Re-TRUST
Partner 4: COSIC

Brecht Wyseur

Katholieke Universiteit Leuven

brecht.wyseur@esat.kuleuven.be

Presentation outline

- COSIC Research Group
- Re-TRUST Scenario
- Software Tamper Resistance
  - Code Obfuscation
  - White-Box Cryptography
- HW/SW Co-obfuscation
- Conclusions

COSIC

- K.U.Leuven, Belgium; Dept. of Electrical Engineering
- COmputer Security and Industrial Cryptography
- Founded in 1978
- Prof. B. Preneel; Prof. I. Verbauwhede; Prof. J. Vandewalle
- http://www.esat.kuleuven.be/cosic/
- 10 post-docs + 33 (phd-)researchers

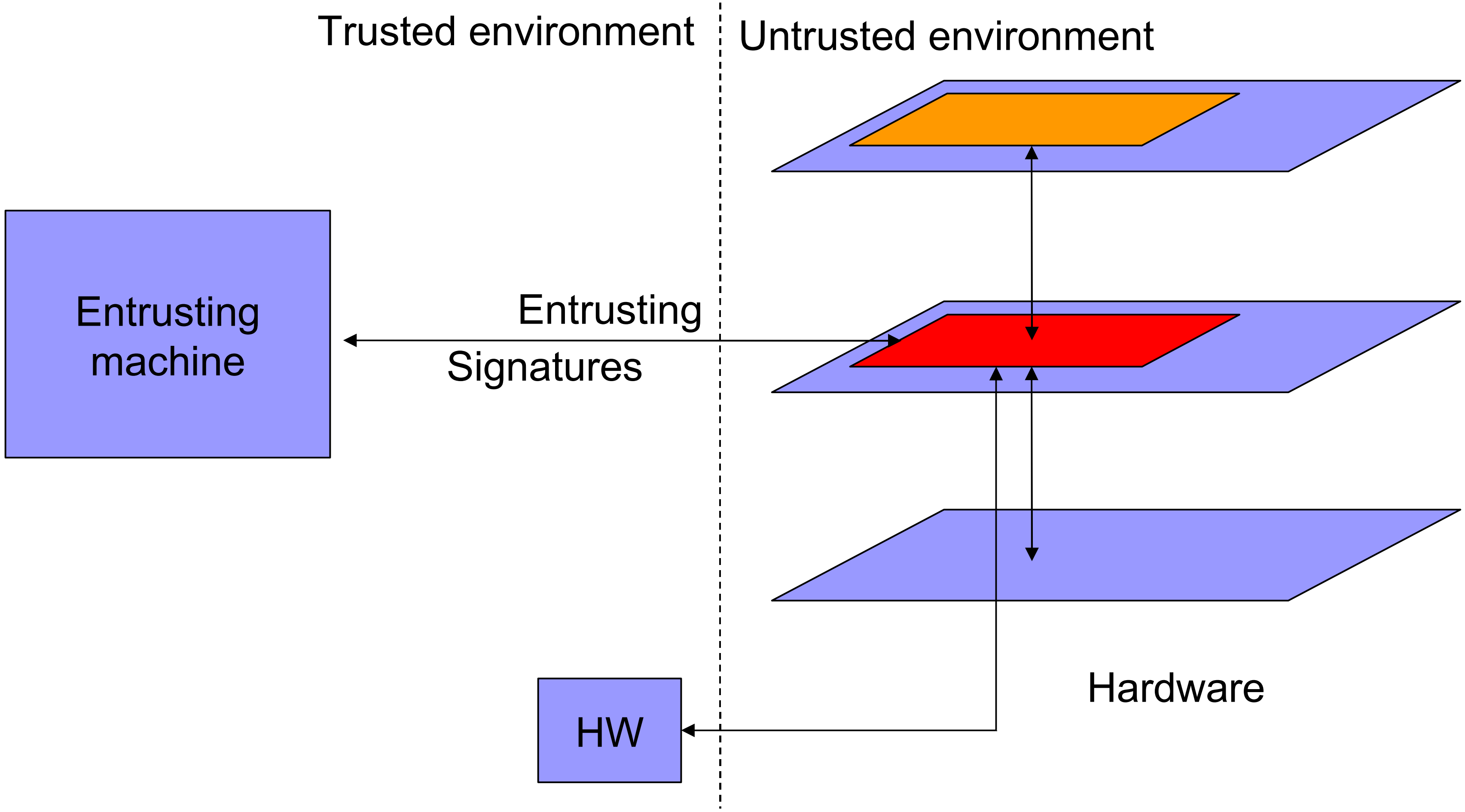
COSIC research activities (1)

- Cryptographic algorithms
  - Design (AES, RIPEMD-160, MQ-IP)
  - Cryptanalysis
  - Secure Implementations: hardware, software, HW/SW co-design, side-channel attacks, white-box crypto
- Protocols: key establishment, anonymous communications, broadcast encryption
- Fundamental research: Boolean functions, secret sharing, algebraic curves, multiparty computation

COSIC research activities (2)

- Privacy & Identity Management
- Trusted platforms and embedded systems
- Security in ubiquitous/pervasive systems
- Software obfuscation
- Document security, watermarking and perceptual hashing

Re-TRUST Scenario

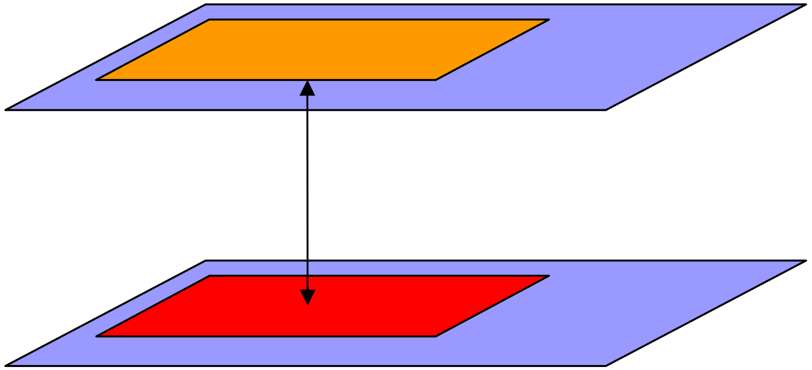


Problem: a malicious user has full access to the untrusted machine

SW Tamper resistance (WP2)

- T2.4 – Increased reverse engineering complexity for software protection
  - Hide software behaviour
    - Source-to-source obfuscation (C/C++)
    - Obfuscation of Java byte code
  - Hide encryption keys
    - White-box cryptography

Code obfuscation



- Goals
  - Interlocking of secure software module (which contains the trusted tag generator)
  - Counter reverse-engineering
  - Placement of dynamic updates
  - Taxonomy

Code obfuscation (2)

- Definition:

  Code obfuscation is *“applying one or more code transformations that make program analysis difficult”.*

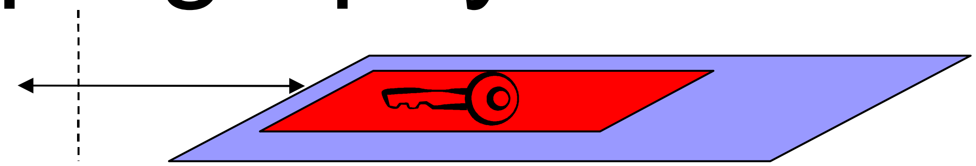
- Different abstraction levels:
  - Transform code: source - intermediate - binary code
  - Code analysis: source (e.g. C), binary (Assembly), …
- Different transformations
  - Layout - data flow - control flow - preventive [Collberg *et al.*]

Code obfuscation (3)

- Obfuscation metrics
  - *potency*, *resilience*, and *stealth* versus *cost* [Collberg *et al.*]
  - Others: ongoing research

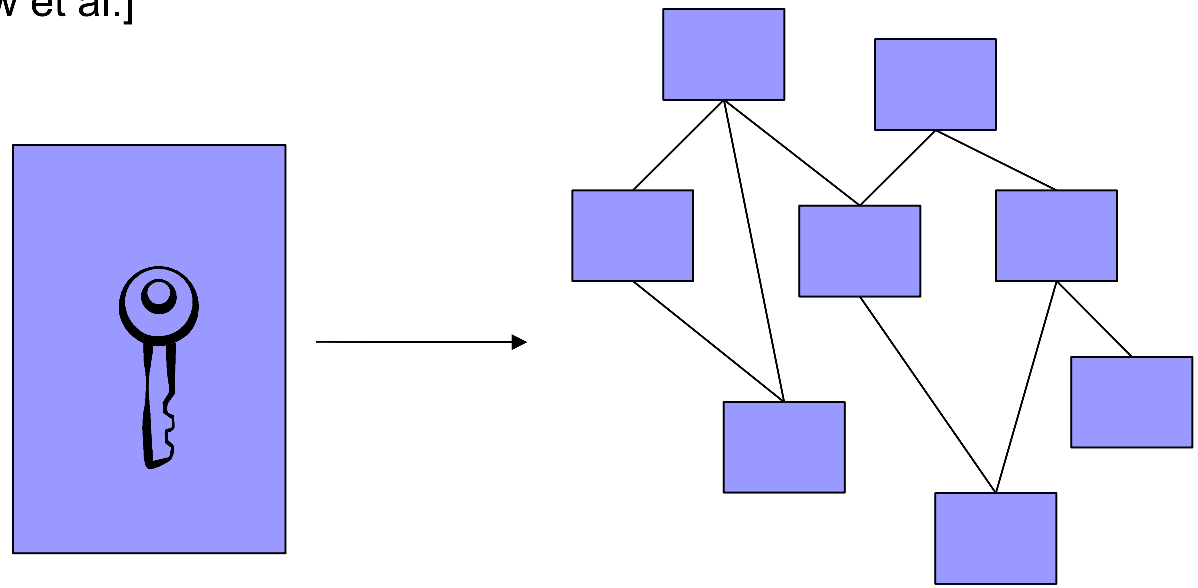
- COSIC expertise:
  - Source code (C/C++) obfuscation
  - Tamper resistance through self-encrypting code
  - …

White-box cryptography

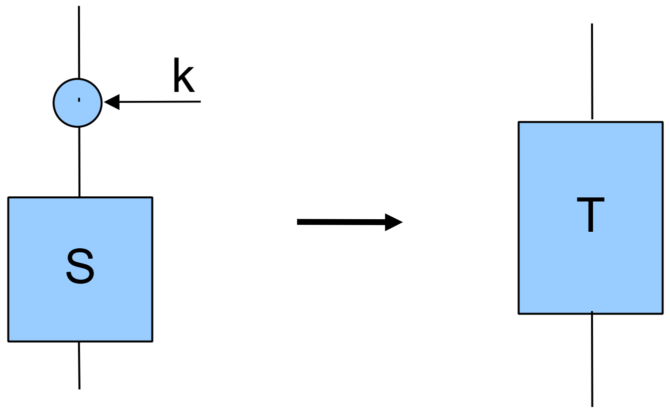


- Goal: hide embedded techniques
- Transform a cryptographic implementation into a series of key-dependant lookup tables.

[S. Chow et al.]



White-box cryptography (2)

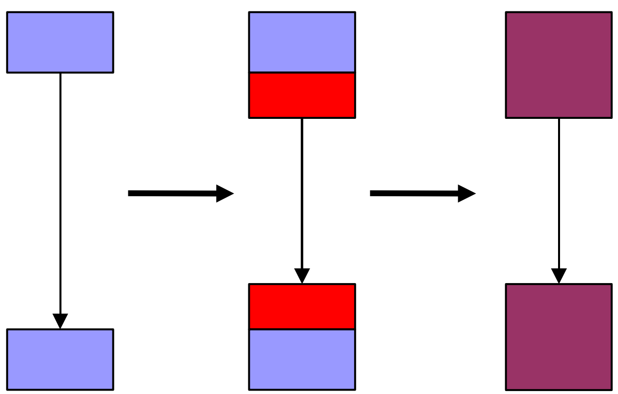


$T(x) = S(x + k)$

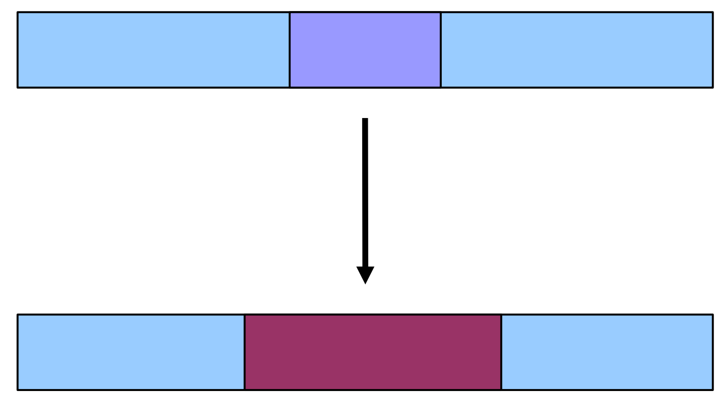
Encoding: $E' = F \circ E \circ G$

F,G random bijections

Internal Encodings



External Encodings



White-box cryptography (3)

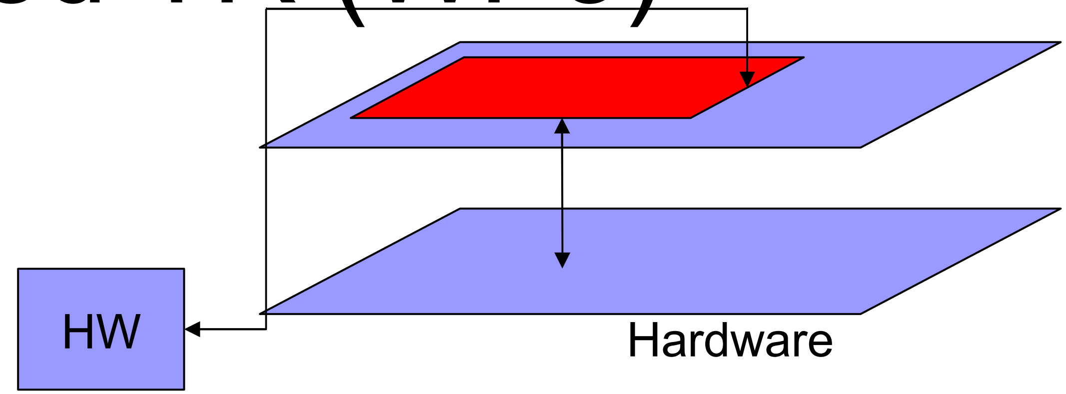
State of the art

- A White-Box DES Implementation for DRM Applications (DRM 2002)
  - Fault Injection Attack (Jacob et al. DRM 2002)
  - Statistical Bucketing Attack + Improved implementation (Link et al. ITCC 2005)
- White-Box AES (SAC 2002)
  - Cryptanalysis (O. Billet et al. SAC 2004)

White-box cryptography (4)

- Main interest for Re-TRUST
  - Hiding embedded keys
  - Challenge protection (against replay attacks)
- Other interests
  - Internal encodings: diversity (tracing)
  - External encodings: Interlocking
  - Avoid dynamic collusion attacks?
  - New paths: timing, dynamic updates, hardware

HW/SW-based TR (WP3)



- How to split functionality between hardware and software
- Trade-off
  - HW-cost
  - Performance
  - Security
  - Flexibility
- Tamper detection is hard when software can easily be replicated [Wurster et al.]

HW/SW-based TR (2)

- Research tracks
  - HW assisted software protection
  - Execution of small amount of code on HW
  - Protection of cryptographic operations
  - HW as a monitoring device
  - HW as an entrusting device
  - Assisting authentication: HW as an identification device, and to establish a symmetric key for communication with entrusting machine

Trusted Computing

- TCG Trusted Platform Module (TPM)
- Basic functionalities:
  - RSA (public key crypto algorithms)
  - Random number generator
  - HMAC
  - Secure hash: SHA1
  - Tick counter/clock
  - Monotic counter
- Attestation signing with a TPM certificate
- Problem: quite slow, limited applicability, limited flexibility (programmability)

Smart Cards

- Offer more flexibility
- Easier to use in cooperation with legacy devices
- Increasing storage capabilities
- Interesting emerging technology (wireless connection, Java/.NET VM, ...)
- For Re-TRUST
  - Symmetric key establishment to counter proxy problem
  - Can take over a few cryptographic operations
  - As an identification device

HW boards

- Less compact
- USB interface / serial port / parallel port
- Scalable
- For Re-TRUST
  - Extended smart card functionality
  - Can take a larger part of computations
  - Use as an entrusting device

Conclusions

- Presented technologies which can be deployed in Re-TRUST
  - Code Obfuscation
  - White-Box Cryptography
  - HW/SW Co-Obfuscation
- IP protection (TIVA, DRM 2005)