Analysing the Leakage-Resistance of the NIST's Lightweight Crypto Standardization Process Finalists

C. Verhamme, G. Cassiers, F.-X. Standaert

UCLouvain, ICTEAM, Crypto Group (Belgium)

NIST Lightweight Crypto Workshop 2022, Virtual
• Introduction/methodology

• Model-level analysis of all finalists

• Interest of levelled implementations

• Hardware design space exploration
  • For Ascon, ISAP and Romulus-T only

• Conclusions (take home messages)
• Introduction/methodology
• Model-level analysis of all finalists
• Interest of levelled implementations
• Hardware design space exploration
  • For Ascon, ISAP and Romulus-T only
• Conclusions (take home messages)
Security targets

• Confidentiality: security against CCA Adv.
• Integrity: Ciphertext Integrity (CI)
  • Composite definitions useful: confidentiality & integrity often call for \( \neq \) physical assumptions
• Leakage in encryption only (1) or enc./dec. (2)
• Nonce misuse-resistance (M) or resilience (m)
• Leakage-resistance (L) or resilience (l)

\[ \Rightarrow \] Choice of security target depends on application
Mode analysis (I)

- Identify main steps, e.g., inner keyed sponge

- (Some steps empty for some modes, ignoring AD)
• Reduce the mode to (weak) assumptions (tightly)

only computation leaks
leak-free components bounded leakage
strong unpredictability with leakage
simulatable leakages hard-to-invert leakages
oracle-free leakages [...]

Mode analysis (II)
Practical evaluation (I)

- Translate assumptions into necessary design goals

<table>
<thead>
<tr>
<th>conf.</th>
<th>KDF/TGF</th>
<th>bulk comp.</th>
<th>tag verif.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DPA (key recovery)</td>
<td>DPA (key recovery) SPA (key recovery)</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-block conf.</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>int.</td>
<td>DPA (key recovery)</td>
<td>DPA (key recovery) SPA (key recovery) unbounded leakages</td>
<td>DPA (tag recovery) unbounded leakages</td>
</tr>
</tbody>
</table>

- Set the target security level ($2^m$ leakages, $2^t$ time)
- *Evaluate implementation cost & performances*
### Practical evaluation (II)

- **Approximate performance overheads**

<table>
<thead>
<tr>
<th></th>
<th>KDF/TGF</th>
<th>bulk comp.</th>
<th>tag verif.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>conf.</strong></td>
<td>$x5 - 10 - 100$</td>
<td>$x5 - 10 - 100$</td>
<td>$x1 - 5$</td>
</tr>
<tr>
<td><strong>int.</strong></td>
<td>$x5 - 10 - 100$</td>
<td>$x5 - 10 - 100$</td>
<td>$x5 - 10 - 100$</td>
</tr>
</tbody>
</table>

- **DPA security**: high-order masking, shuffling, ...
- **SPA security**: parallel implementations, noise, ...
• Introduction/methodology
• Model-level analysis of all finalists
• Interest of levelled implementations
• Hardware design space exploration
  • For Ascon, ISAP and Romulut-T only
• Conclusions (take home messages)
• Example: Romulus-N
  • Integrity: CIL1, CIML1, CIML2

• DPA resistance is needed everywhere, even for the weakest side-channel security target
Level-0 (no mode-level resistance)

- Example: Romulus-N
- Confidentiality: CCAL1, CCAmL1, CCAmL2

- DPA resistance is needed everywhere, even for the weakest side-channel security target

- Similar: Elephant, GIFT-COFB, Tiny-Jambu
• Example: PHOTON-Beetle
• Integrity: CIL1
Example: PHOTON-Beetle

Integrity: CIML1, CIML2

DPA resistance is needed everywhere if misuse or leakage in decryption are exploitable
• Example: PHOTON-Beetle
• Confidentiality: CCAL1
• Example: PHOTON-Beetle
  • Confidentiality: CCAmL1, CCAmL2

• DPA resistance is needed everywhere if misuse or leakage in decryption are exploitable

• Similar: Sparkle, Xoodyak
• Example: Ascon
  • Integrity: CIL1, CIPL2, CIML2

• Top of the hierarchy (for mode-level protections)
• Example: Ascon
  • Confidentiality: CCAL1, CCAmL1
• Example: Ascon
  • Confidentiality: CCAmL2

• Message decrypted before verification
Recovering the ephemeral key with DPA allows an adversary to recover the message in full
• Example I: Romulus-T
  • Integrity: CIL1, CIML1, CIML2
Level-3 (L2 + two passes)

- Example I: Romulus-T
- Confidentiality: CCAL1, CCAmL1, CCAmL2

Note: SPA without averaging for CCAL1 & CCAmL1 with averaging for CCAmL2
• Example II: ISAP
  • Integrity: CIL1, CIML1, CIML2
Level-3 (L2 + two passes)

- Example II: ISAP
- Confidentiality: CCAL1, CCAmL1, CCAmL2

Note: SPA without averaging for CCAL1 & CCAmL1 with averaging for CCAmL2
Outline

• Introduction/methodology
• Model-level analysis of all finalists
  • Interest of levelled implementations
• Hardware design space exploration
  • For Ascon, ISAP and Romulut-T only
• Conclusions (take home messages)
Leveled implementation gains

- In software: reflected in the cycle count
• In hardware: reflected in the energy/byte

⇒ Gains by factors > 10 for long messages / high security
Outline

• Introduction/methodology
• Model-level analysis of all finalists
• Interest of levelled implementations
• Hardware design space exploration
  • For Ascon, ISAP and Romulut-T only
• Conclusions (take home messages)
Hardware design space exploration

- ISAP-1, ISAP-2
  - RK: round-based Ascon permutation
  - Bulk: 1x or 2x round-based Ascon permutation

- Ascon-1/4, Ascon-1
  - KDF/TGF: 80-bit masked permutation, HPC2 [A]
    - S-box: 5 AND gates in parallel in 2 cycles
  - Bulk: 80-bit or 1-round Ascon permutation

- Romulus-T-1, Romulus-T-4
  - KDF/TGF: 128-bit masked Skinny TBC, HPC2 [A]
    - S-box: 2 AND gates in parallel in 6 cycles
  - Bulk: 1x or 4x round-based Skinny TBC

• Ascon & Romulus-T area dominated by masked KDF/TGF
• Moderate cost of levelling (mode could be optimized)
Latency comparisons

- As the message size increases, mostly impacted by the unprotected bulk computations & its tradeoffs
• Introduction/methodology
• Model-level analysis of all finalists
• Interest of levelled implementations
• Hardware design space exploration
  • For Ascon, ISAP and Romulut-T only
• Conclusions (take home messages)
• For similar security, Ascon is more efficient than Romulus-T (at the cost of not providing CCAmL2)
• ISAP security not directly comparable
  • Our guess: hard to attack in parallel hardware
    • More challenging in serial software
For similar security, Ascon is more efficient than Romulus-T (at the cost of not providing CCAML2)
• ISAP security not directly comparable
  • Our guess: hard to attack in parallel hardware
  • More challenging in serial software

3 efficient designs for side-channel security
• Hardware design choices matters a lot!
  • Leads to stronger differences than the primitives
• Security margins are not the same!
  • E.g., CML2 for Ascon with $p^b = 6$ rounds
• For similar security, Ascon is more efficient than Romulus-T (at the cost of not providing CCAmL2)
• ISAP security not directly comparable
  • Our guess: hard to attack in parallel hardware
  • More challenging in serial software
• 3 efficient designs for side-channel security
• Hardware design choices matters a lot!
  • Leads to stronger differences than the primitives
• Security margins are not the same!
  • E.g., CIML2 for Ascon with \( p^b = 6 \) rounds

⇒ Our view: should not drive NIST selection
Qualitative

• Willing CCaml2 security in 2 passes (vs. 1-pass)
  • Yes: ISAP or Romulus-T
  • No: Ascon

• Willing flexible overheads (vs. always on)
  • Yes: Ascon or Romulus-T
  • No: ISAP

• Willing a single algorithm (vs. a suite)
  • Yes: Ascon or ISAP
  • No: Romulus

(Note: short messages require separate treatment...
THANKS!
• **SPARKLE:**

![Diagram of SPARKLE process]

• **Bad TGF (from the leakage viewpoint)**
  • Final key addition creates a DPA target