Can Lightweight Cryptography Deliver Heavyweight Security?

Orr Dunkelman

Computer Science Department
University of Haifa

28th March, 2016
Outline

1 Introduction
   - Lightweight Cryptography
   - Lightweight Cryptography Primitives

2 The Path to Security

3 A Few Examples
   - The AES to LED Transition
   - The KTANTAN Block Cipher
   - ZORRO
   - Squash

4 Conclusions/Discussions
Lightweight Cryptography

- Targets constrained environments.
- Tries to reduce the computational efforts needed to obtain security.
- Optimization targets: size, power, energy, time, code size, RAM/ROM consumption, etc.

Why now?
Lightweight Cryptography is All Around Us

- Constrained environments today are different than constrained environments 15 years ago.
- Ubiquitous computing – RFID tags, sensor networks, IoT, ...
- Low-end devices (8-bit platforms).
- Stream ciphers do not enjoy the same “foundations” as block ciphers.
- New modes of operation for lightweight block ciphers (mostly AE).
- Failure of previous solutions (KeeLoq, Mifare, ...) to meet required security targets.
- Good research direction...
Why Not AES?

- Considered to be **not lightweight** (disregard Intel’s platform).
- Some small related-key (related-subkey) issues.
- Key schedule not perfect.
- Not invented here syndrom.
## Some Lightweight Primitives

<table>
<thead>
<tr>
<th>Block Ciphers</th>
<th>Block Ciphers</th>
<th>Stream Ciphers</th>
<th>Hash Functions</th>
<th>MACs/AEs</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIGHT</td>
<td>AES*</td>
<td>Grain</td>
<td>H-PRESENT</td>
<td>SQUASH</td>
</tr>
<tr>
<td>mCrypton</td>
<td>Midori</td>
<td>Trivium</td>
<td>PHOTON</td>
<td>ACRON</td>
</tr>
<tr>
<td>DESL</td>
<td>Fantomas/Robin</td>
<td>Mickey</td>
<td>QUARK</td>
<td>ASC-1</td>
</tr>
<tr>
<td>PRESENT</td>
<td>PRIDE</td>
<td>F-FCSR-H</td>
<td>Armadillo</td>
<td>ASCON</td>
</tr>
<tr>
<td>KATAN</td>
<td>Rectangle</td>
<td>WG-7</td>
<td>SpongeNT</td>
<td>ALE</td>
</tr>
<tr>
<td>KTANTAN</td>
<td>Picollo</td>
<td>SNOW 3G</td>
<td>GLOUN</td>
<td>C-QUARK</td>
</tr>
<tr>
<td>PRINTcipherPRINCIPHER</td>
<td>ITUbee</td>
<td>Sprout Sprout</td>
<td>Keccak-f*</td>
<td>FIDES</td>
</tr>
<tr>
<td>SEA</td>
<td>Simeck</td>
<td></td>
<td>SipHash</td>
<td>Hummingbird</td>
</tr>
<tr>
<td>Klein</td>
<td>(X/XX)TEA</td>
<td></td>
<td></td>
<td>Joltik</td>
</tr>
<tr>
<td>LBlock</td>
<td>Chaskey</td>
<td></td>
<td></td>
<td>Ketje</td>
</tr>
<tr>
<td>GOST</td>
<td>GOST2</td>
<td></td>
<td></td>
<td>LACLAC</td>
</tr>
<tr>
<td>ZORRO</td>
<td>LEA</td>
<td></td>
<td></td>
<td>Sablier</td>
</tr>
<tr>
<td>TWINE</td>
<td>LED</td>
<td></td>
<td></td>
<td>(i)SCREAM</td>
</tr>
<tr>
<td>Simon</td>
<td>Speck</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRINCE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Partially based on [https://www.cryptolux.org/index.php/Lightweight_Cryptography](https://www.cryptolux.org/index.php/Lightweight_Cryptography)
Security Challenges

- Lightweight ⇒ pick the point on the security/performance curve with as little security margins as possible.
- Use best-of-the-art approaches:
  - Count the number of active S-boxes (wide trail),
  - Scale-down “known” ciphers (Misty1 → KASUMI, AES → LED, ZORRO, DES → DESL, …)
  - Use “secure structures” (GFNs/AES-like/etc.)
  - Ignore related-key attacks...
- Use provable approaches:
  - Even-Mansour (1-Key/Multiple Key)

- As usual ... pray.
The LED Block Cipher

- Introduced by [G+11].
- 64-bit block with 64-bit key (LED-64) or 128-bit key (LED-128).
- LED-64: 8-Step 1-Key Even-Mansour.
- LED-128: 12-Step 2-Key Even-Mansour.
- The “public permutation”: 4-round unkeyed AES-like construction.
The LED Block Cipher (cont.)

- No related-key issues/weakness in key schedule.
- As long as the 8-Step 1-Key Even-Mansour secure (LED-64) or 5-Step 1-Key Even-Mansour secure (LED-128).
## Results on LED (Single-Key)

<table>
<thead>
<tr>
<th>Source</th>
<th>Cipher</th>
<th>Steps</th>
<th>Time</th>
<th>Data</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>[IS12]</td>
<td>LED-64</td>
<td>2</td>
<td>$2^{56}$</td>
<td>$2^{8}$ CP</td>
<td>$2^{8}$</td>
</tr>
<tr>
<td>[D+14]</td>
<td>LED-64</td>
<td>2</td>
<td>$2^{48}$</td>
<td>$2^{16}$ CP</td>
<td>$2^{17}$</td>
</tr>
<tr>
<td>[D+14]</td>
<td>LED-64</td>
<td>2</td>
<td>$2^{48}$</td>
<td>$2^{48}$ KP</td>
<td>$2^{48}$</td>
</tr>
<tr>
<td>[D+13]</td>
<td>LED-64</td>
<td>3</td>
<td>$2^{60.2}$</td>
<td>$2^{49}$ KP</td>
<td>$2^{60}$</td>
</tr>
<tr>
<td>[IS12]</td>
<td>LED-128</td>
<td>4</td>
<td>$2^{112}$</td>
<td>$2^{16}$ CP</td>
<td>$2^{19}$</td>
</tr>
<tr>
<td>[M+12]</td>
<td>LED-128</td>
<td>4</td>
<td>$2^{96}$</td>
<td>$2^{64}$ KP</td>
<td>$2^{64}$</td>
</tr>
<tr>
<td>[NWW13]</td>
<td>LED-128</td>
<td>4</td>
<td>$2^{96}$</td>
<td>$2^{32}$ KP</td>
<td>$2^{32}$</td>
</tr>
<tr>
<td>[NWW13]</td>
<td>LED-128</td>
<td>6</td>
<td>$2^{124.4}$</td>
<td>$2^{59}$ KP</td>
<td>$2^{59}$</td>
</tr>
<tr>
<td>[D+13]</td>
<td>LED-128</td>
<td>6</td>
<td>$2^{124.5}$</td>
<td>$2^{45}$ KP</td>
<td>$2^{60}$</td>
</tr>
<tr>
<td>[D+13]</td>
<td>LED-128</td>
<td>8</td>
<td>$2^{123.8}$</td>
<td>$2^{49}$ KP</td>
<td>$2^{60}$</td>
</tr>
</tbody>
</table>
Related-Key Attacks on LED-64 \([M+12]\)

- Find iterative characteristic \(\Delta \rightarrow \Delta\) through \(P_i\).
- Set key difference to \(\Delta\), plaintext difference to 0 . . .
- 3-Step immediate related-key attack on LED-64, can be extended to 4-Step.
- 6-Step immediate related-key attack on LED-128.
The KTANTAN Block Ciphers [DDK09]

- KTANTAN has 3 flavors: KTANTAN-32, KTANTAN-48, KTANTAN-64.
- Block size: 32/48/64 bits.
- Key size: 80 bits.
- KATAN-\(n\) and KTANTAN-\(n\) are the same up to key schedule.
- In KTANTAN, the key is burnt into the device and cannot be changed.
General Structure of KATAN/KTANTAN

\[ \begin{array}{c}
L_2 \\
\downarrow \\
L_1 \\
\downarrow \\
IR \\
\downarrow \\
k_b \\
\downarrow \\
k_a
\end{array} \]

\[ \begin{array}{c}
\oplus \\
\downarrow \\
\ominus \\
\downarrow \\
\oplus \\
\downarrow \\
\ominus \\
\downarrow \\
\oplus
\end{array} \]
The KTANTAN Block Ciphers — Key Schedule

- Main problem — related-key and slide attacks.
- Solution A — two round functions, prevents slide attacks.
- Solution B — divide the key into 5 words of 16 bits, pick bits in a nonlinear manner.
- Specifically, let $K = w_4 || w_3 || w_2 || w_1 || w_0$, $T = T_7 \ldots T_0$ be the round-counter LFSR, set:

$$a_i = MUX16to1(w_i, T_7 T_6 T_5 T_4)$$

$$k_a = \overline{T_3} \cdot \overline{T_2} \cdot (a_0) \oplus (T_3 \lor T_2) \cdot \overline{T_3} \cdot T_2 \cdot (a_4)$$

$$\oplus (T_3 \lor \overline{T_2}) \cdot MUX4to1(a_3 a_2 a_1 a_0, \overline{T_1 T_0})$$

$$k_b = \overline{T_3} \cdot T_2 \cdot (a_4) \oplus (T_3 \lor \overline{T_2}) \cdot MUX4to1(a_3 a_2 a_1 a_0, \overline{T_1 T_0})$$
Security Analysis — Differential Cryptanalysis

- Computer-aided search for the various round combinations and all block sizes.
- **KATAN32**: Best 42-round characteristic has probability $2^{-11}$.
- **KATAN48**: Best 43-round characteristic has probability $2^{-18}$.
- **KATAN64**: Best 37-round characteristic has probability $2^{-20}$.
- This also proves that all the differential-based attacks fail (boomerang, rectangle).
Related-Key Differentials in KATAN

- No good methodology for that.
- In KATAN32 — each key bit difference must enter (at least) two linear operations and two non-linear ones.
- Hence, an active bit induces probability of $2^{-2}$, and cancels four other bits (or probability of $2^{-4}$ and 6).
- So if there are 76 key bits active — there are at least 16 quintuples, each with probability $2^{-2}$.
- The key expansion is linear, so check minimal hamming weight in the code.
Attacks on the KTANTAN Family

- [BR10] Meet in the middle attacks
  - Data: 2–3 KPs, Time: $\approx 2^{75}$, Memory: $O(1)$.

- [A11] Related-key attacks
  - Data: A few pairs of RK CPs (with 2–4 keys), Time: $2^{30}$, Memory: $O(1)$.

- [W+11] Meet in the middle attacks
  - Data: 4 CPs, Time: $\approx 2^{73}/2^{74}/2^{75}$, Memory: $O(1)$.
What Went Wrong?

- The key schedule.
- The bits which are chosen as the key are not “well distributed”.
- For example, bit 32 of the key, does not enter the first 218 rounds...
- Other bits which are not that common also appear.
- This can be used in several ways (MitM, RK differentials).
ZORRO Block Cipher \([G+13]\)

- Lightweight block cipher that targets side channel security.
- 128-bit block, 128-bit key.
- Single-key iterated Even-Mansour construction.
- 24 rounds, every four rounds the key is XORed to the state.
- Based on the AES.
The ZORRO Block Cipher (cont.)

\[ \oplus^4 R \oplus^4 R \oplus^4 R \oplus^4 R \oplus^4 R \oplus^4 R \]

[Diagram of ZORRO Block Cipher with 6 rounds, each round labeled with \( K \).]
The ZORRO Round Function

**SubBytes**

**ShiftRows**

**MixColumns**

0 4 8 (12)
1 5 9 13
2 6 10 14
3 7 11 15

3 7 11 15
15 3 7 11

0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0

AC

RCᵢ

ShiftRows

MixColumns
Interesting Properties of ZORRO \([W+13]\)

- S-boxes are used only in the first row.
- Circulant matrices have interesting properties when raised to the power. Namely,

\[
\begin{pmatrix}
2 & 3 & 1 & 1 \\
1 & 2 & 3 & 1 \\
1 & 1 & 2 & 3 \\
3 & 1 & 1 & 2 \\
\end{pmatrix}^4 = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{pmatrix}
\]

- So What?
Differential/Linear Properties of ZORRO [W+13]

- Consider differences/masks of the form:

\[
\begin{pmatrix}
  a & a & a & a \\
  b & b & b & b \\
  c & c & c & c \\
  d & d & d & d \\
\end{pmatrix}
\]

- The equality of different columns remains, up to the S-boxes.
- Which are applied only to the first row.
- So let’s try to not activate it...
Differential/Linear Properties of ZORRO (cont.)

\[
\begin{pmatrix}
0 \\
a \\
b
\end{pmatrix}
\xrightarrow{SB}
\begin{pmatrix}
0 \\
a \\
b
\end{pmatrix}
\xrightarrow{MC}
\begin{pmatrix}
0 \\
c \\
d \\
e
\end{pmatrix}
\xrightarrow{SB}
\begin{pmatrix}
0 \\
c \\
d \\
e
\end{pmatrix}
\xrightarrow{MC}
\begin{pmatrix}
0 \\
f \\
0 \\
g
\end{pmatrix}
\xrightarrow{SB}
\begin{pmatrix}
0 \\
f \\
0 \\
g
\end{pmatrix}
\xrightarrow{MC}
\begin{pmatrix}
0 \\
h \\
i \\
j \\
k
\end{pmatrix}
\xrightarrow{SB}
\begin{pmatrix}
0 \\
i \\
j \\
k
\end{pmatrix}
\xrightarrow{MC}
\begin{pmatrix}
0 \\
a \\
0 \\
b
\end{pmatrix}
\xrightarrow{AK}
\begin{pmatrix}
0 \\
a \\
0 \\
b
\end{pmatrix}
\]
Using the iterative characteristic it is possible to devise:

- Differential attack (20-round characteristic, $2^{-108.3}$ probability, 4-R attack, $2^{112.4}$ CPs, $2^{112.4}$ time).
- Linear distinguisher (24-round characteristic, $2^{-52.62}$ bias, 0-R attack, $2^{105.3}$ KPs).
Our Improvements [B+15]

- One can offer better linearization as most operations are linear.
- So try to right linear equations, and represent each active S-box as introducing new variables.
- Use linear algebra to solve.
- The result is a 4-round iterative characteristic with 2 active S-boxes.
- Which means probability of $2^{-54.2}$ and bias of $2^{-26.8}$.
- Also, user better distinguisher $\Rightarrow$ key recovery transformation.
- 20-round linear characteristic
- 4-round attack
- Immediate attack — $2^{45}$ KPs or $2^{41.5}$ CPs (and $2^{45}$ time).
What Went Wrong?

- Too few active S-boxes.
- Circulant matrices, which are good for implementation, may have undesirable security properties.
- Adding the key a few times — may cause some security problems.
Squash [S07]

- Lightweight MAC for lightweight authentication.
- Given a secret $S$, and challenge $R$ produce $h(S, R)$ as answer.
- Main idea, compute $\text{Trunc}(f^2(R, S) \mod N)$.
- $f$ not trivial, but not too nonlinear.
- $N$ to be a composite Mersenne number ($2^n - 1$) to offer easy implementation.
Squash-0 Implementation Tricks

- Set $X = R \oplus S$. Easy.
- Expand $X$ into a modulo $n$ number using LFSR:
  - Reduces storage overhead (only $|X|$ bits stored),
  - Easy to implement,
  - We understand secure/light LFSRs,
- Doubling modulo $2^n - 1$:
  - Easy as $2^n = 1!$
  - Compute $y^2 = y_1 \cdot 2^n + y_2 \Rightarrow y^2 = y_1 + y_2 \mod n$.
  - Oh, and if $y_1 + y_2 > n$, then $y_1 + y_2 - n \approx y_1 + y_2$ for most bits.
  - Suggests security is related to factoring (Rabin)
Squash-0 Implementation Tricks (cont.)

- Truncation to $t$ middle bits:
  - Reduces computational overhead,
  - Reduces storage overhead,
  - Improves security (factoring may not cause failure),

- Approximate Computation:
  - Compute $u$ before the $t$ bits needed for sending,
  - Gives the correct answer with probability $1 - 2^{-u}$,
  - Both sides know $u$, so they err together $\Rightarrow$ they obtain same answer,

- Computing $m_1 + m_2$ can be done using convolution:
  - Bit $j + k$ is the sum of all $m_v \cdot m_{j+k-v}$ (plus carry),
“Smashing Squash” [OV09]

- A series of attacks against Squash-0.
- Exploit the Rabin’s underlying structure (and some cool algebra tricks).
- Running time $O(|S|^2 \log |S|)$ and $O(|S|^2/n)$ chosen challenges.
What Went Wrong?

- Too strong of a structure.
- LFSRs + Mersenne numbers may not be sufficient.
- And using a nonlinear combining function $f$ — well, the entire point was to avoid it.
Conclusions/Discussions

- How much are we willing to pay for security in lightweight schemes?
- What is the target for lightweight schemes optimization?
- Scale-down or “innovate”? 
- Related-key attacks? Weak key schedules? How? Why? What?
- Side channel Security?
Optimization Targets

- Area?
- Power?
- Throughput?
- Energy?
- Latency?
- ...
“Approved” Tricks?

- Burning the key into device?
- Printing the key into the device?
- Increasing internal state by constants?
- Claiming reduced security (128-bit key, 80-bit security)?
- Reduced plaintext size (⇒ reduced internal state)?
- Hiding the “nonlinearity” source?
- Exernalizing I/O cost? (serial access may affect performance)
- . . .
Questions?

Thank you for your attention!