common as a unique identifier or digital fingerprint for semiconductor devices. ArmorFlash takes advantage of random and physical variations of a Flash memory array to produce a non-volatile PUF code. Users can leverage this code both for identification purposes and as an input for key generation.

Figure 4: Example of how ArmorFlash could be configured



Use Cases

All embedded devices could benefit from standard security features in non-volatile memory, mentioned above, and there are a number of markets and applications where advanced features are required.

- Connected Vehicles – commercial and consumer: ADAS level control, premium content authentication, and electronic payment, user authentication, maintenance authorization
- Transportation - Secure logistics and operations support
- Industrial IoT (Industrial 4.0)– Edge servers and other gateway devices
- Enterprise & Data Center – (BMC) Board Management Controller Support
- Energy - Power Grid Infrastructure gateway equipment
- Cellular Network Gateways
- Products requiring premium content protection

In the case of Armor Flash, a single non-volatile memory can be implemented to protect sensitive data and also provide storage for code. It can be a substitute for a simple secure element, storing certificates, keys, and other credentials. The following summarizes common use cases for advanced secure flash devices.

- Storage of highly sensitive information such as keys, credentials, certificates, DRM (Digital Rights Management), content service keys (e.g. audio/video streaming), secure logs, and other PII (Personally Identifiable Information) like biometrics, passwords, contacts, etc.
- Designs that require a secure data memory and a memory for code storage
- Older designs that need to be upgraded for security reasons without changing the microprocessor or microcontroller. Or microprocessors that do not have secure embedded memory
- Designs that utilize a secure element could combine non-volatile memory and a secure element, lowering BOM costs
- Memory supporting OS architectures that have multiple users, such as multi-tenancy and hypervisors that require multiple sets of credentials
- Support for secure provisioning in unsecure manufacturing environments and re-provisioning in the field
- Protection of firmware rollback and cloning
- Memory that is protected against host/device ease-dropping and memory device tampering
- Support for Root of Trust (RoT) requirements through unique ID, authentication, and encrypted links

The following sections will focus on two application use cases.

Automotive – Consumer and Commercial

It is common knowledge that automobiles are getting more complex. Competition among carmakers for buyers is intense and their expectations of increased functionality are high. Similar to the anticipation of the next version of the iPhone, we are accustomed to annual technological improvements that make our driving experience easier and more enjoyable. You only have to look at current autonomous vehicles on the market to see where the evolution of the automobile is taking us. They are essentially mobile platforms to safely get us to where we want to go -- and keep us connected and entertained along the way. Drivers and the insurance industry want the car to act as a co-pilot to aid us in the driving exercise. All this is a lot to ask of carmakers and highlights some of the reasons why complexity continues to grow.

The amount of advanced electronics in autos is growing exponentially. The main reasons for these increases are the growing inclusion of driver-assistance features that include adaptive cruise control, cameras, object identification and notification/crash avoidance. The emission and powertrain control systems are more complex to manage hybrid propulsion. Clusters are being converted from mechanical to electronic displays along with the addition of heads-up displays, or HUDs. Infotainment options and connections to onboard LCD monitors are growing, too. Lighting and other environmental controls are increasing and becoming more automated. Take, for example, a rainfall-detection system that automatically activates the car's windshield wipers when it senses precipitation on the windshield. And then there are requirements for connectivity to the Internet to download mapping and positioning information, entertainment content (video and audio), and manage software updates for onboard computer systems. Also, security systems to authenticate authorized drivers are getting more complex. Lastly, recording systems are being added to capture the last seconds before and after an accident to capture driving patterns, aiding law enforcement and insurance companies.

A phenomenal growth in data has accompanied these new features, and with it a need to store it. The data come in multiple forms: at rest, in use and in motion. Most data are in use or in transit, and a small percentage that is stored should be done so securely. However there are different levels of required security. Logging or calibration data is typically the least secure. The most secure is in the form of personally identifying information, or PII, credentials and keys that provide access to services or levels of capability.

There are many potential threats to the electronics with the emergence and adoption of what's called vehicle-to-everything, or V2X, technologies.

- V2I – vehicle-to-infrastructure
- V2N – vehicle-to-network
- V2V – vehicle-to-vehicle
- V2P – vehicle-to-pedestrian
- V2D – vehicle-to-device
- V2G – vehicle-to-grid

As highlighted in *Table 4: Connectivity Options and Opportunities for Hacking* below, from the ETSI TR-102-638 Intelligent Transport Systems technical report[4], the connectivity options and opportunities for hacking are great.

Table 4: Connectivity Options and Opportunities for Hacking

Applications Class	Application	Use case
Active road safety	Driving assistance - Co-operative awareness	Emergency vehicle warning
		Slow vehicle indication
		Intersection collision warning
		Motorcycle approaching indication
	Driving assistance - Road Hazard Warning	Emergency electronic brake lights
		Wrong way driving warning
		Stationary vehicle - accident
		Stationary vehicle - vehicle problem
		Traffic condition warning
		Signal violation warning
		Roadwork warning
		Collision risk warning
		Decentralized floating car data - Hazardous location
		Decentralized floating car data - Precipitations
		Decentralized floating car data - Road adhesion
		Decentralized floating car data - Visibility
		Decentralized floating car data - Wind
Cooperative traffic efficiency	Speed management	Regulatory / contextual speed limits notification
		Traffic light optimal speed advisory
	Co-operative navigation	Traffic information and recommended itinerary
		Enhanced route guidance and navigation
		Limited access warning and detour notification
		In-vehicle signage
Co-operative local services	Location based services	Point of Interest notification
		Automatic access control and parking management
		ITS local electronic commerce
		Media downloading
Global internet services	Communities services	Insurance and financial services
		Fleet management
		Loading zone management
	ITS station life cycle management	Vehicle software / data provisioning and update
		Vehicle and RSU data calibration.

Source: European Telecommunications Standards Institute

In addition to the use cases above, hacking and controlling a Jeep by Dr. Charlie Miller and Chris Valasek through a flaw in the Uconnect system is a commonly cited example on the dangers of insecure vehicles. This sort of security breach should get the respect it deserves. However there are many practical reasons for security in thwarting more likely forms of attack. Take for example, ensuring that odometers cannot be rolled back to change the monetary value of a vehicle (fraud) or hacking into a car's electronics to take control of it (auto theft). Another not-so-obvious example is keeping a competitor from reverse-engineering and copying the design of an electronic control unit with the intent of using it in their vehicle design (product cloning). Also, an auto must protect against owner hacking to gain access to services related to the vehicle experience like mapping, levels of autonomous service, audio, or video content (service theft). Then there is the emergence of electronic commerce, a good example of which is the adoption of Integrated Toll Modules to replace RFID stickers on windshields. Increasingly personal information is being stored in onboard systems, such as contacts and passwords. Like a computer system at home or your smart phone, this data must be kept secure and privacy has to be maintained.

Finally there is an intertwining link between safety and security. Applying the proper levels of security, in of itself, provides the determinism needed for safe operation. This is often overlooked when thinking about security alone. A robust security framework protects against unauthorized actions taken by individuals, while improving safety by incorporating additional controls in the system design. So, an advanced secure Flash device like Macronix SecureFlash can help improve safety and security at the same time.

Industrial - Edge Compute Platforms

An edge compute platform is the next wave of computing. It is due to explosive growth of connected devices, AI (artificial intelligence), and the vast amount of data that needs to be processed from these applications. Device examples include autonomous machines, smart city applications, smart home, and smart power grids, to name a few. The cloud represents a centralized computing architecture that needs to be decentralized to efficiently process sensor data. This also helps to solve latency issues for time-critical applications like autonomous vehicles. Now-available AI edge computing platforms have become very powerful. There are now entry-level AI edge computing device that sell for less than $100, delivering as much as 472 GFLOPs of compute performance. These devices at the edge are now able to handle greater amounts of data, some of which is highly sensitive.

With the decentralization of computing, moving back to remote locations raises security concerns. These devices can become vulnerable to attack via the public networks they are connected to. There will potentially be a number of users that need access. Each user has a certain level of security access that needs to be managed through the use of keys and credentials. Also levels of authorized capabilities and related authorization data need to be stored. Software updates have to be accomplished safely and securely. So, the need for external non-volatile memory will grow, along with the requirement to keep it secure. These powerful computing platforms typically don't have large amounts of embedded Flash and require some amount of external non-volatile Flash memory to store code. Some edge devices are cost sensitive so having non-volatile memory with secure data regions and regular non-volatile memory for code storage in one device is preferable. This secure memory can act and take the place of separate secure elements component, combining and reducing the number of devices needed in a design. This is where advanced memory products like ArmorFlash can help.

Conclusion

The growth of connected devices continues to explode and along with it the exponential growth of data. Computing architectures are moving to a decentralized model to adapt and is bringing about the growth of powerful edge computing platforms that include support for applications like personal automobiles, autonomous vehicles, remote medicine, smart infrastructures and homes, to name a few. This all has to be done safely and securely in an untrusted environment. Therefore non-volatile memory requires a range of security mechanisms and policies to ensure identity, confidentiality, integrity, authenticity, and availability. Advanced secure memory storage features found in Macronix Armor Flash, is a critical component to achieving these objectives.

References

1 – IoT Analytics, State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating, August 18, 2018, *https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/*
2 - TEE Internal Core API Specification, *https://globalplatform.org/specs-library/?filter-committee=tee*
3 – Replay Protected Memory Block, Retrieved on 5/7/2019 from: *https://www.jedec.org/sites/default/files/docs/JESD84-B51.pdf*
4 - ETSI TR-102-638 Intelligent Transport Systems technical report, Retrieved on 5/7/2019 from: *https://www.etsi.org/deliver/etsi_tr/102600_102699/102638/01.01.01_60/tr_102638v010101p.pdf*

Revision History

Table 5: Revision History

Revision No.	Description	Page	Date
Rev. 1	Initial Release	ALL	August 16, 2019

Except for customized products which have been expressly identified in the applicable agreement, Macronix's products are designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only, and not for use in any applications which may, directly or indirectly, cause death, personal injury, or severe property damages. In the event Macronix products are used in contradicted to their target usage above, the buyer shall take any and all actions to ensure said Macronix's product qualified for its actual use in accordance with the applicable laws and regulations; and Macronix as well as its suppliers and/or distributors shall be released from any and all liability arisen therefrom.

For the contact and order information, please visit Macronix's Web site at: http://www.macronix.com