New Results and Insights on ForkAE

Elena Andreeva¹  Arne Deprez²  Jowan Pittevils²  
Arnab Roy¹  Amit Singh Bhati²  Damian Vizár⁵

Alpen-Adria University Klagenfurt, Austria
imec-COSIC, KU Leuven, Belgium
CSEM, Switzerland

NIST LWC workshop 2020
0. ForkAE: recap

1. Cryptanalysis of ForkSkinny

2. Implementation results

3. SAEF: security update

4. Extending the use case

  + New forkcipher encryption modes
ForkAE: Forkcipher

≈ Two parallel TBC calls at lower cost

iterate-fork-iterate the well-cryptanalyzed SKINNY components

⇒ \((r_{\text{init}}, r_0, r_1)\) configuration with \(r_0 = r_1\)

| Primitive F            | \(n\) | \(t\) | \(t + |K|\) |
|------------------------|------|------|-----------|
| ForkSkinny-64-192      | 64   | 64   | 192       |
| ForkSkinny-128-192     | 128  | 64   | 192       |
| ForkSkinny-128-256     | 128  | 128  | 256       |
| ForkSkinny-128-288     | 128  | 128  | 288       |
ForkAE: PAEF

\[ N^{|000|} \]  \( A_1 \)  \( F_K \)  \( 0^n \)  \( T_A \)

\[ N^{|000|} \]  \( A_2 \)  \( F_K \)  \( \ldots \ldots \)  \( T_A \)

\[ N^{|100|} \]  \( M_1 \)  \( F_K \)  \( T_A \)  \( C_1 \)

\[ N^{|100|} \]  \( M_2 \)  \( F_K \)  \( \ldots \ldots \)  \( C_m \)

\[ N^{|100|} \]  \( M_m \)  \( F_K \)  \( \ldots \ldots \)  \( C_n \)

\[ N^{|001|} \]  \( A_a \)  \( F_K \)  \( T \)

\[ N^{|100|} \]  \( M_a \)  \( F_K \)  \( T \)

\[ n \]-bit AE security

\[ \text{Adv}^{\text{privacy}}_{\text{PAEF}}(A) \leq \text{Adv}^{\text{PRFP}}_F(D) \]

\[ \text{Adv}^{\text{auth}}_{\text{PAEF}}(A) \leq \text{Adv}^{\text{PRFP}}_F(D) + \frac{q_v \cdot 2^n}{(2^n - 1)^2} \]
ForkAE: SAEF

\[
\begin{align*}
\text{Adv}_{\text{SAEF}}^\text{privacy}(A) & \leq \text{Adv}_F^{\text{PRFP}}(D) + 2\frac{(\sigma - q)^2}{2^n} \\
\text{Adv}_{\text{SAEF}}^\text{auth}(A) & \leq \text{Adv}_F^{\text{PRFP}}(D) + \frac{2(\sigma - q + 1)^2}{2^n} + \frac{\sigma(\sigma - q)}{2^n} + \frac{q_v(q + 2)}{2^n}
\end{align*}
\]

Andreeva, Deprez, Pittevils, Roy, Singh, Vizár
New Results and Insights on ForkAE
NIST LWC workshop 2020
Cryptanalysis

Status of ForkSkinny

- No weakness till date from publicly known cryptanalysis
- It continues to benefit from the security margin of SKINNY
- The best attack on SKINNY covers \( \approx 50\% \) of the total nr of rounds
Cryptanalysis

Status of ForkSkinny

- No weakness till date from publicly known cryptanalysis
- It continues to benefit from the security margin of SKINNY
- The best attack on SKINNY covers $\approx 50\%$ of the total nr of rounds

ForkSkinny cryptanalysis (Bariant et al. ToSC 2020)

- ForkSkinny-128-256 (128-bit tweak, 128-bit key): 24 out of 48 rounds
- ForkSkinny-128-256 (no tweak): 26 rounds attacked
  - Not part of the ForkAE family
Cryptanalysis

General cryptanalysis (of forkcipher)

- ForkSkinny does not have the weaknesses of ForkAES
  - Reconstruction queries: a specific of forkciphers
  - ForkAES had a weakness wrt to these, cryptanalysis exploited it
  - ForkSkinny does not have such reconstruction query weakness
Cryptanalysis

General cryptanalysis (of forkcipher)

- ForkSkinny does not have the weaknesses of ForkAES
  - Reconstruction queries: a specific of forkciphers
  - ForkAES had a weakness wrt to these, cryptanalysis exploited it
  - ForkSkinny does not have such reconstruction query weakness

Remarks

- Reduced round instances should have $r_0 = r_1$
- ForkSkinny has comfortable security margin
  - The nr of rounds can be reduced by $\geq 5$, i.e. $r_0 = r_1 = 26$.
  - We are currently exploring further reduction
Portable SW implementations

- We started with: constant-time implementations at https://github.com/rweather/lightweight-crypto

- Improved decryption with preprocessed TKS:
  - 38% less clock cycles
  - 1kB smaller ROM size
  - 252-696 bytes higher RAM usage
Table-based SW implementations

• Suitable for platforms without a cache, e.g. Cortex-M0

• Round function $\rightarrow$ 18 lookups $+$ 19 XOR

• Performance on Cortex-M0 (wrt our portable implementations):
  ✓ Enc / Dec up to 20% / 25% faster
  ✓ Increased memory use: 4 tables of 1kB each
  ✓ Memory overhead decrease: store 1 table with slight loss of performance
Neon SIMD SW implementations

- Implementation for Neon SIMD on Arm Cortex-A9

- 128-bit instances (S-box in parallel in a single branch):
  - 30% less clock cycles
  - 0.5 kB reduction in ROM size
  - RAM size equal

- 64-bit instance (S-box in both branches in parallel):
  - 29% less clock cycles
  - ROM size approx. equal
  - RAM size increased
Low-area ForkSkinny HW architectures

ForkReg

Restart

Retrace

Andreeva, Deprez, Pittevils, Roy, Singh, Vizár

New Results and Insights on ForkAE
## Word-based architectures results

<table>
<thead>
<tr>
<th>ForkReg</th>
<th>Restart</th>
<th>Retrace</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Enc and Dec</td>
<td>• Encryption only</td>
<td>• Enc and Dec</td>
</tr>
<tr>
<td>• <strong>Best speedup</strong></td>
<td>• <strong>Best area</strong></td>
<td>• <strong>Goldilocks zone</strong></td>
</tr>
<tr>
<td>• 1.09-1.25 area of Skinny</td>
<td>• 0.97-1.11 area of Skinny</td>
<td>• 0.93-1.04 area of Skinny</td>
</tr>
<tr>
<td>• up to 129% throughput of Skinny</td>
<td>• up to 79% throughput of Skinny</td>
<td>• up to 126% throughput of Skinny</td>
</tr>
</tbody>
</table>

Results obtained w/ NanGate 45NM library, no clock gating or latches, datapath sizes of 1/16 block size.
More about implementations

- SW implementations
  - ✔ Check https://github.com/byt3bit/forkae
  - ✔ Updated results will be presented at CARDIS 2020
  - ✔ Implementations benchmarked at https://lwc.las3.de/

- HW implementations

- Questions to antoon.purnal@kuleuven.be

https://github.com/byt3bit/forkae/
More about implementations

- SW implementations
  - Check [https://github.com/byt3bit/forkae](https://github.com/byt3bit/forkae)
  - Updated results will be presented at CARDIS 2020
  - Implementations benchmarked at [https://lwc.las3.de/](https://lwc.las3.de/)

- HW implementations

- Questions to antoon.purnal@kuleuven.be

[https://github.com/byt3bit/forkae/](https://github.com/byt3bit/forkae/)
SAEF: Security

\[
\text{Adv}_{\text{priv}}^{\text{SAEF}}(A) \leq \text{Adv}_{F}^{\text{PRFP}}(D) + 2 \frac{(\sigma - q)^2}{2^n}
\]

\[
\text{Adv}_{\text{auth}}^{\text{SAEF}}(A) \leq \text{Adv}_{F}^{\text{PRFP}}(D) + 2 \frac{(\sigma - q + 1)^2}{2^n} + \frac{\sigma(\sigma - q)}{2^n} + \frac{q_v(q + 2)}{2^n}
\]

\[n/2\text{-bit nonce-based AE security}\]
$n/2$-bit OAE security \[\text{[ASV, SAC 2020]}\]

\[
\text{Adv}^{\text{oprpr}|\text{prf}}_{\text{SAEF}}(A) \leq \text{Adv}^{\text{PRFP}}_{F}(D) + \frac{3 \cdot \sigma^2}{2^{n+1}}
\]

\[
\text{Adv}^{\text{mr-auth}}_{\text{SAEF}}(A) \leq \text{Adv}^{\text{PRFP}}_{F}(D) + \frac{\sigma^2 + 4 \cdot q \nu}{2^n}
\]
Against attacker repeating (i.e., misusing) nonces:

Privacy
OPerm\([n]\) + random tag

Authenticity
Unforgeability
Against attacker repeating (i.e., misusing) nonces:

\[ M_1 C_1, M_2 C_2, M_3 C_3, M_4 C_4 + N, A \rightarrow E_K \]

**Privacy**

\[ \text{OPerm}[n] + \text{random tag} \]

**Authenticity**

Unforgeability

\[ \rightarrow \text{Leaks length of common } n\text{-aligned prefix of plaintexts if } N, A \text{ repeat} \]

\[ \rightarrow \text{Forging is as hard as with unique nonces} \]
SAEF: Implications

Efficient OAE

- e.g. 0.8 complexity of COLM-SKINNY

Safe for blockwise (adaptive) processing [EV, FSE 2017]

- Constrained environment (latency, limited memory, …)

Security under nonce misuse

- Integrity undamaged

- Well-defined privacy level
SAEF: Case studies

Nonce misuse in Lightweight applications

✓ Cheap HW platforms, forced resets, fault attacks etc
✓ Chosen Prefix, Secret Suffix attack on OAE (HTTPS) [HRRV 15]
✓ Possibly chosen prefix constant length $\Rightarrow$ CPSS shut down (MQTT)
✓ OAE-secure AE is a good, pragmatic solution
SAEF: Case studies

Nonce misuse in Lightweight applications

✓ Cheap HW platforms, forced resets, fault attacks etc
✓ Chosen Prefix, Secret Suffix attack on OAE (HTTPS) [HRRV 15]
✓ Possibly chosen prefix constant length ⇒ CPSS shut down (MQTT)
✓ OAE-secure AE is a good, pragmatic solution

Blockwise encryption

✓ Large data (temp. firmware image, graphics assets, maps etc) often on ext. flash
✓ Blockwise encryption typically unavoidable
✓ OAE-secure AE is safe to use
Extending the use case

ForkAE: an efficient candidate for **lightweight** applications, especially with predominantly short messages
Extending the use case

ForkAE: an efficient candidate for lightweight applications, especially with predominantly short messages

but also for defense in depth, offering the interesting combination of light weight and robustness.
Efficient encryption with Forkciphertext

- **Generalized counter mode (GCTR)**
  - ✓ random IV AND/OR nonce
  - ✓ tweakable forkciphertext
  - ✓ many ways to generate tweak/block input
  - ✓ direct use (encryption only)
  - ✓ as a component (such as in Deoxys II)

- **Systematic study of GCTR variants** [under submission]
  - ✓ high efficiency, up to BBB security
  - ✓ stay tuned!
Thank you!

damian.vizar@csem.ch