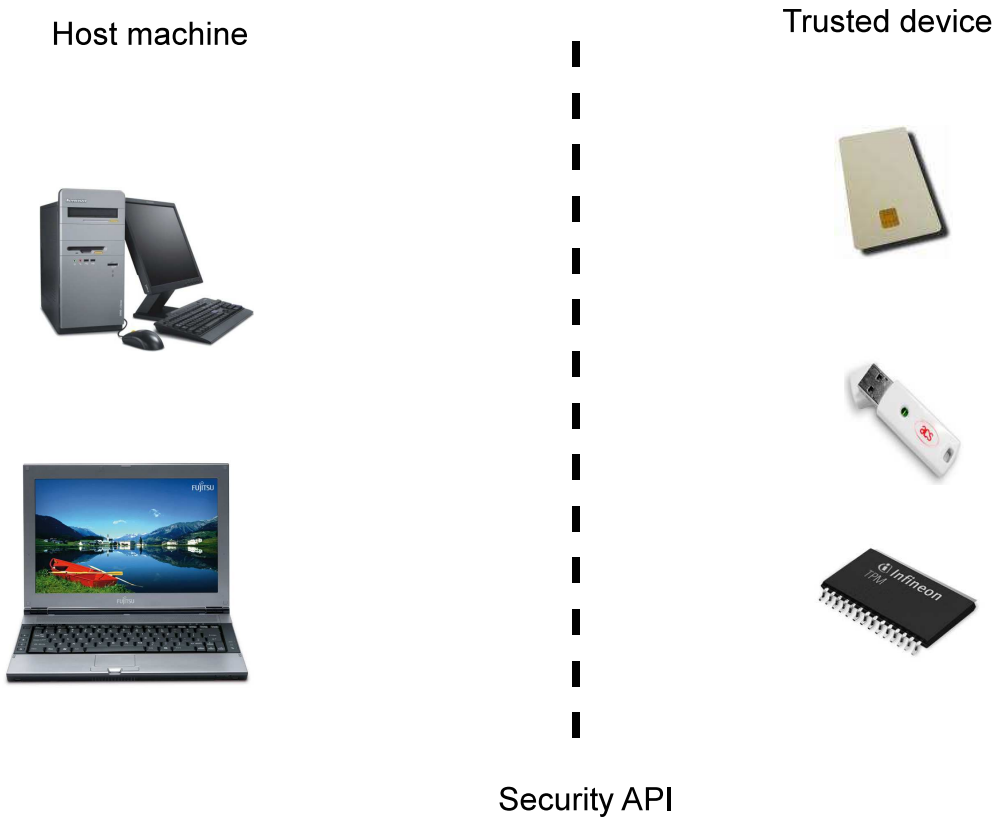
A Generic Security API for Symmetric Key Management on Cryptographic Devices

Véronique Cortier and Graham Steel
LORIA, Nancy (France) LSV, Cachan (France)

July 10th, 2009

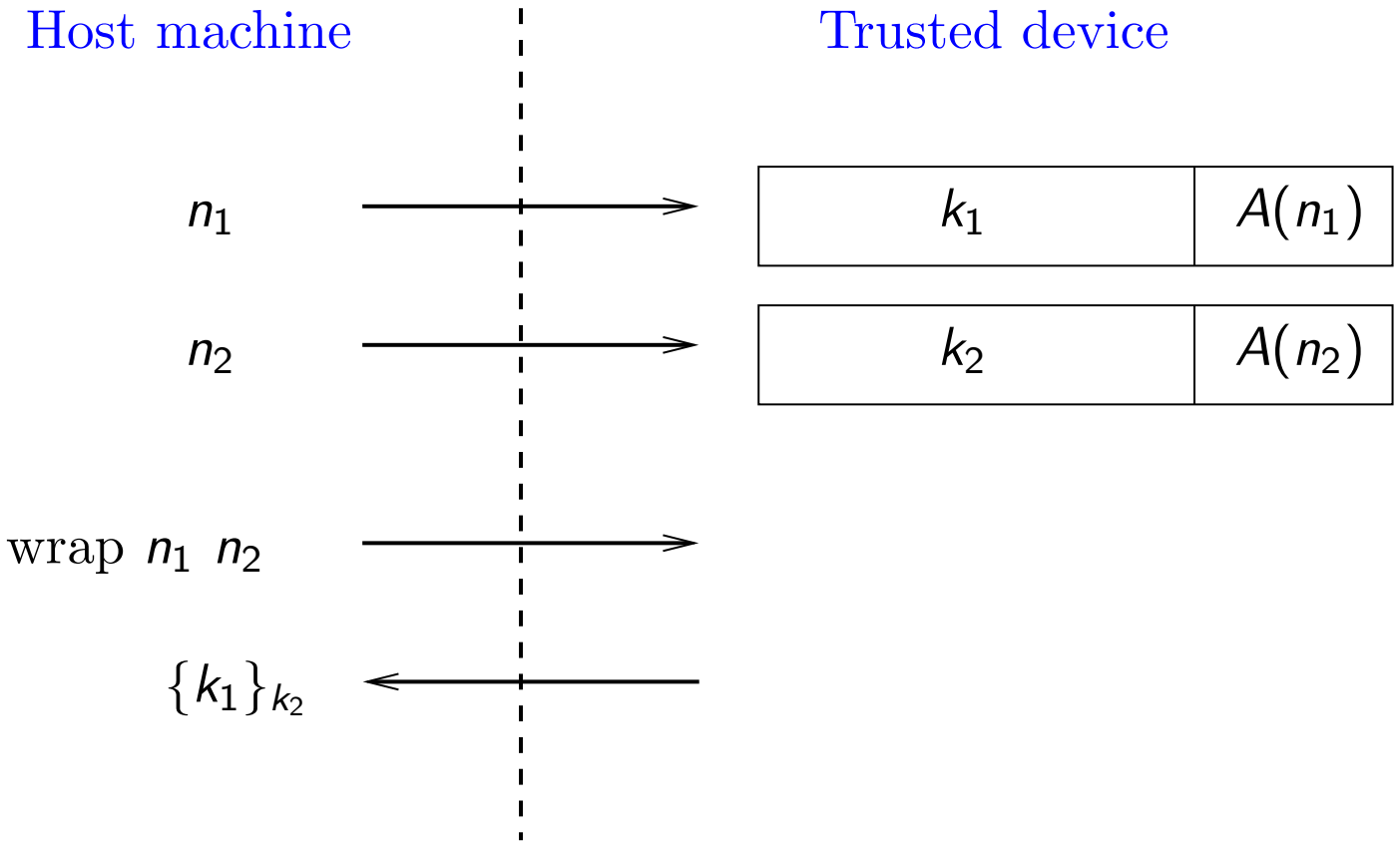
To appear at Esorics'09

Security APIs



Goal : Enforce security of data stored inside the trusted device, even when connected to untrusted host machines.

Example : PKCS#11



Key Separation Attack (Clulow, 2003)

Intruder knows : $h(n_1, k_1)$, $h(n_2, k_2)$.

State : $\text{wrap}(n_2)$, $\text{decrypt}(n_2)$, $\text{sensitive}(n_1)$, $\text{extract}(n_1)$

Wrap : $h(n_2, k_2)$, $h(n_1, k_1) \rightarrow \{k_1\}_{k_2}$

Decrypt : $h(n_2, k_2)$, $\{k_1\}_{k_2} \rightarrow k_1$

Key Separation Attack (Clulow, 2003)

Intruder knows : $h(n_1, k_1)$, $h(n_2, k_2)$.

State : $\text{wrap}(n_2)$, $\text{decrypt}(n_2)$, $\text{sensitive}(n_1)$, $\text{extract}(n_1)$

Wrap : $h(n_2, k_2), h(n_1, k_1) \rightarrow \{k_1\}_{k_2}$

Decrypt : $h(n_2, k_2), \{k_1\}_{k_2} \rightarrow k_1$

- Many variations of this attack exist, not easy to fix.
- Data are not related to identities.

Implementing protocols

Security APIs are used to implement endpoints of protocols

Idea :

- Abduct security API policy from suite of protocols it is supposed to implement
- Design generic API that can be instantiated to any protocol
- Prove properties for the generic API that hold no matter what protocol is implemented

Outline of the talk

Generic API : Concepts

Handles for objects stored on device

$$h(N, K, i, S)$$

K : secret data stored on the trusted device

N : value of the handle for K

Generic API : Concepts

Handles for objects stored on device

$$h(N, K, i, S)$$

K : secret data stored on the trusted device

N : value of the handle for K

i : Static security types for objects on device :

0. Public data (exists outside device)
1. Secret on device, not usable as key
2. Short term (session) key
3. Long term (key transport) key

Generic API : Concepts

Handles for objects stored on device

$$h(N, K, i, S)$$

K : secret data stored on the trusted device

N : value of the handle for K

i : Static security types for objects on device :

0. Public data (exists outside device)
1. Secret on device, not usable as key
2. Short term (session) key
3. Long term (key transport) key

S : Agent identifiers that specify part of the policy

$\longrightarrow$ Crucial for supporting compromised parties

Generic API : Generate Commands

$$\stackrel{N,K}{\Rightarrow} K_a(h_a^g(N, K, i, S)) \quad i \geq 1 \qquad \textbf{(Secure Generate)}$$

$$\stackrel{N,K}{\Rightarrow} K_a(K), K_a(h_a^g(N, K, 0, \mathsf{All})) \qquad \textbf{(Public Generate)}$$

where

- K_a local knowledge on the machine/agent a
- $N \in \mathsf{VarNonce}$,
- $K \in \mathsf{VarNonce}$ if $i = 1$,
- $K \in \mathsf{VarKey}$ if $i = 2$.

Generic API : Encrypt

$$K_a(h_a^{\alpha}(X, K, i_0, S_0)), K_a(x_1), \ldots, K_a(x_k),$$
$$K_a(h_a^{\alpha_1}(X_{n_1}, y_1, i_1, S_1)), \ldots, K_a(h_a^{\alpha_l}(X_{n_l}, y_l, i_l, S_l))$$
$$\Rightarrow K_a(\{x_1, 0, \ldots, x_k, 0, y_1, i_1, S_1, \ldots, y_l, i_l, S_l\}_K) \quad \textbf{(Encrypt)}$$

Require

- $i_0 > i_j$ (keys only encrypt data of strictly lower security level)
- $S_0 \subseteq S_j$

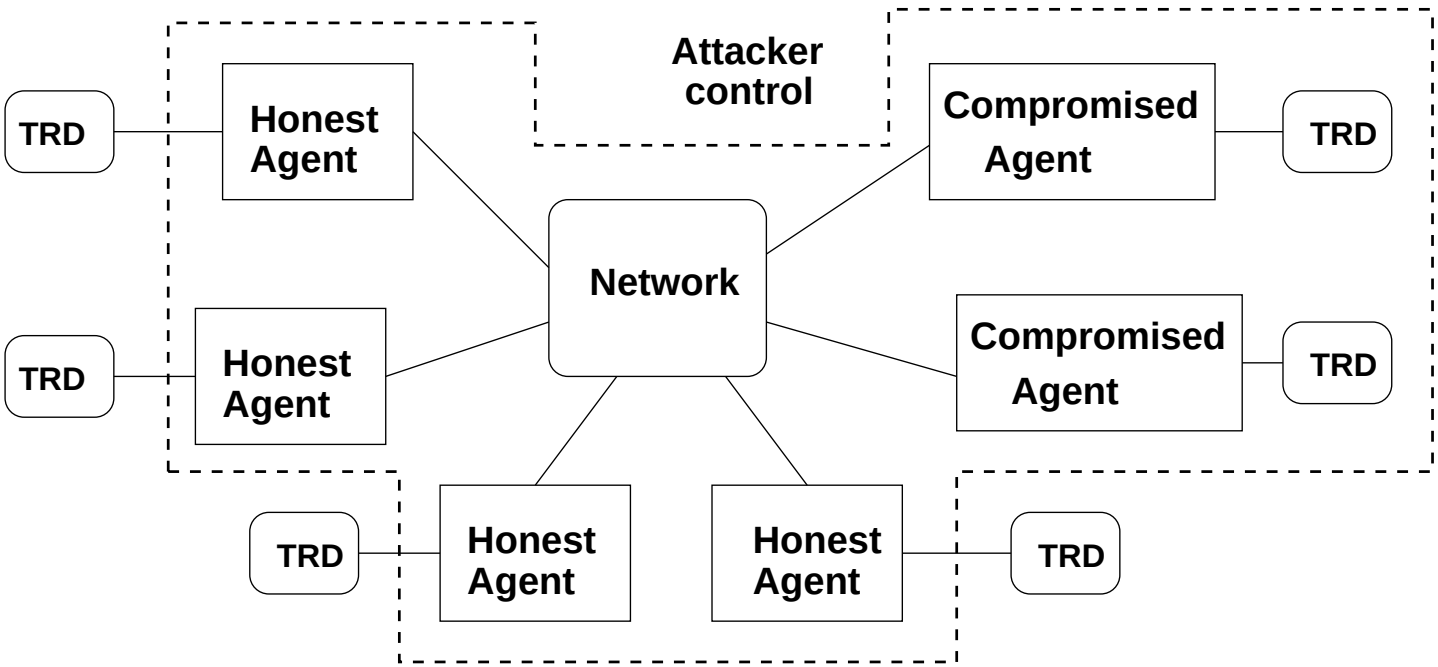
Generic API : Decrypt

$$K_a(h_a^{\alpha}(X, K, i_0, S_0)), K_a(\{x_1, 0, \ldots, x_k, 0, y_1, i_1, S_1, \ldots, y_l, i_l, S_l\}_K),$$
$$K_a(h_a^g(X_1, x_1, 0, \mathsf{All})), \ldots, K_a(h_a^g(X_s, x_s, 0, \mathsf{All})),$$
$$K_a(h_a^g(Y_1, y_1, i_1, S_1)), \ldots, K_a(h_a^g(Y_l, y_l, i_l, S_l))$$
$$\stackrel{N_{r+1},\ldots,N_l}{\Rightarrow} K_a(x_{s+1}) \ldots, K_a(x_k),$$
$$K_a(h^r(N_{r+1}, y_{r+1}, i_{r+1}, S_{r+1})) \ldots, K_a(N_l, y_l, i_l, S_l)$$
(Decrypt/Test)

provided

- $i_0 > i_j$ (keys only encrypt data of strictly lower security level)
- $S_0 \subseteq S_j$ Secure data will be accessible at most to agents in S, which are all authorized agents.

Attacker scenario



Attacker model

As usual, the attacker can apply cryptographic primitives.

$$
\begin{aligned}
K_{\text{int}}(x), K_{\text{int}}(y) &\Rightarrow K_{\text{int}}(\langle x, y \rangle) \\
K_{\text{int}}(\langle x, y \rangle) &\Rightarrow K_{\text{int}}(x) \\
K_{\text{int}}(\langle x, y \rangle) &\Rightarrow K_{\text{int}}(y) \\
K_{\text{int}}(x), K_{\text{int}}(y) &\Rightarrow K_{\text{int}}(\{x\}_y) \\
K_{\text{int}}(\{x\}_y), K_{\text{int}}(y) &\Rightarrow K_{\text{int}}(x)
\end{aligned}
$$

Attacker model

As usual, the attacker can apply cryptographic primitives.

$$
\begin{aligned}
K_{\text{int}}(x), K_{\text{int}}(y) &\Rightarrow K_{\text{int}}(\langle x, y \rangle) \\
K_{\text{int}}(\langle x, y \rangle) &\Rightarrow K_{\text{int}}(x) \\
K_{\text{int}}(\langle x, y \rangle) &\Rightarrow K_{\text{int}}(y) \\
K_{\text{int}}(x), K_{\text{int}}(y) &\Rightarrow K_{\text{int}}(\{x\}_y) \\
K_{\text{int}}(\{x\}_y), K_{\text{int}}(y) &\Rightarrow K_{\text{int}}(x)
\end{aligned}
$$

The attacker also controls any host machine (incl. honest ones).

$$
\begin{aligned}
K_a(x) &\Rightarrow I(x) \\
I(x) &\Rightarrow K_a(x) \\
K_b(h_b^{\alpha}(x, y, i, S)) &\Rightarrow I(y) \qquad \text{if } b \text{ compromised}
\end{aligned}
$$

Property 1 : Secrecy Invariant

"Secret data of honest users should not be known to the intruder."

i.e. data values k for which there are handles of the form $h_a^\alpha(n, k, i, S)$, where S is a subset of honest users, are unknown to the intruder.

$$\forall a \in \text{Agent}, \forall x, y \in \text{Msg}, \forall i \in \{1,2,3\}, \forall \alpha \in \{r, g\}, \forall S \subseteq H$$
$$\mathcal{S} \vdash h_a^\alpha(x, y, i, S) \Rightarrow \mathcal{S} \not\vdash y \quad \textbf{(Sec)}$$

Theorem

Sec is preserved by application of any of the API commands.

Property 1 : Secrecy Invariant

"Secret data of honest users should not be known to the intruder."

i.e. data values k for which there are handles of the form $h_a^\alpha(n, k, i, S)$, where S is a subset of honest users, are unknown to the intruder.

$$\forall a \in \text{Agent}, \forall x, y \in \text{Msg}, \forall i \in \{1,2,3\}, \forall \alpha \in \{r, g\}, \forall S \subseteq H$$
$$\mathcal{S} \vdash h_a^\alpha(x, y, i, S) \Rightarrow \mathcal{S} \not\vdash y \quad \textbf{(Sec)}$$

Theorem

Sec is preserved by application of any of the API commands.

Limitation : This does not guaranty any security in case some honest keys get compromised (e.g. using brute force attack).

Compromised keys

Assume the adversary has compromised some key K associated to some handle $h(N, K, 2, S)$ with S set of honest agents.

Consider $h(N', K', 1, S')$ with S' honest, $S \subseteq S'$

$$\text{Encrypt} \quad h(N, K, 2, S) \quad h(N', K', 1, S') \Rightarrow \{K'\}_K$$

Compromised keys

Assume the adversary has compromised some key K associated to some handle $h(N, K, 2, S)$ with S set of honest agents.

Consider $h(N', K', 1, S')$ with S' honest, $S \subseteq S'$

$$\text{Encrypt} \quad h(N, K, 2, S) \quad h(N', K', 1, S') \Rightarrow \{K'\}_K$$

$\longrightarrow$ The adversary can get any data of lower security level !

$\Rightarrow$ Short term keys should be regularly erased from the trusted device :

- They are short keys anyway
- Necessary to save the memory of the device

Freshness tests

Erasing compromised keys does not suffice !

Assume the adversary has compromised some key K associated to some (erased) handle $h(N, K, 2, S)$ with S set of honest agents.

And assume the adversary has seen $\{\ldots, K, 2, S, \ldots\}_{K_{ab}}$

Freshness tests

Erasing compromised keys does not suffice !

Assume the adversary has compromised some key K associated to some (erased) handle $h(N, K, 2, S)$ with S set of honest agents.

And assume the adversary has seen $\{\ldots, K, 2, S, \ldots\}_{K_{ab}}$

Decrypt $h(N', K_{ab}, 3, S')$ $\{\ldots, K, 2, S, \ldots\}_{K_{ab}} \Rightarrow h(N'', K, 2, S)$

And we are back to the previous case...

Freshness tests

Erasing compromised keys does not suffice !

Assume the adversary has compromised some key K associated to some (erased) handle $h(N, K, 2, S)$ with S set of honest agents.

And assume the adversary has seen $\{\ldots, K, 2, S, \ldots\}_{K_{ab}}$

Decrypt $h(N', K_{ab}, 3, S')$ $\{\ldots, K, 2, S, \ldots\}_{K_{ab}} \Rightarrow h(N'', K, 2, S)$

And we are back to the previous case...

$\Rightarrow$ The token needs to perform some freshness tests.

Property 2 : Secrecy After Compromise

- Intruder may learn some session keys
- Honest users erase old handles
- API enforces tests - no session keys accepted without a test

Theorem

Sec is still preserved.

Implementing protocols

1. Our API can implement all symmetric key management protocols of the Clark & Jacob library.
2. More generally, most of the protocols can be implemented provided that :
   - They are secure !
   - In case of nested encryption, nested cyphertext do not need to be secret.
   - Symmetric key protocols only

Benchmark

Protocol (section in Clark-Jacob)	API	API^r
Needham-Schroeder SK (6.3.1)	+	-
NSSK amended version (6.3.4)	+	+
Otway-Rees (6.3.3)	+	+
Yahalom (6.3.6)	+	-
Carlsen (6.3.7)	+	+
Woo-Lam Mutual Auth (6.3.11)	+	+

`http://www.lsv.ens-cachan.fr/~steel/genericapi/`

Related Work

Cachin and Tandran, presented at CSF'09 (on Wednesday 9th)

- Cryptographic model
- Symmetric and asymmetric encryption, key derivation
- But assume a unique token, having a global view of the system

Further Work

- Cryptographic soundness
- Long-term key update
- PKI (asymmetric crypto, certificates,...)
- Timestamps