Designing in A Trusted Platform Module (TPM)

Kerry Maletsky

September 12, 2005

Boston, MA

EmbeddedSystems Conferences San Francisco Boston Asia Munich

Agenda

EmbeddedSystems Conferences San Francisco Boston Asia Munich

What is security?
(In the embedded system context)

- Some Ideas:
  - Do I know what device I’m talking to?
  - Do I trust that it can maintain its private data adequately?
  - Can a device be confident in its own state?

- The Trusted Platform Module can address these concerns.

What is a TPM?

- A TPM is a key storage and generation device providing an affordable “secure vault” for embedded systems.
- Four Primary Capabilities of TPMs
  - Store User and Management Private Keys
  - On-chip Private Key Computation
  - Storage of Measurements of Configuration Values
  - Initialization and Management Functions

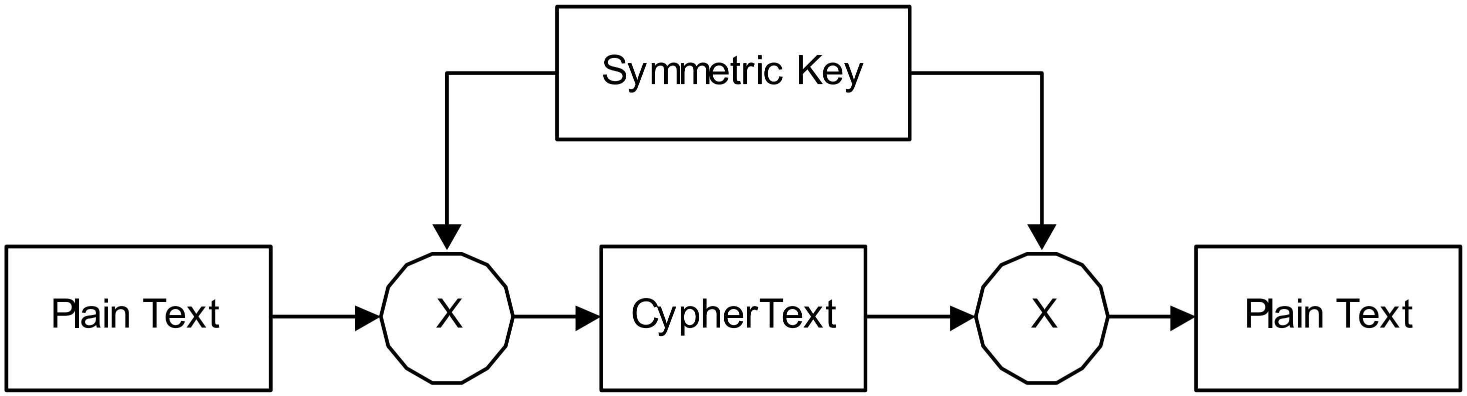
EmbeddedSystems Conferences San Francisco Boston Asia Munich

Cryptography Basics

- Symmetric Key Encryption
- Asymmetric Key Encryption
- RSA Encryption/Decryption Model
- RSA Signature/Verify Signature Model
- Wrapping Keys

Cryptographic Basics – Symmetric Key Encryption

Symmetric Key Algorithms use the ***same key*** to encrypt and decrypt data. AES & DES are examples of widely used symmetric algorithms.



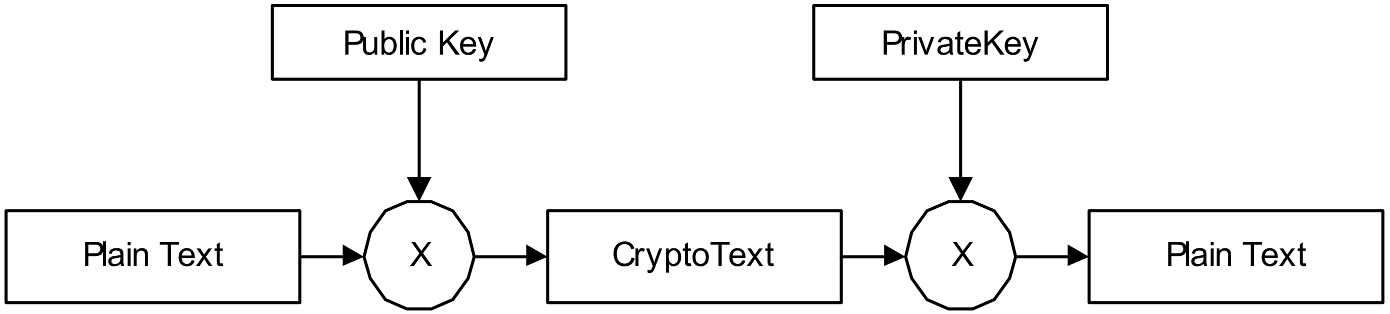
Both parties need to have and store the same secret key. But how does the sender get this key to the receiver?

EmbeddedSystems Conferences San Francisco Boston Asia Munich

Cryptographic Basics – Asymmetric Key Encryption

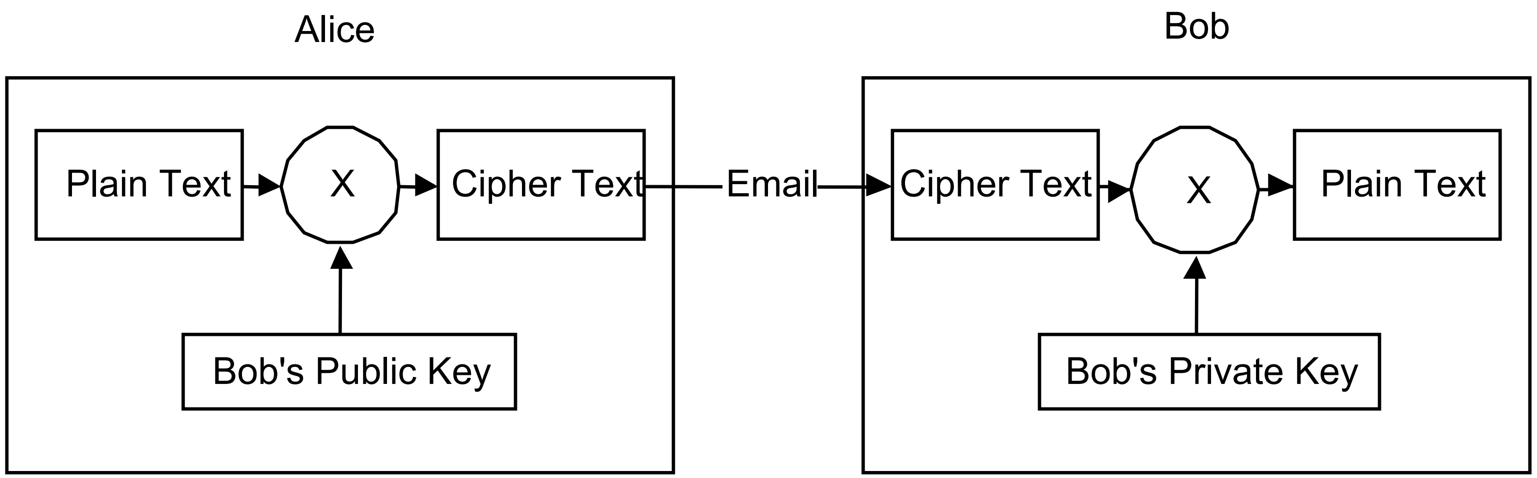
Asymmetric Algorithms use ***two related keys*** - public and private - to encrypt and decrypt data.

RSA is an example of an asymmetric algorithm.



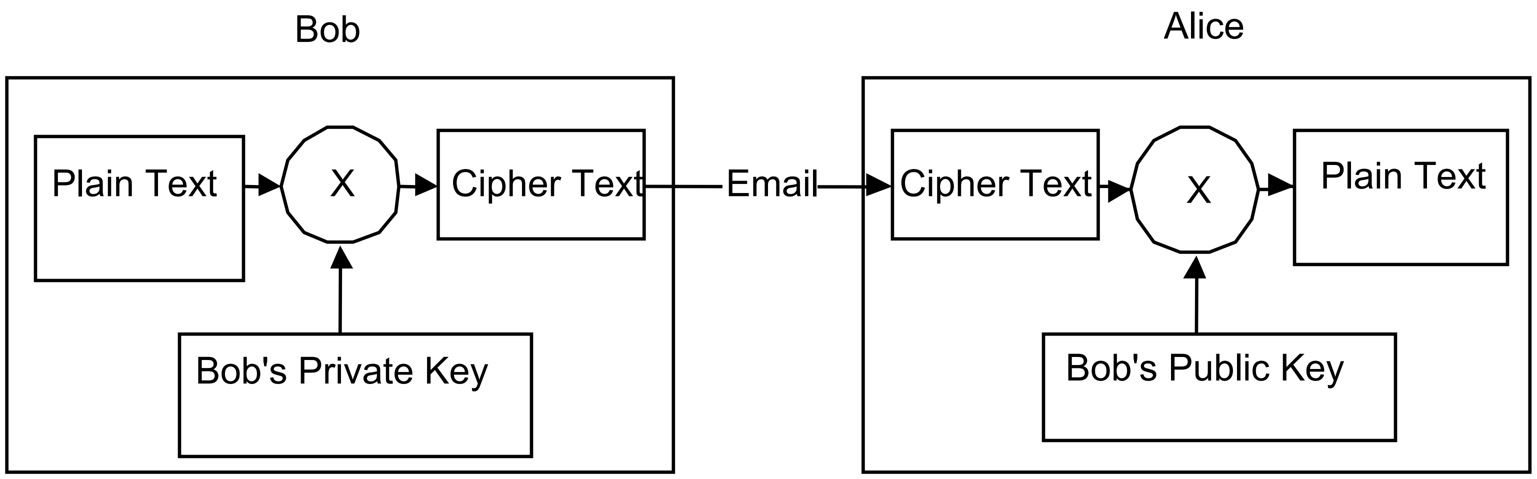
Public keys are public, known to everyone. Private keys are known only by their owner, and are never shared.

Cryptographic Basics – RSA Encryption/Decryption Model



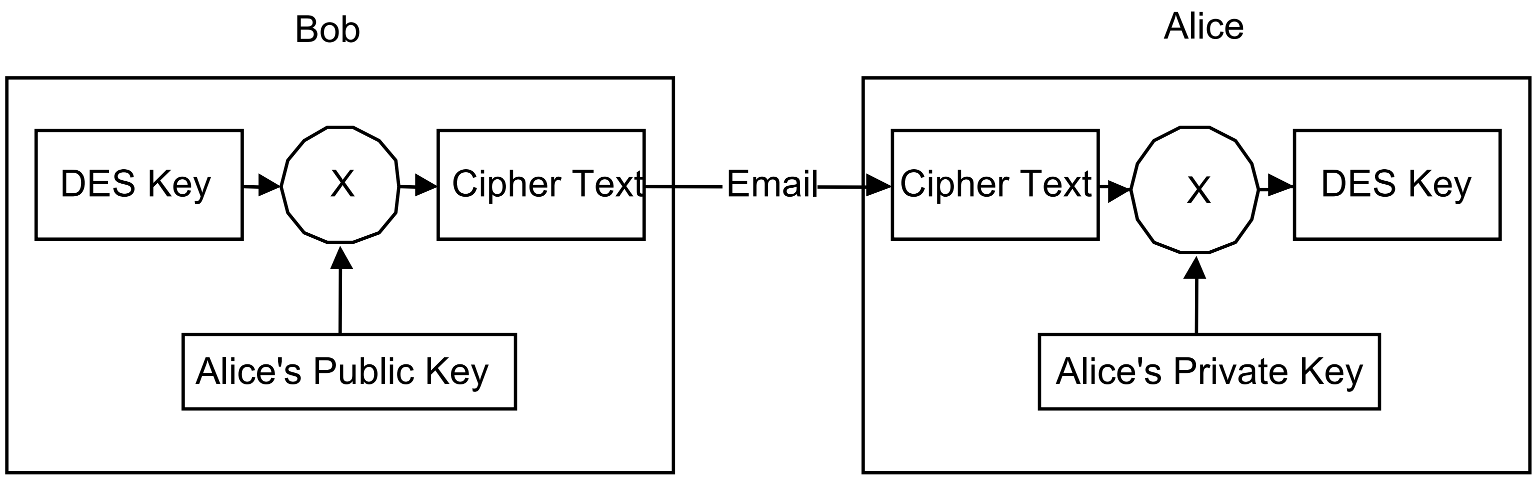
Alice’s computer doesn’t do anything secret. But Bob’s computer needs to use Bob’s private key without compromising it.

Cryptographic Basics – RSA Signature / Verify Signature Model



Bob's computer needs to use Bob's private key without compromising it. But Alice's computer doesn't do anything secret.

Cryptographic Basics – Wrapping Symmetric Keys



Asymmetric algorithms are typically slow. So we use those algorithms to exchange a completely random ‘session key’ and then use a fast symmetric algorithm to encrypt the longer message.

Cryptographic Basics – Symmetric vs. Asymmetric Algorithms

- Symmetric algorithms (DES, 3DES, AES)
  - Smaller keys
  - Very fast execution
  - Key distribution problem
- Asymmetric algorithms (RSA, ECC, etc.)
  - Larger keys
  - Slower execution
  - Key distribution is not an issue

TPM Architecture

- TCG Intro
- Keys stored in a TPM
- TPM Cryptographic Capabilities
- Key Management
- Key Types
- Useful TPM Commands

Who is TCG?

- **Trusted Computing Group**
  - www.trustedcomputinggroup.org
  - Focused on bringing security to PCs and other computing devices
  - But working groups and applications for embedded systems as well.
- **Industry wide standards body**
  - Non profit, RAND patent licensing policy
- **Leading Technology Companies**
  - High degree of security expertise

1

Slide 13

1 Do we want to note that more than 15 million TPMs shipped to date for PC space and migration to embedded beginning:

WGs embedded developers might want to know about - storage, peripherals, networking (IWG)
, 7/20/2005

What keys are in a TPM?

- **TPM stores multiple private RSA keys.**
- **TPM performs the private key operation internally.**
  - The key never needs to be known outside the TPM.
  - TPM prevents everyone (even the user) from knowing the private key value.
- **TPM provides a secure way to do key backups.**

EmbeddedSystems Conferences San Francisco Boston Asia Munich

TPM – Cryptographic Capabilities

- RSA Encryption & Decryption
  - Seal (encrypt data that can only be decrypted on this platform)
  - Bind (encrypt data that can only be decrypted by a key, which can be stored on this platform)
  - 512, 1024, & 2048 bit keys supported
- Random Number Generation
- SHA-1 Hashing
- Key Loading / Unloading
  - TPM implements a key cache – uses RSA to encrypt keys that can then be stored externally and reloaded later.

TPM – Cryptographic Capabilities (continued)

- AES Encryption & Decryption
  - Used only for encryption / decryption of command packets and to safely store internal TPM data on an external device
- Real Time Clock
  - Secure time stamps
- Hardware Security
  - To defend against physical attacks on the TPM

EmbeddedSystems Conferences San Francisco Boston Asia Munich

Key Management:
The TPM Key Hierarchy

- Keys are the cornerstone of a system’s security architecture.
- Keys must be generated, stored and manipulated.
  - Key operations
  - Key classes and hierarchy

Key Management – RSA Key Generation

- Quality Hardware Random Number Generator (RNG) crucial to strong keys
  - Software-only based RNGs can sometimes be cracked (output predicted)
- RSA key generation requires two very large prime numbers

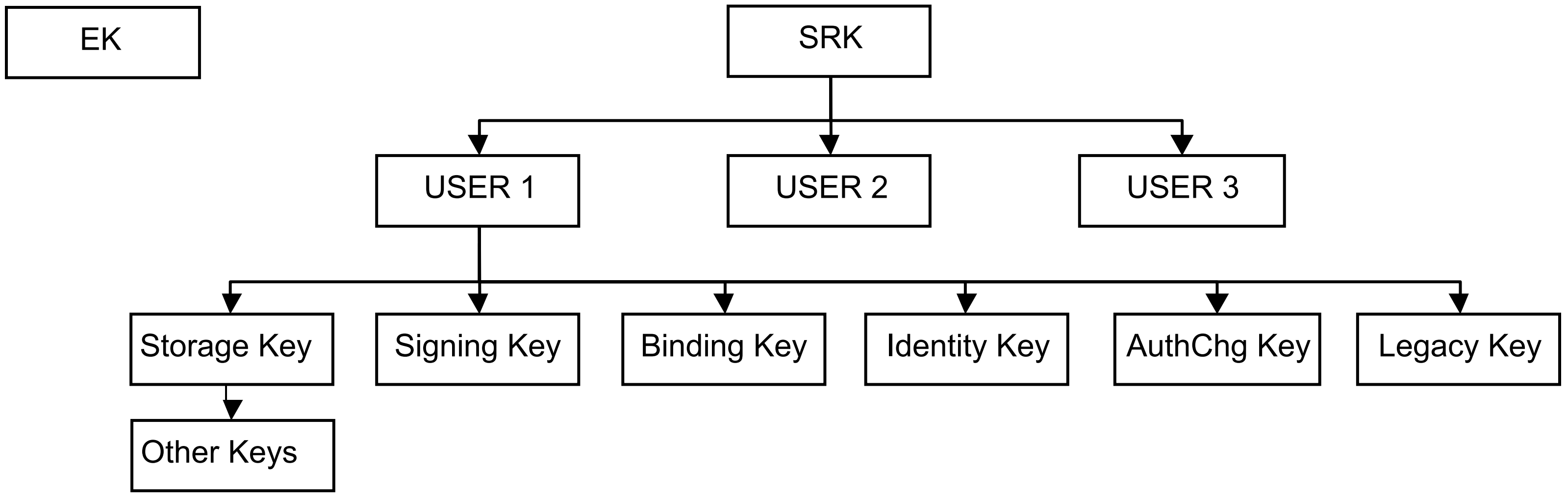
Key Management – Storing Keys

- Secure storage of keys is crucial to trusting the security architecture.
- TPMs provide a hardware vault, protected against common software attacks.
  - Private keys are stored in the TPM and are never exposed “in the clear” outside of a TPM.
  - All Private Key operations occur internal to the TPM.
- No capacity limits
  - TPM can encrypt the keys and swap them in/out as necessary
  - Secured using 2048 bit RSA

Key Types – Archive/Migration Keys

- Archive keys allow for user keys and data to transferred from TPM to TPM.
  - Critical when users upgrade systems or if TPM hardware fails.
- Archive keys “wrap” all of its child keys
  - Keys are protected while stored on a network or CD.
- User data can be restored by migrating the archive key into a new system.
- Migration is optional (spec. when generating keys)
  - Enterprise users may like this, others may not

TPM – Basic TPM Key Hierarchy



TPM – Basic TPM Key Hierarchy

- All TPMs contain a basic key management structure.
- OEMs arrange for each TPM to include a unique Endorsement Key (EK)
  - Each and every system is different
- Owners create a Storage Root Key (SRK) to “take ownership.”
  - Enables the owner to assign individual user keys

TPM – Endorsement Key (EK)

- The EK is the basic root of trust for identification
  - Every TPM has a unique EK
  - A certificate of this EK (generated by a trusted certificate authority) attests to the security properties of the system
- EK's can only be generated once, usually before the completion of the system manufacturing stage.
- EK's never leave the TPM

TPM – Storage Root Key (SRK)

- **The SRK is the root of the storage tree for all keys related to the TPM (except EK)**
  - Generated by the owner of the system
  - SRKs can never leave a TPM.
- **Owners can remove SRK to relinquish ownership of a TPM.**
  - Elimination of all keys within or related to the TPM
  - Facilitates change in asset ownership.

TPM – Other Key Types

- **Signing Key**
  - Can be used to generate a signature. Could sign externally provided data or internal TPM state.
- **Storage Key**
  - Can be used to store either keys (encrypted in the key hierarchy) or data (using the TPM_Seal operation). (SRK is a non-migratable storage key)
- **Binding Key**
  - Can be used to decrypt (unbind) data that was bound to the public key by a remote entity.

EmbeddedSystems Conferences San Francisco Boston Asia Munich

TPM – Other Key Types

- **Identity Key**
  - With an accompanying certificate from a trusted certificate authority (CA), attests to the identity of the TPM and its security level.
- **Legacy Key**
  - Can be used for binding or signatures. May be necessary for interoperability with existing systems, not recommended for new designs.

Useful TPM Commands

- TPM_LoadKey
  - Load a key onto the TPM. Typically it will be retained unless explicitly removed by the system. Key can be generated by the TPM or an external system.
- TPM_Sign
  - Sign data that is presented to the TPM. Can be used to sign an outgoing email message, for instance, or sign a challenge directed to the TPM by some remote system, such as a network host.
- TPM_UnBind
  - Decrypt data that has been encrypted with a public key, the private portion of which is stored on the TPM. Useful to exchange a random session key that's been used to encrypt a larger block

Useful TPM Commands

- TPM_Seal/Unseal
  - Connect data to a particular TPM. Seal encrypts the data so that it can only be decrypted by this particular TPM when it is in an environment that has three properties: 1) knows the sealed authorization secret, 2) Can load and use the parent key to which the data is sealed, 3) PCRs are in the right state.
- TPM_OIAP/OSAP
  - Start an authorization session. OIAP is “object independent” – the secret is required for each command, OSAP is ‘object specific” and a shared secret is used for multiple commands connected to the same entity
- TPM_GetCapability
  - Get configuration or status information from the TPM.

Useful TPM Commands

- TPM_Extend
  - Add data to a PCR. Specifically, the input is intended to be a 20 byte hash that is hashed into the current PCR value. The system must store the sequence of elements hashed into the PCR.
- TPM_Quote
  - Sign the current value of a PCR – provides trust that the component hashes were presented to the TPM and in the order specified.
- TPM_CreateWrapKey
  - Generate a unique RSA key according to the parameters specified (key size, migration, secret value, etc), wrap (encrypt) it with the parent key and send to the external system. The system should store the key in the key cache manager database.

Useful TPM Commands

- TPM_GetRandom
  - Get a random number from the TPM random number generator. TPM’s provide a very high quality FIPs type generator that can be trusted for all applications.
- TPM_MakeIdentity
  - Along with TPM_ActivateIdentity and a CA, permits an authorized user to generate an Attestation Identity Key (AIK). Multiple AIKs can be generated, so the user can have multiple identities that are unique from each other.
- TPM_TakeOwnership
  - Typically done once when the TPM (system) is purchased. Creates the SRK, stores the TPM owner authorization secret, etc.

TPMs in an Embedded System

- TPM Benefits
- Application Examples

Why Use a TPM in an Embedded Design?

- **Flexibility**
  - Broad command architecture allows use in a wide array of application areas
- **Standards-based**
  - Uses standard cryptographic algorithms and protocols that are widely accepted
  - Will interoperate with other systems using software implementations

Why Use a TPM in an Embedded Design?

- **High Security**
  - Third party certification (Common Criteria EAL 3+, 4+)
  - TCG commands & protocols have been widely analyzed
- **Turnkey**
  - Sold complete with internal firmware, no chip programming required
  - No algorithmic expertise required.
- **Exportable**

Embedded Application Examples

- Confidence in Current State
  - **Modern equipment can be configured in many ways, may work with a variety of software modules**
  - **Problem: Modules may be corrupted (maliciously or inadvertently), for safety or reliability reasons it may be necessary to do a real time audit, etc.**
  - **How the TPM can help:**
    - As each module is loaded, its value (hash) is added to the PCR registers. The core software or hardware ensures that nothing is loaded without being hashed.
    - At any time the system or a remote entity can ask the TPM to sign the PCR state with a TPM key. Can verify that at that particular time, that particular system was in that particular state.

Embedded Application Examples

- Trusted Download of Software Updates
  - **Systems are so complex that the use of FLASH memory to permit in-system updates is commonplace.**
  - **Problem: How does the OEM manage these downloads – to prevent faulty software from being run or to require proper payment.**
  - **How the TPM can help:**
    - Perform the asymmetric algorithm signature verification so that the system does not need to incorporate that logic.
    - Store the root of trust for the verification chain securely so that it cannot be modified.
    - Authenticate device to download site, maintain audit trails of updates.

Embedded Application Examples

- Secured Network Communications
  - **An industrial control network needs to know both that the nodes are authentic and that the communications channels are not being corrupted**
  - **Problem: With modest computing capability and high volumes, sometimes it isn't practical to build in high security.**
  - **How the TPM can help:**
    - Use signatures to authenticate nodes, rely on CC certification to be sure they aren't copies.
    - Use key exchange mechanisms (TPM_Unbind) to build dynamic session keys for communications
    - Low cost – practical for a 10,000 node web.

Embedded Application Examples

- Reliable peripheral identification
  - **Especially in high value systems, it’s important to ensure that add-on or replacement parts are authentic.**
  - **Problem: There’s often no way for the system to achieve the necessary confidence.**
  - **How the TPM can help:**
    - The OEM manufacturer includes a TPM in the part, along with a signature of a TPM key. When necessary, the system requests the new part to sign a random number to verify its authenticity.
    - Parts can be personalized – either to work with a particular system or to be configured in a special way – by using the TPM to secure data that then requires some system secret to decode.

Embedded Application Examples

- Local Secure Storage
  - Many embedded systems store, either permanently or temporarily, data that could be sensitive. For example, data to be printed or faxed in office eqpt., transaction information in a POS, etc.
  - Problem: These same pieces of equipment need to be connected over a network, making them vulnerable to a remote attack.
  - How can the TPM help:
    - Encrypt all stored information in disk/flash.
    - Keys aren't stored in accessible locations, rather inside the TPM 'vault'.

Embedded Application Examples

- Personnel Authorization
  - **Specific individuals may be authorized to perform varying operations on a piece of equipment**
  - **Problem: How does the system store both the identifiers (e.g. password) as well as protect the capability to be authorized?**
  - **Ways TPM Can Help**
    - Authorization check can be done inside the TPM, passwords never need to be stored in the clear. TPM can also support token or smart card authentication schemes.
    - Authorized capability can be encrypted and only released when above happens correctly. (e.g. encrypt special program blocks using TPM_Seal, etc).

EmbeddedSystems Conferences San Francisco Boston Asia Munich

Owner and User Definitions and Capabilities

- TPM Authorization Protection
  - TPM Authorization Block
  - TPM Entities and Authorization Secrets
  - TPM Nonce and TSS Nonce
  - Authorization Session Lifetimes
  - Authorization Digest Calculation

TPM Authorization Block

- Authorization chain dependent upon the initial creation of an authorization session
- Authorization session creates initial nonce or TPM nonce and caches parameters required to calculate authorization digest
- Input Authorization Block:

AUTH HANDLE	- Handle that references the auth cache
NONCE ODD	- System generated 20 byte random value.
CONT AUTH	- Flag that when set terminates the session.
AUTH DIGEST	- 20 byte command input digest (HMAC).

- Output Authorization Block:

NONCE EVEN	- TPM generated 20 byte random value.
CONT AUTH	- Flag that reports the auth session state.
AUTH DIGEST	- 20 byte command output digest (HMAC).

TPM Entities and Authorization Secrets

- TPM Entities are related to operations that can be performed by the TPM. Examples:

TPM OWNER	- Owner of the TPM.
SRK	- Key pair belonging to the owner.
USER KEY	- General key pair for users or software

- Authorization Secrets
  - Entities know secrets used to authorize TPM operations
  - Ephemeral shared secrets can be generated by software using an entity secret so that multiple operations can be performed without repeated authorization requests.

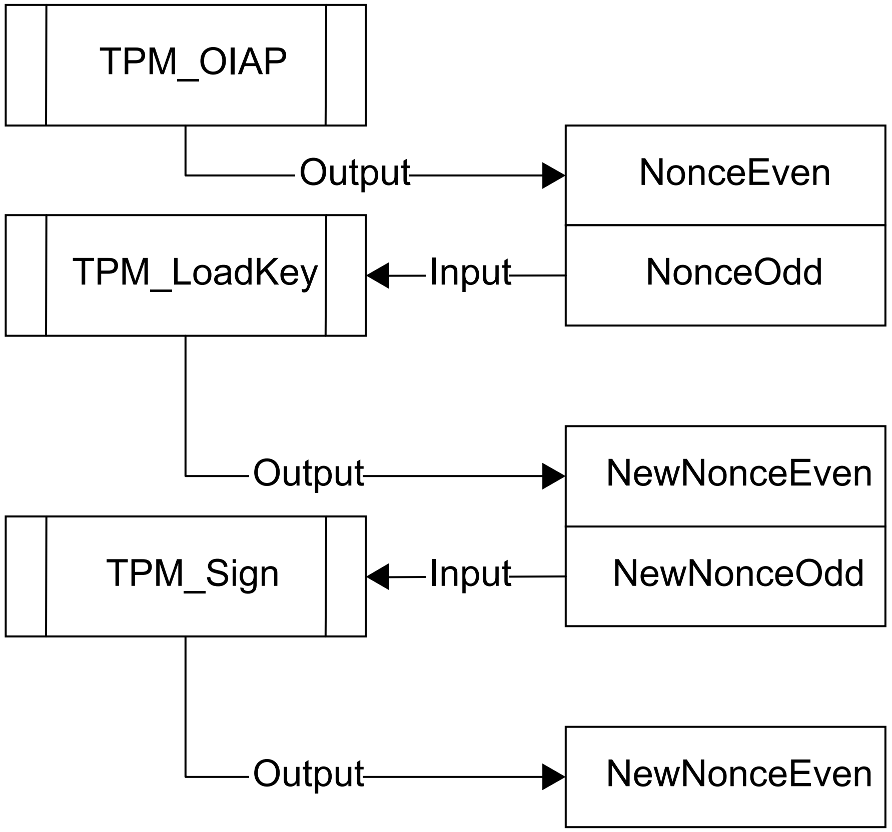
TPM Nonce and TSS Nonce Definitions and Example Flow

20-byte random value

TPM nonce = nonce even

TSS nonce = nonce odd

Nonce chaining protects against message replay and man-in-the-middle attacks



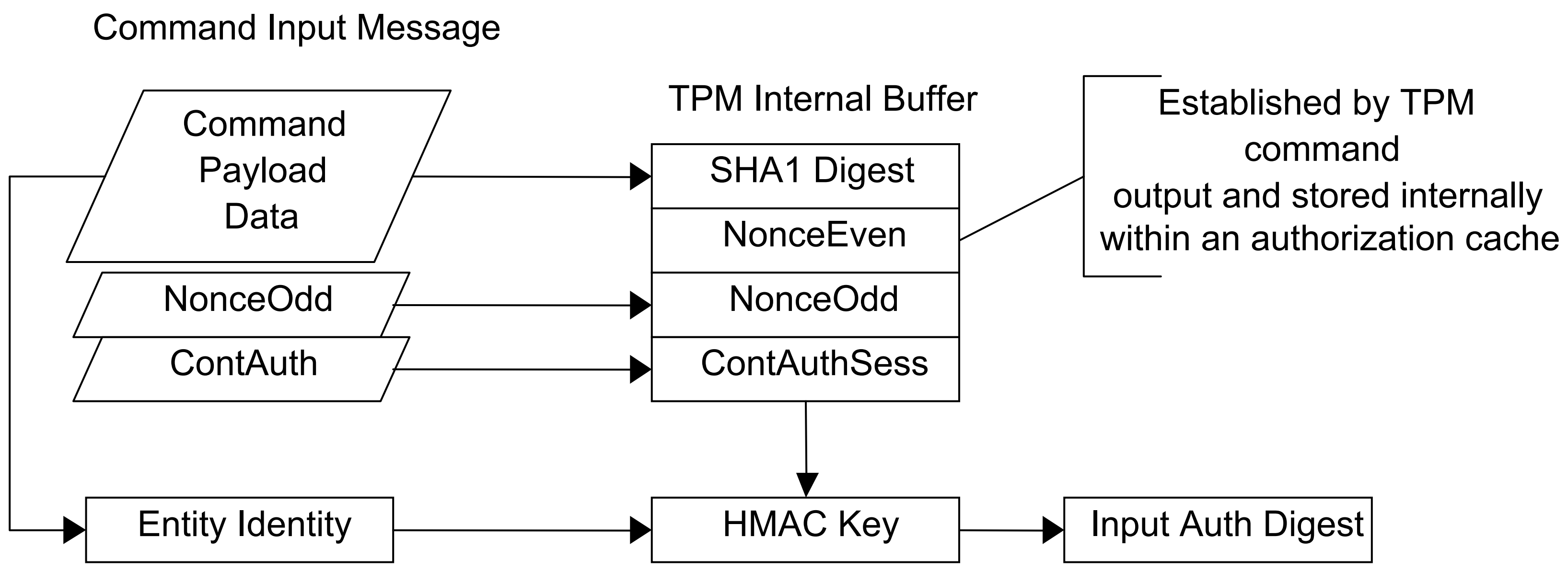
EmbeddedSystems Conferences San Francisco Boston Asia Munich

Authorization Session Lifetimes

- Sessions are left open when TPM commands are successful.
- Sessions are closed when TPM commands fail or when the continue authorization byte is set to zero.
- The continue authorization byte can eliminate the need to explicitly call the TPM_TermHandle command.

Authorization Input Digest Calculation

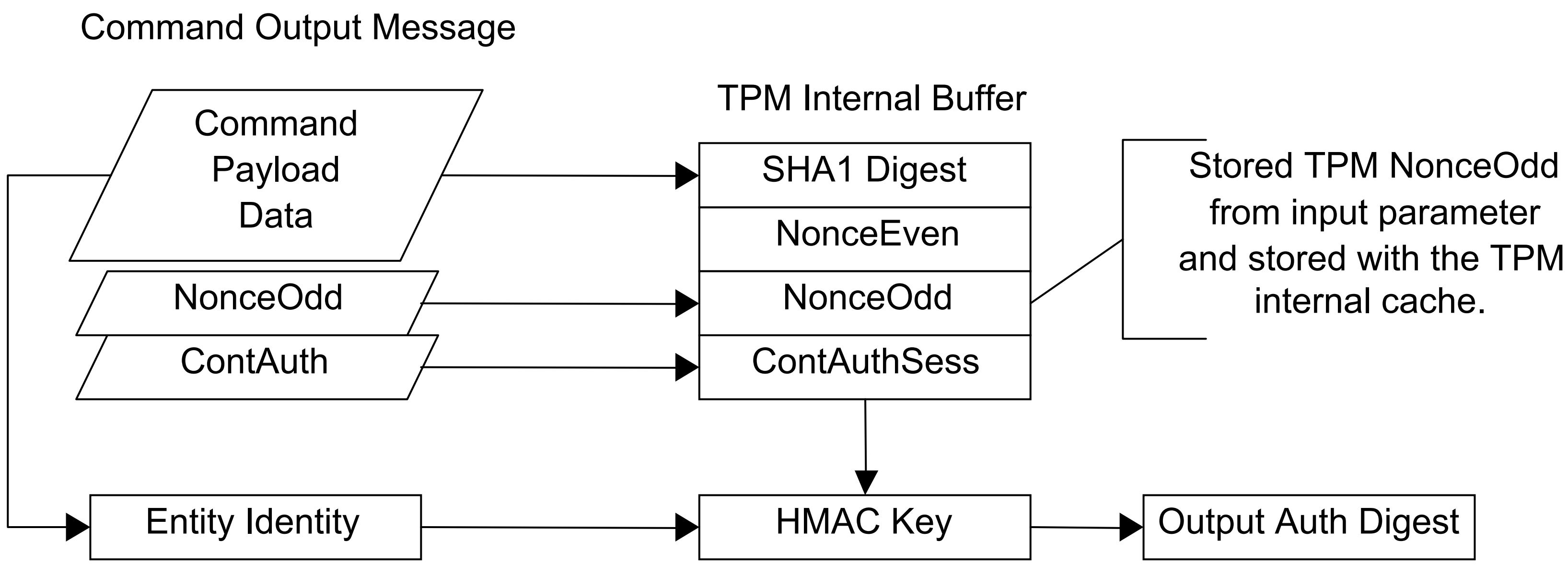
Authorization digest calculated from selected parameters and the continuation authorization parameter.



Input Authorization Digest Calculation

Authorization Output Digest Calculation

An Output digest calculation is similar, but uses parameters associated with the output message.



Output Authorization Digest Calculation

Introduction to the TCG Message Blocks

- Input Blocks
- Output Blocks

TPM Message Basics – Input Block

All TPM Messages Have Basic Common Parameters & Optional Parameters

TAG	- Defines the level of authorization.
PARAMETER SIZE	- The size of the incoming block.
ORDINAL	- The Command Code of the TPM function.
KEY HANDLE	- OPTIONAL
KEY HANDLE	- OPTIONAL
PAY LOAD	- Any parameters required by the command.
============ Authorization Block One (optional) =============	
AUTH HANDLE	- Handle identifying the authorization resources
NONCE ODD	- Nonce generated by the TSS.
CONT AUTH	- Flag the determines if the auth session is closed.
AUTH DIGEST	- The authorization HMAC.
============ Authorization Block Two (optional) =============	
AUTH HANDLE	- Handle identifying the authorization resources
NONCE ODD	- Nonce generated by the TSS.
CONT AUTH	- Flag the determines if the auth session is closed.
AUTH DIGEST	- The authorization HMAC.

EmbeddedSystems Conferences San Francisco Boston Asia Munich

TPM Message Basics – Output Block

All TPM Messages Have Basic Common Parameters & Optional Parameters

TAG	- Defines the level of authorization.
PARAMETER SIZE	- The size of the incoming block.
RETURN CODE	- Status of the command execution.
PAY LOAD	- Output of execution results.
============ Authorization Block One (optional) =============	
NONCE EVEN	- Nonce generated by the TPM.
CONT AUTH	- Flag the determines if the auth session is closed.
AUTH DIGEST	- The authorization HMAC.
============ Authorization Block Two (optional) =============	
NONCE EVEN	- Nonce generated by the TPM.
CONT AUTH	- Flag the determines if the auth session is closed.
AUTH DIGEST	- The authorization HMAC.

Building a TCG Message Input Command

- TPM_Startup Command Example
  - Startup is the first command sent to a TPM
    - Several options: Startup, Startup Clear, Startup Deactivate
    - Startup Clear example detailed

Startup Clear Example

- The Clear option clears the state of the TPM.
- Input message block:

0x00 0xC1	- Non-Authorized command input tag.
0x00 0x00 0x00 0x0C	- Parameter Size.
0x00 0x00 0x00 0x99	- Command Code.
0x00 0x01	- Startup Clear Parameter.

- Output message block:

0x00 0xC4	- Non-Authorized command output tag.
0x00 0x00 0x00 0x0A	- Parameter Size.
0x00 0x00 0x00 0x00	- Return Code.

TCG Specification v1.1b versus v1.2

- What version is appropriate for Embedded Designs?
  - v1.1 TPMs offer complete cryptographic and security functions in hardware.
    - Standards-based, high security
  - v1.2 TPM specification adds new commands, PC specific hardware
    - Necessary for new PC architectures and operating systems
    - May be overkill for many embedded designs

EmbeddedSystems Conferences San Francisco Boston Asia Munich

Putting It All Together – Summary

- TPMs provide standards based security
  - Turnkey, low cost
  - High security, third party certifications, verified protocols
- Excellent embedded system solution
  - Low engineering cost to add to system
  - No detailed cryptographic knowledge required
- Standard product
  - Available from multiple sources, various IO channels, performance levels, etc.
  - All compatible with each other and with standard software

Acronyms

- AIK – Attestation Identity Key
- CA – Certificate Authority
- DES – Data Encryption Standard
- EK – Endorsement Key
- HMAC – Hash-based Message Authentication Code
- NIST – National Institute of Standards and Technology
- OIAP – Object Independent Authorization Protocol
- OSAP – Object Specific Authorization Protocol
- PCR – Platform Configuration Register
- RNG – Random Number Generator
- RSA – Rivest, Shamir and Adelman
- RTC – Real Time Clock
- SHA – Secure Hash Algorithm
- SRK – Storage Root Key
- TCG – Trusted Computing Group
- TDD – TPM Device Driver
- TDDL – TDD Layer
- TPM – Trusted Platform Module
- TSS – TCG Software Stack