On Chaskey

Work in progress...

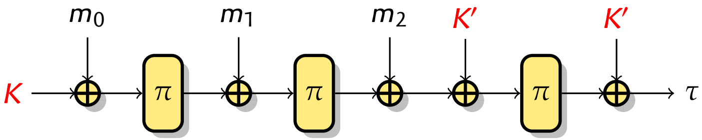
Gaëtan Leurent

Inria

ESC 2015

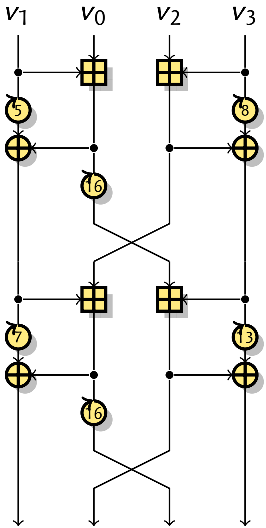
Chaskey

- Fast lightweight MAC, without nonce
- CBC-MAC with an Even-Mansour cipher
- Birthday security
  - 128-bit key
  - 128-bit state
  - Security claim: 2^{48} data, 2^{80} time.



- Sponge based, no permutation inverse

Chaskey permutation



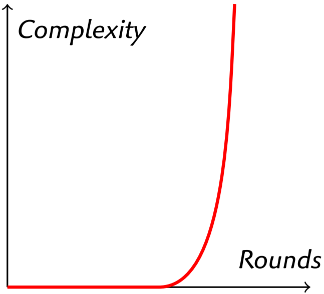
Mini Siphash

- ARX
- 32-bit words
- 128-bit state
- 8 rounds

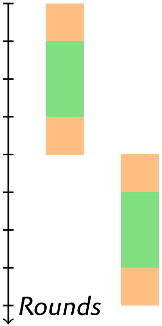
Cryptanalysis of ARX schemes

- No iterative differential/linear trails
- Small difference in the middle and propagate

- Only short trails with high probability

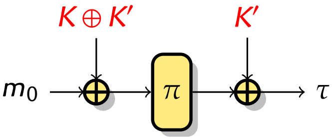


- Can we combine two trails?



Cryptanalysis of Chaskey

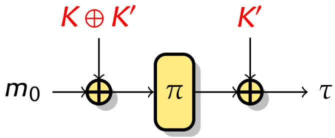
- Use single-block messages
  - Chaskey becomes an Even-Mansour cipher



- No decryption oracle
  - Boomerang not possible
  - Differential-Linear cryptanalysis does not require π^{-1}

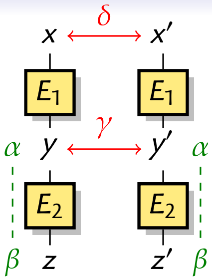
Cryptanalysis of Chaskey

- Use single-block messages
  - Chaskey becomes an Even-Mansour cipher



- No decryption oracle
  - Boomerang not possible
  - Differential-Linear cryptanalysis does not require π^{-1}

Differential-Linear Cryptanalysis



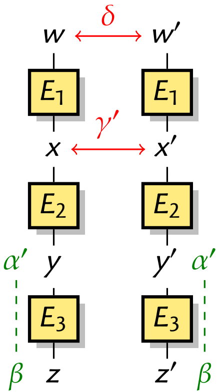
- Divide E in two sub-ciphers $E = E_2 \circ E_1$
  - Let $y = E_1(x)$, $z = E_2(y)$
- Find a differential $\delta \to \gamma$ for E_1
  - $\Pr\left[E_1(x \oplus \delta) = E_1(x) \oplus \gamma\right] = p$
- Find a linear approximation $\alpha \to \beta$ of E_2
  - $\Pr\left[\alpha \bullet y = \beta \bullet E_2(y)\right] = \frac{1}{2}(1+\varepsilon)$
- Query a pair $(x, x' = x \oplus \delta)$:

$$
\begin{aligned}
y \oplus y' &= \gamma && \text{proba } p && (1)\\
\alpha \bullet (y \oplus y') &= \alpha \bullet \gamma && \text{proba} \approx p + 1/2(1-p) = 1/2(1+p) && (2)\\
\beta \bullet z &= \alpha \bullet y && \text{proba } 1/2(1+\varepsilon) && (3)\\
\beta \bullet z' &= \alpha \bullet y' && \text{proba } 1/2(1+\varepsilon) && (4)\\
\beta \bullet (z \oplus z') &= \alpha \bullet \gamma && \text{proba } 1/2(1+p\varepsilon^2) && (5)
\end{aligned}
$$

- Distinguisher with complexity $\approx p^{-2}\varepsilon^{-4}$

Application to Chaskey

- Accurate analysis of differential-linear attack is hard [BLN, FSE'14]
  - Proba for wrong pair is not 1/2
  - Many differential trails with same δ
  - Many linear trails with same β
- Evaluate middle rounds experimentally
  - Shorter trails $\delta \to \gamma'$, $\alpha' \to \beta$
    - Single bit difference γ'
    - Single bit mask α'
  - Eval $\Pr\left[\alpha' \bullet (E_2(x) \oplus E_2(x \oplus \gamma')) = 1\right]$
    - Biased output bit, with 1-bit input difference
    - Select the best single bit γ', α'



A 6-round distinguisher

- E_1: 1 round, $p = 2^{-5}$
  - $v_0[26], v_1[26], v_2[6,23,30], v_3[23,30] \rightarrow v_2[22]$
- E_2: 4 rounds, $b \approx 2^{-6.05}$
  - $v_2[22] \rightarrow v_2[16]$
- E_3: 1 round, $\varepsilon \approx 2^{-2.6}$
  - $v_2[16] \rightarrow v_0[5], v_1[23,31], v_2[0,8,15], v_3[5]$

- Differential-linear bias: $p \cdot b \cdot \varepsilon^2 \approx 2^{-16.25}$
- Distinguisher with complexity $c/p^2b^2\varepsilon^4 \approx c \cdot 2^{32.5}$

Improved attack

1. We guess some key-bits in order to increase the probability of the linear and differential trails.
2. Partition the data, and keep subsets with higher bias
3. Multiple differentials and structures

- Techniques inspired by:
  - Improved linear cryptanalysis of addition [Biham & Carmeli, SAC '14]
  - Salsa20 Probabilistic Neutral Bits [AFKMR, FSE '08]

Improved linear

First non-linear operation

$x = (a \oplus k_a) \boxplus (b \oplus k_b)$ $\tilde{a} = a \oplus k_a,\ \tilde{b} = b \oplus k_b$

- Goal: predict bit $x[k]$ for inputs (a, b)
- Classic linear: $x[k] \approx a[k] \oplus b[k] \oplus b[k-1]$
  - $Pr\Big[x[k] = a[k] \oplus b[k] \oplus b[k-1]\Big] = 3/4$
- Guessing key bits gives bits of $\tilde{a}$ and $\tilde{b}$

Improved linear

First non-linear operation

$x = (a \oplus k_a) \boxplus (b \oplus k_b)$ $\tilde{a} = a \oplus k_a, \tilde{b} = b \oplus k_b$

- If $(\tilde{a}_{k-1}, \tilde{b}_{k-1}) = (0, 0)$
  there is no carry

$$\begin{array}{cccccc} & & & 0 & & \\ & ? & a & 0 & ? & ? \\ + & ? & b & 0 & ? & ? \\ \hline & ? & x & ? & ? & ? \end{array}$$

- Therefore $x_k = \tilde{a}_k \oplus \tilde{b}_k$

- If $(\tilde{a}_{k-1}, \tilde{b}_{k-1}) = (1, 1)$
  there is always a carry

$$\begin{array}{cccccc} & & & 1 & & \\ & ? & a & 1 & ? & ? \\ + & ? & b & 1 & ? & ? \\ \hline & ? & x & ? & ? & ? \end{array}$$

- Therefore $x_k = \tilde{a}_k \oplus \tilde{b}_k \oplus 1$

- We throw out one half of the data
- But the distinguisher requires 4 times less data

Improved linear

First non-linear operation

$x = (a \oplus k_a) \boxplus (b \oplus k_b)$ $\qquad \tilde{a} = a \oplus k_a,\ \tilde{b} = b \oplus k_b$

- If $(\tilde{a}_{k-1}, \tilde{b}_{k-1}) = (0,0)$ there is no carry

$$\begin{array}{cccccc} & & & 0 & 0 & \\ & ? & a & 0 & 0 & ? \\ + & ? & b & 1 & 0 & ? \\ \hline & ? & x & ? & ? & ? \end{array}$$

- Therefore $x_k = \tilde{a}_k \oplus \tilde{b}_k$

- If $(\tilde{a}_{k-1}, \tilde{b}_{k-1}) = (1,1)$ there is always a carry

$$\begin{array}{cccccc} & & & 1 & 1 & \\ & ? & a & 0 & 1 & ? \\ + & ? & b & 1 & 1 & ? \\ \hline & ? & x & ? & ? & ? \end{array}$$

- Therefore $x_k = \tilde{a}_k \oplus \tilde{b}_k \oplus 1$

- We throw out one fourth of the data
- But the distinguisher requires 4 times less data

Improved linear

First non-linear operation

$x = (a \oplus k_a) \boxplus (b \oplus k_b)$ $\qquad$ $\tilde{a} = a \oplus k_a$, $\tilde{b} = b \oplus k_b$

- If $(\tilde{a}_{k-1}, \tilde{b}_{k-1}) = (0, 0)$ there is no carry

$$
\begin{array}{cccccc}
 & & & 0 & 0 & \\
 & ? & a & 0 & 0 & ? \\
+ & ? & b & 1 & 0 & ? \\
\hline
 & ? & x & ? & ? & ?
\end{array}
$$

$\tilde{a}_{k-1}$	$\tilde{a}_{k-2}$	0 0	0 1	1 0	1 1	← $\tilde{b}_{k-1}$ $\tilde{b}_{k-2}$
0	0	+	+	+	?	
0	1	+	+	?	−	
1	0	+	?	−	−	
1	1	?	−	−	−	

- Therefore $x_k = \tilde{a}_k \oplus \tilde{b}_k$
- We throw out one fourth of the data
- But the distinguisher requires 4 times less data

Improved linear

- We can also predict some input bits of the next additions
- But it gets messy...

Experimental approach

- Identify candidate bits (by hand)
- Collect data:
  - Filter according to candidate bits
  - Measure bias
- Build vector of bias, and look for symmetries
  - Symmetries allow the reduce the number of filtering bits

Improved differential

First non-linear operation

$x = (a \oplus k_a) \boxplus (b \oplus k_b), x' = (a' \oplus k_a) \boxplus (b' \oplus k_b)$ $\tilde{a} = a \oplus k_a, \tilde{b} = b \oplus k_b$

- Goal: generate pairs (a, b) with $x \oplus x' = 2^k$
- Classic differential: $a \oplus a' = 2^k, b = b'$
  - $Pr\left[x \oplus x' = 2^k\right] = 1/2$
- Guessing key bits gives bits of $\tilde{a}$ and $\tilde{b}$

Improved differential

First non-linear operation

$x = (a \oplus k_a) \boxplus (b \oplus k_b),\ x' = (a' \oplus k_a) \boxplus (b' \oplus k_b)$ $\qquad \tilde{a} = a \oplus k_a,\ \tilde{b} = b \oplus k_b$

- If $\tilde{b}_{k-1} = 0$, no carry

$$\begin{array}{cccccc} & & 0 & & & \\ & - & - & x & - & - \\ + & - & - & 0 & - & - \\ \hline & - & - & x & - & - \end{array}$$

- If $\tilde{b}_{k-1} = 1$, carry

$$\begin{array}{cccccc} & & ? & x & & \\ & - & - & x & - & - \\ + & - & - & 1 & - & - \\ \hline & ? & x & x & - & - \end{array}$$

- We throw out one half of the data
- But the distinguisher requires 4 times less data

Improved differential

First non-linear operation

$x = (a \oplus k_a) \boxplus (b \oplus k_b),\ x' = (a' \oplus k_a) \boxplus (b' \oplus k_b)$ $\qquad \tilde{a} = a \oplus k_a,\ \tilde{b} = b \oplus k_b$

- If $\tilde{b}_{k-1} = 0$, no carry
- If $\tilde{b}_{k-1} = 1$, carry
- Use multiple differentials: multiple bits input difference
  - Encrypt structure of plaintexts, build pairs depending on key guess
- If different signs, no carry

```
        n
    -   u   n   -   -
+   -   -   1   -   -
---------------------
    -   -   x   -   -
```

- If same signs, carry

```
        u
    -   u   u   -   -
+   -   -   1   -   -
---------------------
    ?   x   x   -   -
```

- We throw out one fourth of the data
- But the distinguisher requires 4 times less data

Improved differential

- We can also predict some input bits of the next additions
- But it gets messy...

Experimental approach

- Identify candidate bits (by hand)
- Collect data:
  - Filter according to candidate bits
  - Measure probability
- Build vector of probabilities, and look for symmetries
  - Symmetries allow the reduce the number of filtering bits

Remark

Need more key bit guesses to improve differential than to improve linear

Improved 6-round attack

- To summarize:
  half round at the top and bottom almost for free
- Improved 6 round attack
  - Implemented
  - Algorithmic tricks to reduce the time complexity (using counters)
  - Data complexity: 2^{25} (v. 2^{35})
  - Time complexity: 2^{29} (elementary operations)
  - Recovers 13 key bits with high probability
- TODO: full key recovery

A 7-round attack?

- The attack can be extended to 7 rounds
- E_1: 1.5 round
  - $p = 2^{-17}$
- E_2: 4 round
  - Best bias: $v_0[31] \to v_2[20]$, $b \approx 2^{-6.1}$
- E_3: 1.5 round
  - $\varepsilon \approx 2^{-7.6}$
- Simple distinguisher: bias $\approx 2^{38.3}$
- Work in progress to identify good bits
- Expected data complexity $\approx 2^{45} - 2^{48}$

Conclusion

- Differential-Linear attack quite efficient for ARX designs
- Security margin of Chaskey rather slim (7/8 rounds broken)

Comparison with SipHash

- Same round function with 64-bit words
- Fewer rounds
- Inputs smaller blocks in the state
  - Input differential can not affect the full state
  - DL can analyze fewer rounds backward
- 4 output words are xored together
  - Smaller bias with forward rounds