SIEMENS

CORPORATE TECHNOLOGY

A Formal Security Model of the Infineon SLE88 Smart Card Memory Management

David von Oheimb, Volkmar Lotz
Siemens AG, Corporate Technology, Security

`{David.von.Oheimb,Volkmar.Lotz}@siemens.com`

Georg Walter
Infineon Technologies AG, Security & Chip Card ICs

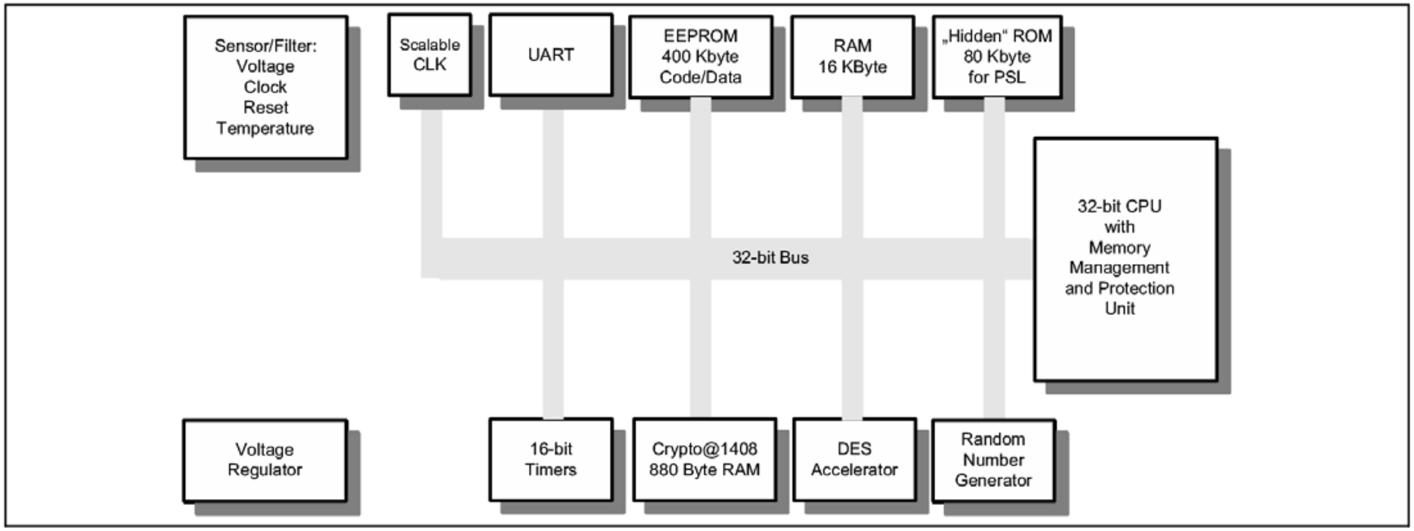
`Georg.Walter@infineon.com`

Information &
Communications
Security

Overview

- **Context**
- **SLE88 Memory Management**
  - Overview of Functionality
  - Security Objectives
- **SLE88 System Model**
- **Security Properties**
  - Enforcing access control through attributes
  - Protection of security-critical memory areas
- **Results**

32-bit smart card processor Infineon SLE88



- **Used for e.g. secure identification for UMTS and pay-TV**
- **Novelty of the SLE88: multi-application support**
- **New functionality: Memory Management Unit**
    - virtual address space
    - protection on both virtual and physical level
    - separation of packages

CORPORATE TECHNOLOGY

Context: SLE88 security

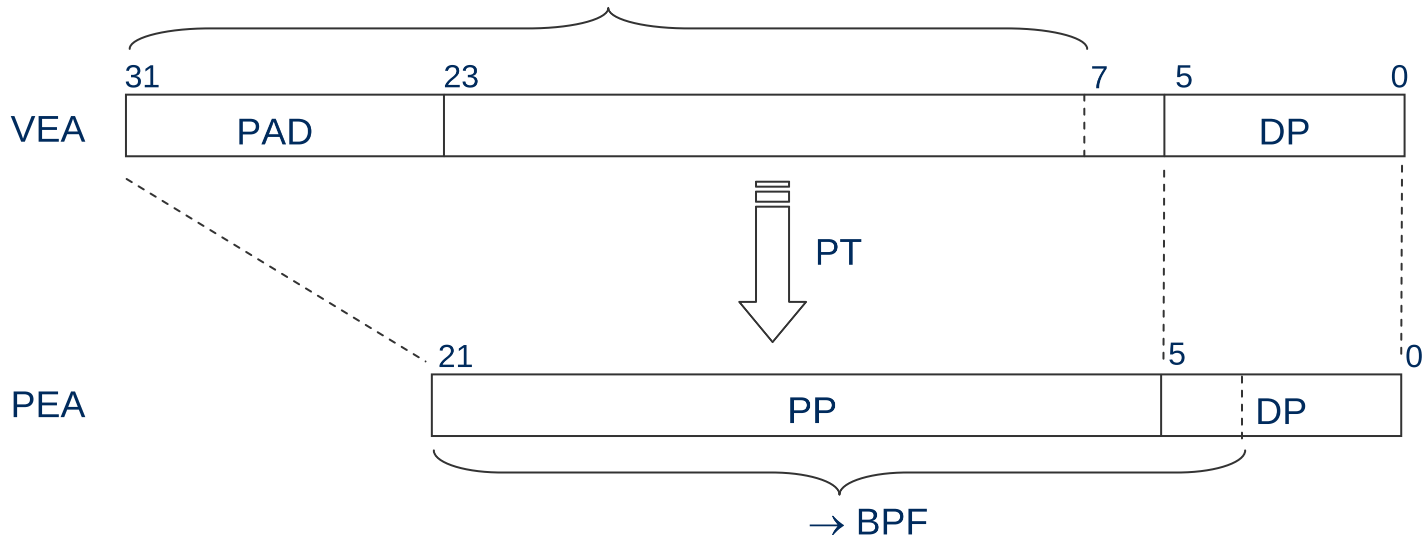
- **Certification of SLE88 according to Common Criteria EAL5+**
- **Existing LKW security model of SLE 66 [LKW00, vOL02] applies**
- **Additional security functionality for SLE88:**

  Memory Management Unit protects
  - Read/write/execute access to memory cells
  - Designated entry points to critical packages (“port commands”)
- **Intended to achieve security objectives:**
  - Restricted memory access
  - Separation of applications, OS, and chip security functionality (SL)
- **Augmenting the LKW model**

  with a separate memory management model suffices

Address Space

0	SL	privileged
1	PSL/HAL	privileged
2	OS	privileged
3..15	reserved	
16..255	regular	→ EAR



VEA	Virtual Effective Address	DP	Displacement
PEA	Physical Effective Address	PAD	Package Address
PT	Page Table	EAR	Effective Access Right
PP	Page Pointer	BPF	Block Protection Field

CORPORATE TECHNOLOGY

Information & Communications Security

Access Control Mechanisms

- ***Block Protection Field (BPF)***

  applies to 4-bit blocks of physical addresses

- ***Effective Access Rights (EARs)***

  apply to 8-bit blocks of virtual addresses

Source package:	Same as target		Other than target	
Target EAR	Read	Write	Read	Write
WW	+	+	+	+
WR	+	+	+	MPA/+
RR	+	MPA	+	MPA/+
W-	+	+	MPA/+	MPA/+
R-	+	MPA	MPA/+	MPA/+
X-	MPA	MPA	MPA/+	MPA/+

Security Requirements

The TOE must provide the Smartcard Embedded Software with the capability to define restricted access memory areas. The TOE must then enforce the partitioning of such memory areas so that access of software to memory areas is controlled as required, for example, in a multi-application environment.

- **Critical aspects:**
  - shared memory
  - modification of EAR table
  - protection achieved by BPF ("fail-safe"?)
  - port commands *(not shown here)*

System Model: SLE88 Memory

Formal definition of the virtual address space:

types *DP* — 6-bit displacement within *VEA*s and *PEA*s
= ...

typedecl *VEA_mid_lo* — 2-bit part of *VEA_mid* with identical EARs
typedecl *VEA_mid_hi* — 16-bit part of *VEA_mid* with different EARs
types *VEA_mid* — 18-bit middle part of *VEA*
= "$VEA\_mid\_hi \times VEA\_mid\_lo$"
VEA — 32-bit virtual effective address
= "$PAD \times VEA\_mid \times DP$"
VEA_dEAR — 24-bit upper part of *VEA* determining EARs
= "$PAD \times VEA\_mid\_hi$"
VP — 26-bit virtual page pointer
= "$PAD \times VEA\_mid$"

System Model: State

Formal definition of the system state:

- physical memory
- address translation
- access control settings
- execution state

```
record state =
— abstraction of physical memory:
  memory    :: "PEA      ⇒ value"  — including peripherals
  BPF_PASL  :: "PEA_dBPF ⇒ bool"   — BPF stating SL-only access to page blocks
— abstraction of page table (package descriptions and translation lookaside buffers):
  PT_map    :: "VP       ⇝ PP"     — page mapping, relative to packages
  PT_EAR    :: "VEA_dEAR ⇒ EAR"    — EARs for 256-byte sections
— abstraction of execution state:
  curr_PAD  :: "PAD"               — currently executing package
  stack     :: "PAD list"          — package part of return addresses
```

System Model: Inputs and Outputs

```
datatype message =
    Code_Fetch   VEA    — is meant to precede each other type of instruction
— read/write operations:
| Read_Mem      VEA
| Write_Mem     VEA value
— control transfer operations:
| Jump VEA
| Call VEA
| Return
| Write_RetAddr    VEA
— operations for setting security attributes and page table entries:
| Write_BPF_PASL   PEA_dBPF   bool
| Write_PT_EAR     VEA_dEAR   EAR
| Write_PT_map     VP         "PP option"

— outcome of operations:
| Ok | No   — access granted or denied without generating a trap
| MPA | MPSF | RLCP | MPBF | PRIV | MCR   — traps
```

System Model: Memory Access

Auxiliary function for checking access control conditions

Request for access **mode** at virtual address **va** in state **s** returns **Ok**, if:

- **va** is mapped to a physical address
- access is (privileged or) permitted according to EAR table
- BPF is consistently assigned (or special access by SL)

constdefs — read/write access restrictions to main memory

```
 mem_access :: "access_mode ⇒ state ⇒ VEA ⇒ message"
"mem_access mode s va ≡ case PEA s va of None ⇒ MPBF | Some pa ⇒
  let source = curr_PAD s; target = PAD va in
  (if is_Pri source ∧ mode≠Execute ∧ source≠target ∧ target≠Pri SL ∨
      mode ∈ (if source=target then RWX_own else RWX_other) (EAR s va)
   then (if ((BP_PASL s pa = (target = Pri SL)) ∨ BP_PASL s pa ∧
                    source = Pri SL ∧ mode ≠ Execute ∧ target ≠ Pri SL)
         then Ok else MPSF)
   else (if mode ≠ Execute then MPA else MPBF))"
```

CORPORATE TECHNOLOGY

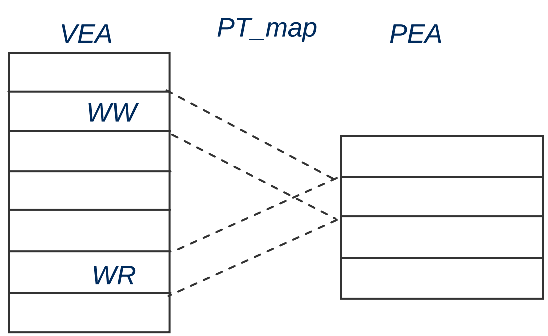
System Model: Interacting State Machine

```
ism SLE88_MM =
  ports interface
    inputs  "{In}"
    outputs "{Out}"
  messages message   — instructions received or indications of success sent
  states data state init "s0" name "s"   — the initial state is s0
  transitions

...

Read_Mem:
  in   In   "[Read_Mem va]"
  out  Out  "[mem_access Read s va]"
Write_Mem:
  in   In   "[Write_Mem va v]"
  out  Out  "[mem_access Write s va]"
  post memory := "(mem_access Write s va = Ok ∨
                  mem_access Write s va = MPSF ∧ belated_MPSF ∧
                  PAD va = Pri SL ∧ ¬BP_PASL s (the (PEA s va))) ?
                 (memory s)(the (PEA s va) := v)"
...
```

CORPORATE TECHNOLOGY

Security Properties (1): “Granted Accesses Do Respect EAR Settings”

$$
\begin{array}{l}
\textbf{theorem}\ \mathit{interpackage\_Write\_Mem\_respects\_EAR}\texttt{: "}\bigwedge \mathit{va}\ \mathit{va'}.\\
[\![((p,s),c,(p',s')) \in \mathit{Trans};\ \mathit{hd}\ (p\ \mathit{In}) = \mathit{Write\_Mem}\ \mathit{va}\ v;\ \mathit{hd}\ (p'\ \mathit{Out}) = \mathit{Ok};\\
\ \mathit{PAD}\ \mathit{va} \neq \mathit{curr\_PAD}\ s]\!] \Longrightarrow \mathit{is\_Pri}\ (\mathit{curr\_PAD}\ s) \wedge \mathit{PAD}\ \mathit{va} \neq \mathit{Pri}\ \mathit{SL} \vee\\
\ \mathit{EAR}\ s\ \mathit{va} = \mathit{WW} \wedge (\boxed{\mathit{EARs\_consistent}\ s} \longrightarrow \mathit{PEA}\ s\ \mathit{va'} = \mathit{PEA}\ s\ \mathit{va} \longrightarrow\\
\qquad\qquad\qquad\qquad \mathit{PAD}\ \mathit{va} \neq \mathit{PAD}\ \mathit{va'} \longrightarrow \mathit{EAR}\ s\ \mathit{va'} = \mathit{WW})\texttt{"}
\end{array}
$$



Consistency of EARs:

- In case of non-injective PT_map, the effective protection is determined by weakest EAR
- Conflicts are possible
- Should aliasing be prohibited?
- Solution: Define consistency requirements on EARs: all WW or all RR
- Property only holds in case of EAR consistency

Information & Communications Security

Security Properties (2): "Protection of SL Memory"

theorem *only_SL_changes_SL_memory:*
$$"\llbracket ts \in TRuns;\ ((p,s),c,(p',s')) \in set\ ts;\ curr\_PAD\ s \neq Pri\ SL;\ PEA\ s\ (Pri\ SL,\ lva) = Some\ pa \rrbracket \Longrightarrow memory\ s\ pa = memory\ s'\ pa"$$

Used lemmas (invariants):

- SL parts of page table and EAR table can only be modified by SL
- EARs referring to SL are always set in a way that access by non-SL packages is denied
- For SL memory areas, the BPF tag is always set

Required axioms (assumptions):

- Initial state satisfies requirements on BPF and initial EAR values
- Benign behaviour of SL (correct setting of BPF values, page table entries, and EAR table entries)

Conclusion

- **Identification**: necessary assumptions on initial state and behaviour of SL
- **Analysis**: effects of non-injective address mappings
- **Analysis**: role of block protection fields (BPF)
- **Proof**: security functionality is adequate to satisfy security requirements (on abstract level of specification)
- **Proof**: security specification is consistent (with some additional arguments referring to consistency of HOL)
- **Security model** satisfies all requirements of ADV_SPM.3 and thus contributes to EAL5 certification
- **Effort**: 2 person months