

# Construction of a Trusted Virtual Security Module

**Ronald Toegl**

Secure and Correct Systems Group (SCoS)  
Institute for Applied Information Processing and Communications (IAIK)  
Graz University of Technology, Austria  
[ronald.toegl@iaik.tugraz.at](mailto:ronald.toegl@iaik.tugraz.at)



This work is supported by the Österreichische Forschungsförderungsgesellschaft (FFG) through project acTvSM, funding theme FIT-IT, no. 820848.

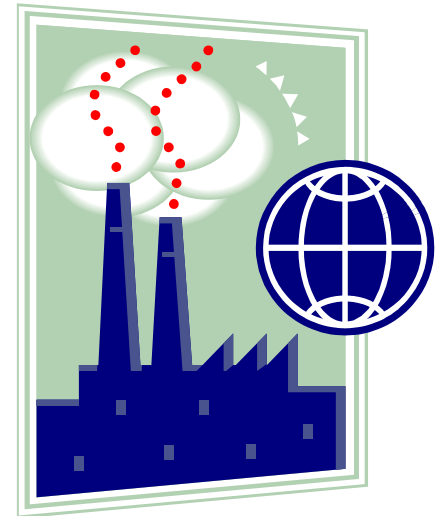
Acknowledgments go to: Florian Reimair, Martin Pirker, Andreas Niederl,  
Michael Gissing, Michael Gebetstroither

# Agenda

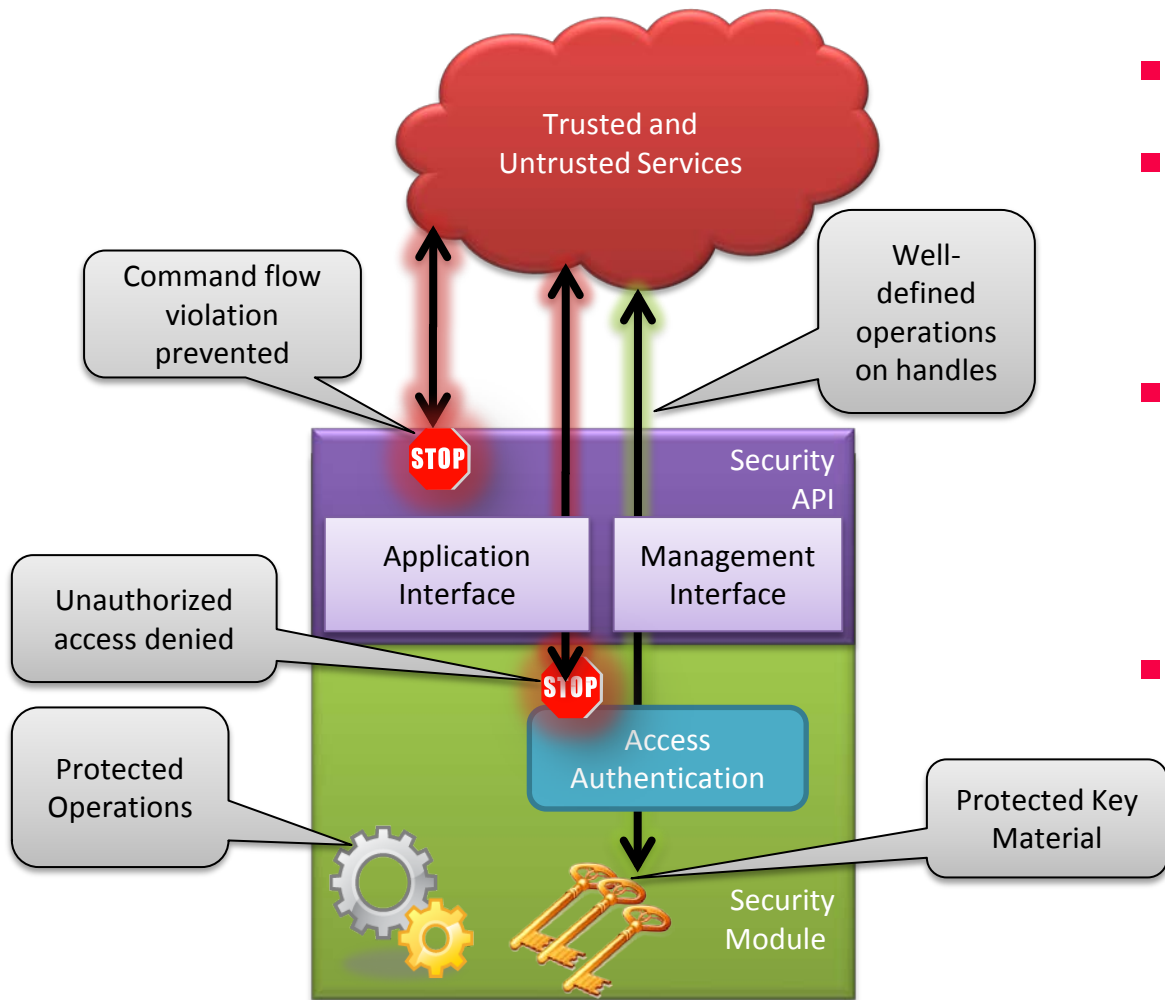
- Motivating Scenario
- Software Key Stores, HSMs..  
..and how about a TvSM?
- A platform for trusted virtualization
- A virtual security module
- API & Analysis Outlook
- Conclusion

# Motivating Scenario

- A company replaces custom, paper-based processes involving  $10^3$  people
- New processes based on (possibly advanced) electronic signatures
- Key material must be protected
- In a flexible, scalable, available way
- At a very low price



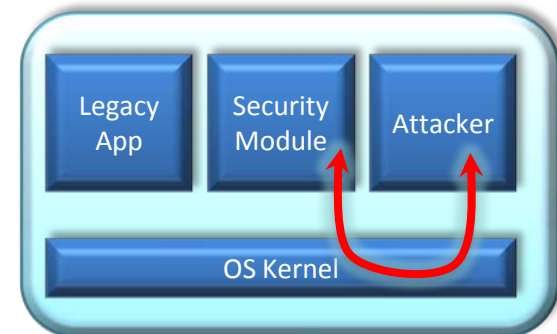
# Security Modules



- Cryptographic engines
- Key Management
  - Creation
  - Import/export
- Keep internal state over sessions
  - (Master) keys
  - Access secrets
- Protect against
  - Semi-/Non-/Invasive
  - Remote attacks

# Security Module Implementations

- **Software key stores**
  - Simple OS process
  - Protect against some basic attacks
  - Cheap



- **Hardware Security Modules**
  - Separated from general purpose system
  - Temper-resilient encasings
  - Potentially costly – if performance matters



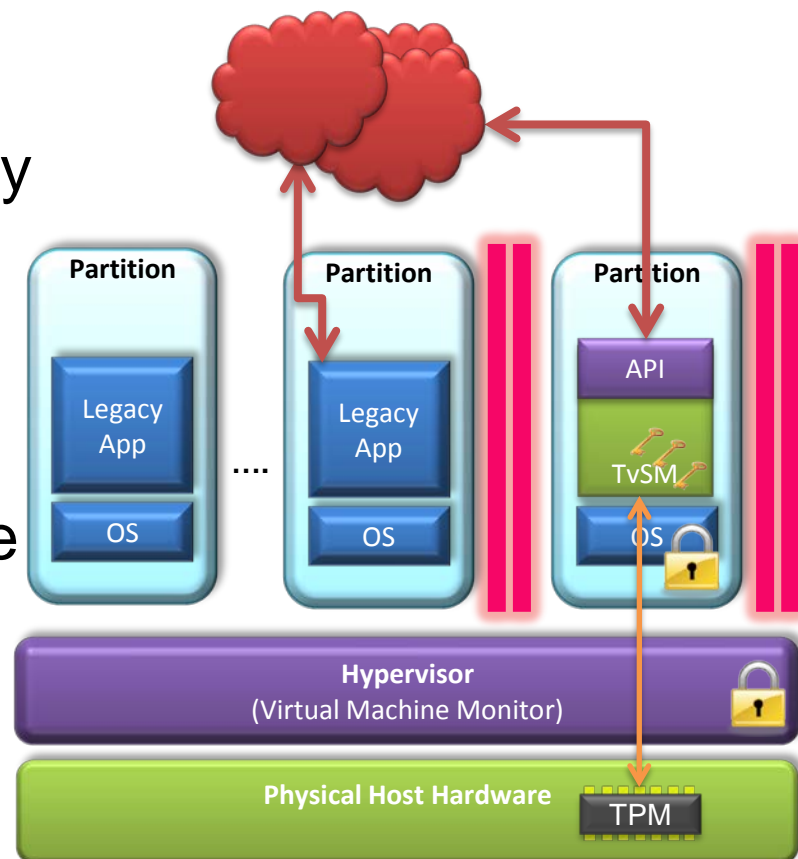
# An Alternative

- *Use hardware security found in commodity PCs to protect a software security module.*
- Trusted Platform Module (TPM)
  - Small, slow, specialized HSM
  - CC-EAL 4+ certified
  - Collects measurements on software configurations
- Intel Trusted eXecution Technology (TXT)
  - Dynamic switch to trusted CPU state at any time
- CPU support for Virtualization
  - Strong isolation of security critical from legacy code
  - Virtual Machines: Scalable, flexible, cheap

# Proposed Architecture: Trusted virtual Security Module

Implementation is

- Bound to HW-protected TPM key
- Ensured integrity
- Isolated from other services
- Specialized for specific use case
- Exposed through a minimized interface only
- Many instances



# Trusted Platform Module - Sealing

- Platform Configuration Registers (PCRs): can hold a **chain a trust** (i.e. BIOS, peripheral firmware, boot loader, kernel, drivers, libs, apps, configs)

- TPM **Sealing**:

Encrypt data under a **non-migratable** key and to a set of PCR values

$$PCR_i^{t+1} = \text{SHA-1}(PCR_i^t || x)$$

$$PCR_i^{t_0} = \begin{cases} 0xFF^{20} & 17 \leq i \leq 22 \text{ with static boot} \\ 0x00^{20} & 17 \leq i \leq 22 \text{ after dynamic (re-)boot} \\ 0x00^{20} & \text{else.} \end{cases}$$

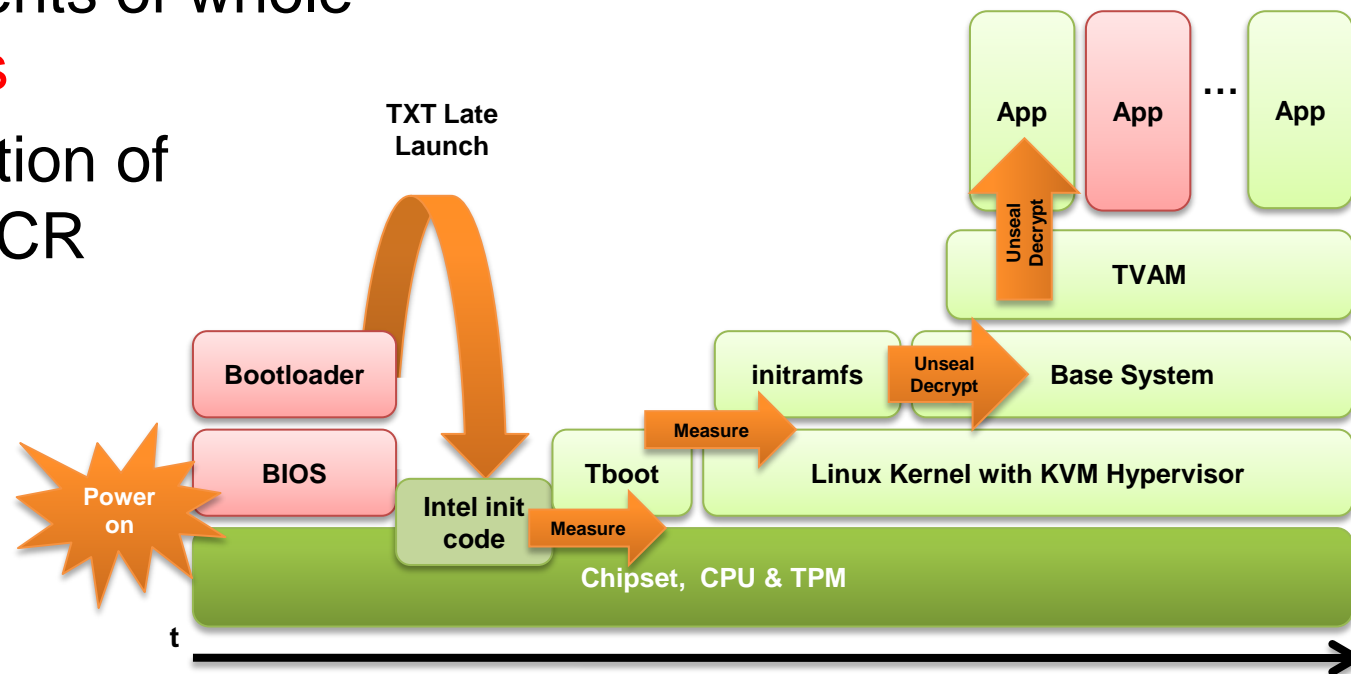
(→ integrity and confidentiality, no offline access)

- PCR configuration must be consistent, deterministic, and expressive

# Virtualization Platform

- Linux with KVM for virtualization
- TXT late launch: chain of trust only depending on well-known components
- Measurements of whole **file systems**
- Pre-calculation of expected PCR values

→ *Sealing becomes practical*



# Virtual Security Module

- Features typical of a Security Module
  - Session Management
  - Authorization of operations, based on key pairs
  - Performing cryptographic signatures with RSA
  - Typing of keys
- TRNG Key pair creation
- Master key management
- Wrapping (encrypting) and Unwrapping of keys for external storage
- Flexible, fast and scalable

# API Design Decisions

- No full PKCS#11 interface
- Single use case: management and usage of RSA keys for cryptographic signatures
- Avoid arithmetic functions
- Group related functionalities
- Use creation-time fixed types for keys according to their use
- Use verified authentication protocol – SKAP [Chen, Ryan]

Package `iaik.tc.tvsm.prototype.api`

## Interface Summary

<a href="#">Authentication</a>	The Interface Authentication.
<a href="#">Datablob</a>	The Interface Datablob.
<a href="#">DateTimeReport</a>	Holds a date/time object as well as the time needed for retrieval.
<a href="#">JCEKeyContainer</a>	Serves as container for JCE compatible keys.
<a href="#">Key</a>	The Interface Key.
<a href="#">KeyAttributes</a>	Defines the basic attributes of cryptographic keys.
<a href="#">KeyPair</a>	The Interface KeyPair.
<a href="#">MaintenanceSession</a>	Represents a number of maintenance and setup f
<a href="#">NoAuthentication</a>	for use where no authentication is needed.
<a href="#">OperatingSession</a>	Provides all everyday operations on the TvSM.
<a href="#">Result</a>	The Interface Result.
<a href="#">Secret</a>	for use in secret authentication.
<a href="#">Session</a>	Represents a communication session with the TvS
<a href="#">SessionFactory</a>	A factory for creating backend Session objects.
<a href="#">SessionInitRequest</a>	holds the User required for session authentication
<a href="#">SessionInitResponse</a>	holds the authorization handle for rmi session retri
<a href="#">TimestampResult</a>	holds the resulting datablob after the timestamp o signature and the timestamp as object
<a href="#">User</a>	for use in identity-based authentication.
<a href="#">Username</a>	for use in Username only authentication.

Interface `KeyAttributes`[iaik.tc.tvsm.prototype.api](#)public interface `KeyAttributes`

Defines the basic attributes of cryptographic keys. It allows to specify the type of and general parameters such as key length.

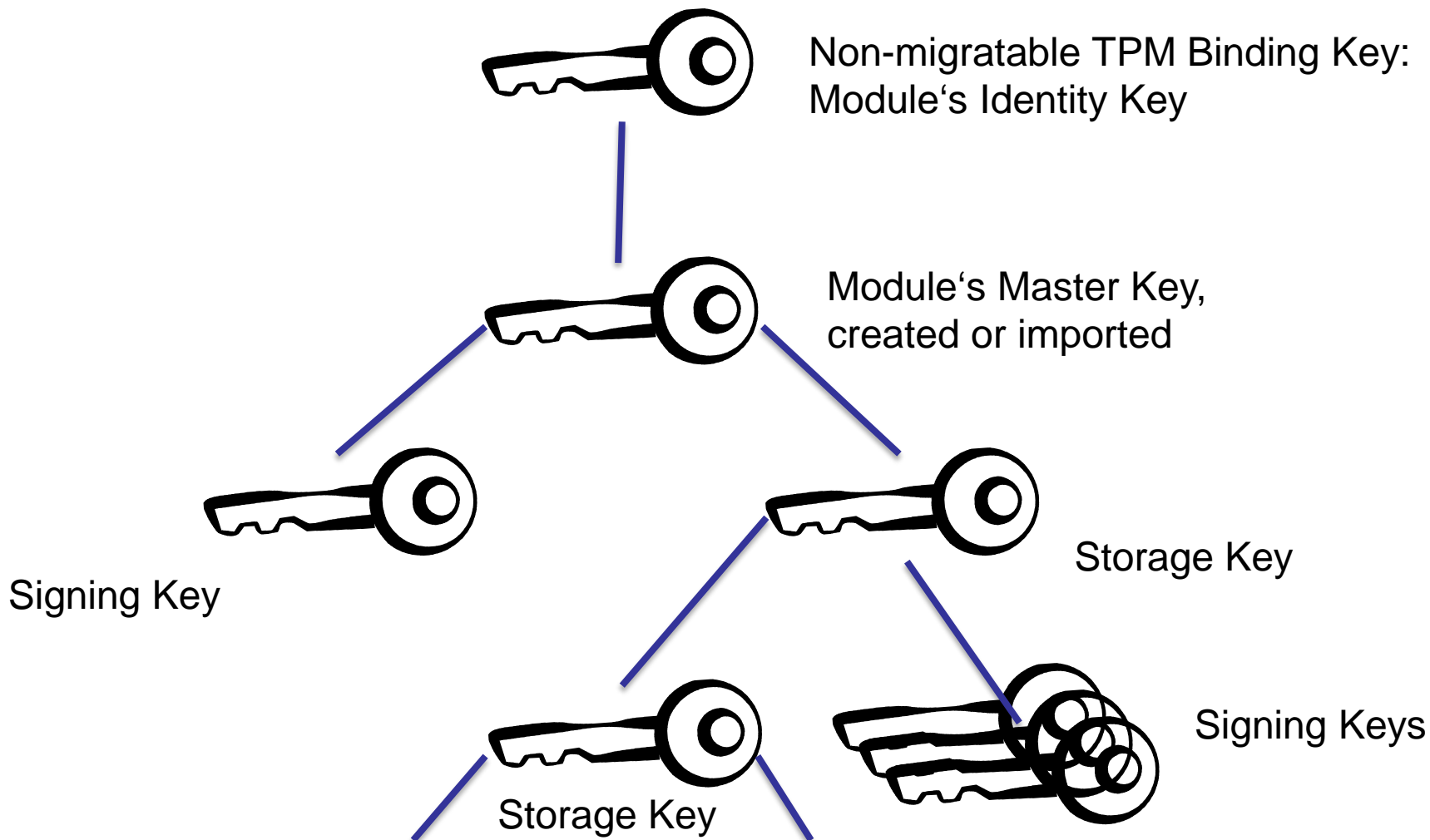
## Field Summary

int	<a href="#">MASTERKEY</a>	The Constant MASTERKEY.
int	<a href="#">MIGRATEABLE</a>	The Constant MIGRATEABLE.

## Method Summary

Result	<a href="#">createKey</a> ( <a href="#">KeyHandle</a> parent, <a href="#">Authentication</a> parentAuthentication, <a href="#">Authentication</a> newSecret, <a href="#">KeyAttributes</a> keyParams)	Creates a fresh application key.
Result	<a href="#">digest</a> (String algorithm, byte[] data, <a href="#">Authentication</a> auth)	Processes a message digest from a given data blob.
Result	<a href="#">exportKey</a> ( <a href="#">KeyHandle</a> key, <a href="#">Authentication</a> auth)	Export key.
Result	<a href="#">getMasterKeyHandle</a> ( <a href="#">Authentication</a> requestAuth)	Provides the handle to the Master Key.
Result	<a href="#">importKey</a> ( <a href="#">KeyPair</a> softwareKey, <a href="#">Authentication</a> newKeyAuth, <a href="#">KeyHandle</a> parentKey, <a href="#">Authentication</a> authParentKey, <a href="#">KeyAttributes</a> attributes)	Imports an external <a href="#">KeyPair</a> into the protection of the TvSM.
Result	<a href="#">loadKey</a> ( <a href="#">Key</a> myKey, <a href="#">KeyHandle</a> keyHandleParent, <a href="#">Authentication</a> authForParentKey)	Loads a key and returns the handle.
Result	<a href="#">migrateKey</a> ( <a href="#">KeyHandle</a> key, <a href="#">KeyHandle</a> newParent, <a href="#">Authentication</a> auth)	Rewraps key with given new parent.
Result	<a href="#">sign</a> ( <a href="#">Digest</a> hash, <a href="#">KeyHandle</a> keyHandle, <a href="#">Authentication</a> auth)	Performs a cryptographic signing operation with specified key on the provided hash digest.
Result	<a href="#">timestamp</a> ( <a href="#">Datablob</a> data, String hashAlgorithm, <a href="#">KeyHandle</a> keyhandle, <a href="#">Authentication</a> auth)	Performs a cryptographic timestamping operation with the specified key on the specified input data.
Result	<a href="#">unloadKey</a> ( <a href="#">KeyHandle</a> handle, <a href="#">Authentication</a> auth)	Unloads a key handle from the TvSM.

# Typed Key Hierarchy



### 3.1 MaintenanceSession

Given a unique Identity Key  $k_{ID}$ , non-migratably bound to the TPM and using  $k_{ID}$  in the SKAP session.

$$\text{createMasterKey} : \mathbf{A} \xrightarrow[\text{new } k_M^{\text{pub}}, k_M^{\text{priv}}, k_M \cdot \mathbf{A} = \mathbf{A}]{\mathbf{A} \setminus \{MSTR, STO\}} h(n_M, k_M) \quad (1)$$

$$\text{getMasterKey} : \rightarrow k_M^{\text{pub}} \quad (2)$$

$$\text{migrateMasterKey} : k_{ID}^{\text{pub}} \xrightarrow{k_m \cdot \mathbf{A} \setminus \{MIG, MSTR\}} \text{enc}(k_M, k_{ID}^{\text{pub}}), \text{HMAC}_{k_M^{\text{priv}}}(\mathbf{A}) \quad (3)$$

$$\text{importMasterKey} : \text{enc}(k_M, k_{ID}^{\text{pub}}), \mathbf{A}, \text{HMAC}_{k_M^{\text{priv}}}(\mathbf{A}) \xrightarrow{\mathbf{A} \setminus \{MIG, MSTR\} \ \& \ \text{HMAC valid}} \quad (4)$$

$$\text{destroyMasterKey} : \rightarrow \quad (5)$$

Function definition from D-Y view:

$$\text{function} : \text{input} \xrightarrow[\text{internal state change}]{\text{conditions checked}} \text{output}$$

Key attributes  $\mathbf{A}$  are a set of  $MSTR, MIG, SIG, STO, TPM\_KEY, EXT$ . and size and algorithm information.  $MSTR$  is always  $STO$ .  $SIG$  is never  $STO$ .

### 3.2 OperatingSession

Given the Master Key  $k_M$  established in a maintenance session and using it in the SKAP session:

$$\text{createKey} : h(n_1, k_1), \mathbf{A} \xrightarrow[\text{new } k_2^{\text{pub}}, k_2^{\text{priv}}, k_2.\mathbf{A}=\mathbf{A}}{k_1.\mathbf{A}\text{-STO}} \text{enc}(k_2, k_1), \text{HMAC}_{k_2^{\text{priv}}}(\mathbf{A}, 1) \quad (6)$$

$$\text{loadKey} : \text{enc}(k_2, k_1), \text{HMAC}_{k_2^{\text{priv}}}(\mathbf{A}, 1), h(n_1, k_1) \xrightarrow[\text{new } n_2]{k_1.\mathbf{A}\text{-STO} \ \& \ \text{HMAC valid}} h(n_2, k_2) \quad (7)$$

$$\text{unloadKey} : h(n, k) \rightarrow \text{invalidates } h \quad (8)$$

$$\text{getMasterKeyHandle} : \rightarrow h(n_M, k_M) \quad (9)$$

$$\text{sign} : m, h(n, k) \xrightarrow{k.\mathbf{A}\text{-SIG}} \text{sig}(m, k^{\text{priv}}) \quad (10)$$

$$\text{importKey} : k_2, h(n_1, k_1), \mathbf{A} \xrightarrow[\text{new } n_2]{k_1.\mathbf{A}\text{-STO}} \text{enc}(k_2, k_1), \text{HMAC}_{k_2^{\text{priv}}}(\mathbf{A}, 1) \quad (11)$$

$$\text{exportKey} : h(n, k), \text{HMAC}_{k_2^{\text{priv}}}(\mathbf{A}, 1) \xrightarrow{k.\mathbf{A}\text{-EXT}} k \quad (12)$$

$$\text{migrateKey} : h(n_1, k_1), h(n_2, k_2) \xrightarrow{k_1.\mathbf{A}\text{-MIG} \ \& \ k_2.\mathbf{A}\text{-STO}} \text{enc}(k_1, k_2), \text{HMAC}_{k_1^{\text{priv}}}(\mathbf{A}, 2) \quad (13)$$

„Draft“

# Planned: Analysis

- *As only API is exposed, a verified secure API might imply a verified secure behavior.*

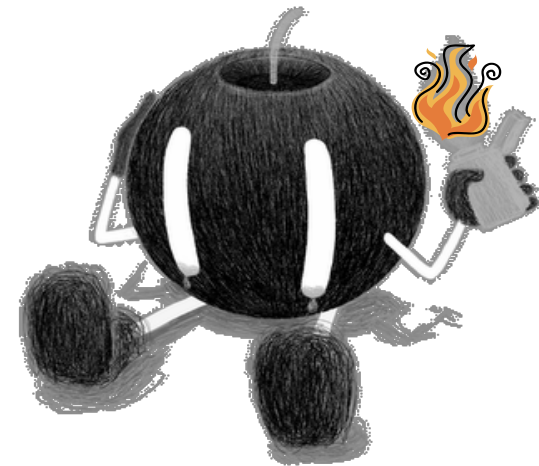
- No convenient languages or tools available

Still:

- We have a small, structured API
- Promising literature on PKCS#11 analysis..

# Limitations

- Many components involved
- Need to trust Intel, Linux and their implementations
- TPM does not protected against non-trivial HW attacks
- No temper-resistant hardware encasing for secure deletion of keys in case of attack



# Discussion

- We are proposing a **trusted** software module on top of a virtualization platform.
- Proof-of-concept platform available from <http://trustedjava.sf.net>
- Security Module and API under development