POLEN: Combining Polymorphism and Program Encryption to Guarantee Data and Code Confidentiality

Lionel Morel – joint work with Nicolas Belleville, Damien Couroussé, Irenée Groz.
Context - Motivations

- Side-channel attacks
  - Hiding
  - Masking

- Fault attacks
- Fault Tolerance
- Fault Detection

SW/HW co-design of Security Solutions:

- Complementarity of HW ans SW solutions
- Systematic application of SW counter-measures
  - Work within the compiler
Polen: HW and Compiler to address SCA + Reverse

**Side-channel attacks**

**POLYMORPHISM**: Automatic hiding based on dynamic code generation

**Reverse-engineering**

**ENCRIPTION**: Compile-time program encryption + HW support for decryption
Code Polymorphism

- Principle: regularly change the behavior of a component, at runtime, while maintaining unchanged its functional properties.

POLLEN : Current Status
Polen: Current Status

- **Cogito-odo** now targets **RISCV ISA (+ARMv7)**
- **Illvm-RISCV** back-end generating **encryption-ready** binaries
- Standalone **binary encryptor**
- **HW decryption** added to the **Spike** Instruction Set Simulator
- Currently working on the encryption of polymorphic instances

<table>
<thead>
<tr>
<th></th>
<th>SCA</th>
<th>Static Reverse</th>
<th>Dynamic Reverse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encrypted Program</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Code Polymorphism</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Encrypted program + Polymorphism</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>