Towards Analysis of Various Protection Techniques

Brecht Wyseur, KULeuven
December 15th 2008, Paris
Overview

- Software protection techniques
  - Crypto guards
  - Obfuscation techniques
  - Fuzzing (analysis technique)
  - White-Box …

- Hardware assisted software protection techniques
  - Remote attestation with a TPM on a legacy OS
  - Physically Observable Cryptography (POC)
WP2 techniques

- Crypto guards
- Obfuscation techniques
- White-box …
Crypto guards

- **Construction [1]**

- **Goal:** protect software implementations against analysis and tampering.

- **On demand encryption**

- A *crypto guard* is a small piece of code, that dynamically decrypts code with a key derived from other code bytes.

- **Idea:** deploy a large network of nested code guards to make life of an adversary hard.

Crypto guards -- analysis

- Implementation on SPEC CPU2006 test suite
- Experiments to measure cost in execution time
  - 1. Bulk encryption
  - 2. On demand encryption

<table>
<thead>
<tr>
<th>Program</th>
<th>Total func</th>
<th>On demand</th>
<th>Speed cost</th>
<th># guard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mcf</td>
<td>22</td>
<td>20</td>
<td>1.09</td>
<td>28</td>
</tr>
<tr>
<td>Milc</td>
<td>159</td>
<td>146</td>
<td>8.17</td>
<td>543</td>
</tr>
<tr>
<td>Hmmer</td>
<td>234</td>
<td>184</td>
<td>3.20</td>
<td>873</td>
</tr>
<tr>
<td>Lbm</td>
<td>19</td>
<td>12</td>
<td>1.00</td>
<td>20</td>
</tr>
<tr>
<td>Sphinx_livepretend</td>
<td>210</td>
<td>192</td>
<td>6.65</td>
<td>1277</td>
</tr>
</tbody>
</table>
Crypto guards -- analysis

3. Performance vs. security trade-off

- Hot code heuristic
  hot code = code that is frequently called (k% times)
- Exp. 3: for k = 0.90: bulk encryption for hot code; on demand for remainder.

<table>
<thead>
<tr>
<th>Program</th>
<th>Total func</th>
<th>On demand</th>
<th>Speed cost</th>
<th># guard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mcf</td>
<td>22</td>
<td>19</td>
<td>1.04</td>
<td>24</td>
</tr>
<tr>
<td>Milc</td>
<td>159</td>
<td>135</td>
<td>1.95</td>
<td>486</td>
</tr>
<tr>
<td>Hmmer</td>
<td>234</td>
<td>183</td>
<td>1.15</td>
<td>862</td>
</tr>
<tr>
<td>Lbm</td>
<td>19</td>
<td>8</td>
<td>1.00</td>
<td>17</td>
</tr>
<tr>
<td>Sphinx_livepretend</td>
<td>210</td>
<td>181</td>
<td>1.72</td>
<td>1257</td>
</tr>
</tbody>
</table>
Crypto guards -- analysis

- Experiment results

Cost in execution time

<table>
<thead>
<tr>
<th>SPEC CPU2006</th>
<th>mcf</th>
<th>milc</th>
<th>hammer</th>
<th>lbm</th>
<th>sphinx_livepretend</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ct</td>
<td>1</td>
<td>8</td>
<td>3</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>

- Unprotected
- On-demand
- Combined

\( K = 0.90 \)
Obfuscation techniques

- **Goal:** make static and dynamic analysis difficult

- **Techniques**
  - Control Flow Flattening
  - Opaque predicates
Fuzzing [2]

- Submit random/unexpected data to an application, and monitor resulting errors
- Adaptive white-box testing technique

- In initial phase.
- Seems suitable to assess “invariants monitoring” techniques (invariants = constraints)

White-Box Cryptography

- Goal: implement cryptographic primitives in such a way that they remain secure in a white-box attack context.

- How to assess the security of WBC?
  - WBC techniques are very custom designed per primitive
  - Assess security of $O(E_k)$
  - $O(E_k)$ is secure $\Rightarrow$ $E_k$ is secure (in black-box context)

- Traditional assessment of security in cryptography
  - Direct proof (information theoretically secure)
  - Proof by reduction (to some hard problem)
  - Ad-hoc security
Security analysis of WBC

Black-Box

- Ad-hoc security
  - Block ciphers
- Process of scrutinizing
  - Cryptanalysis
  - Design criteria (S-boxes, avalanche effect, diffusion properties, MDS, …)

White-Box

- White-boxed block cipher
- Metrics (diversity, ambiguity)
- Process of scrutinizing
  - Cryptanalysis
  - WB design criteria (differential properties, no MDS, …)
## Security analysis of WBC

<table>
<thead>
<tr>
<th>Black-Box</th>
<th>White-Box</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Proof by reduction</strong></td>
<td><strong>Symmetric ciphers from</strong></td>
</tr>
<tr>
<td></td>
<td><strong>asymmetric crypto</strong></td>
</tr>
<tr>
<td>Typical for asymmetric crypto</td>
<td><strong>(cheating?)</strong></td>
</tr>
<tr>
<td>Provably secure based on some</td>
<td>Model: def. obfuscation + context</td>
</tr>
<tr>
<td>problems <em>believed to be hard.</em></td>
<td>[3].</td>
</tr>
<tr>
<td>Define model + attack goals:</td>
<td>P = NP?</td>
</tr>
<tr>
<td>security notions</td>
<td></td>
</tr>
</tbody>
</table>

**Direct proof**

White-Box Remote Program Execution

**Framework:**

```
  Server

  ε(P, i)

  ε(P(a, i))

  OVM

  client

  a
```

**Goals**

- *Obfuscated Virtual Machine (OVM)* able to execute generic programs (note: Barak et al.; Goldwasser et al. do not apply)
- program obfuscation as secure as underlying cipher
  
  E.g.: level of “trust” in integrity of execution: \(1 - 2^{-m}\), where
  
  \[\varepsilon : \text{GF}(2)^n \rightarrow \text{GF}(2)^{n+m}\]
WBRPE – security analysis

- Problem: the *Obfuscated Virtual Machine (OVM)* leaks *EVERY* computation (CPU and memory calls)

- How to make a secure OVM?
  - From SFE (as presented at RE-TRUST 2008)
    - Problem: size for *reasonable* circuits.
  - Create a custom secure building block (towards a TM)
    - Then, composing building blocks
    - We are able to construct a secure VM for a narrow set of circuits
    - Generalizing: universally composable cryptography (Canetti 2001)
  - In practice (for now) – augment a VM (e.g., JVM)
    - Deploy obfuscation techniques
WP3 techniques

Remote attestation

Physically observable cryptography
Remote attestation with a TPM

Framework [4]

Analysis

Requirements

- Requirements on PIONEER (unpredictable, optimal checksum function, unpredictable random walk through memory); TEAS (unpredictable, well obfuscated checksum function)
- Trusted bootloader; trusted clock ticker

Efficiency

Physically Observable Cryptography

- Goal: model a side-channel adversary, and attempt to obtain (provable) security on circuit implementations

Models
- Micali & Reyzin
  - Reduction proofs
  - Problem: for each extension, new assumptions required
- Ishai, Sahai, and Wagner
  - Private circuits I: probing attacks
  - Private circuits II: tamperable circuits
  - Problem: realistic assumptions?

Future research
- Improved models
- New constructions
Physically Observable Cryptography

- Micali & Reyzin model
  - Secure basic primitive
  - Reduce security of other constructions to security of the basic primitive

- Micali & Reyzin studied basic theoretic constructions
  - (PO) OWF → (PO) PRNG
  - Disadvantage: inefficient, not used in practice

- KUL
  - Study of practical constructions (RSA-CPA; RSA-OAEP; RSA-FDH)
  - Problem: requirements needed for each step

- Future work
  - Change model
  - Develop new schemes (not likely; will face similar problems)
Physically Observable Cryptography

- Ishai-Sahai-Wagner model ("Private circuits I")
  - Boolean circuit implementation
  - Adversary can probe $t$ wires
  - $t$-security: adversary does not gain any information

- Construction
  - Based on secret sharing
  - Result: circuit with $O(nt^2)$ gates

- Problem: controversial model – ‘normal’ adversaries do not probe, but measure power consumption
Conclusion

- Assess the security of techniques
  - Metrics, empirical studies, fuzzing
    - Obfuscation techniques
  - Scrutinizing
    - White-box implementations of block ciphers
    - Hash functions (remote attestation)
  - Provable security (reduction proofs)
    - White-box implementations from asymmetric primitives
    - White-Box Remote Program Execution (WBRPE)
    - Physically Observable Cryptography (POC)

- Assumptions
  - Trusted TPM, Trusted bootloader
  - Model (obfuscation definition; leakage model)
Conclusions

- Our main question: how well are our defenses against ‘real-world’ adversaries?