Key-Recovery Attacks Against the MAC Algorithm Chaskey

Chrysanthi Mavromati

Capgemini-Sogeti R&D Lab and UVSQ Laboratoire PRiSM, France

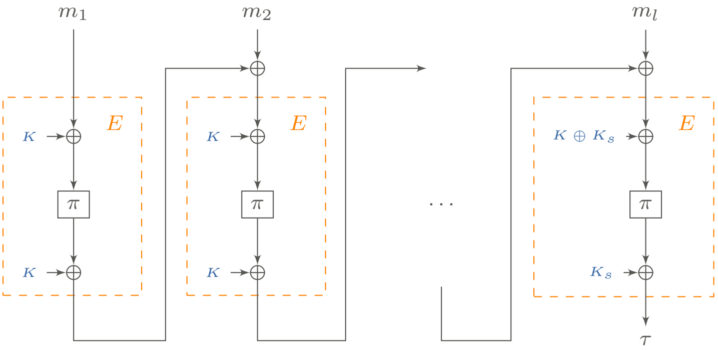
SAC 2015

The MAC Algorithm Chaskey

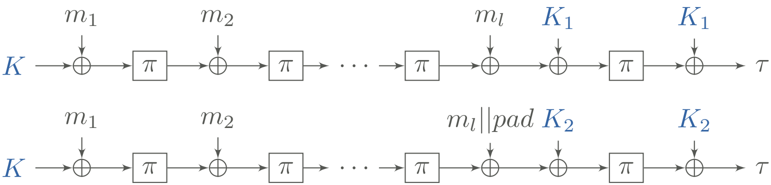
Permutation-based MAC algorithm introduced by Mouha et al. in 2014.

- Underlying permutation based on ARX design.
- Similar to CBC-MAC construction with an Even-Mansour block cipher.
- A 128-bit key K generates two subkeys K_s with $s \in \{1, 2\}$:

$$K_1 = \alpha K \text{ and } K_2 = \alpha^2 K.$$



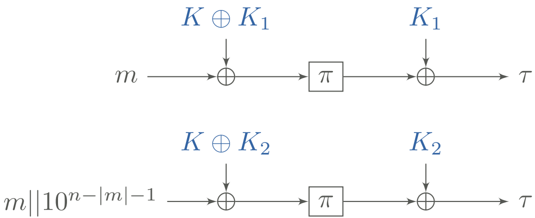
The MAC Algorithm Chaskey



- With data D, it is proven secure up to $T = 2^{128}/D$ evaluations of π.
- Best data/time tradeoff: $D = T = 2^{64}$.

The MAC Algorithm Chaskey

Becomes an Even-Mansour cipher when single-block messages are used.



Results on Even-Mansour Scheme

[Daemen], [Biryukov et al.], [Dunkelman et al.]

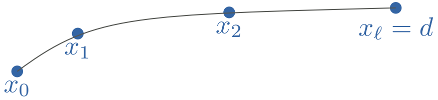
Many known attacks against the EM scheme which confirm the main security proof: $DT = 2^n$.

[Fouque et al.] at Asiacrypt 14

A new collision-based attack using the distinguished point technique against the EM scheme in the multi-user setting.

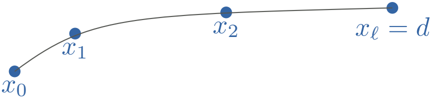
Finding Collisions: the Distinguished Point Technique

- Define a function f on a set S of size N.
- Define a distinguished subset S_0 of S.
- Build chains from random startpoints: $x_{i+1} = f(x_i)$.
- Stop chain when $x_\ell = d \in S_0$.
- Store (x_0, d, ℓ).

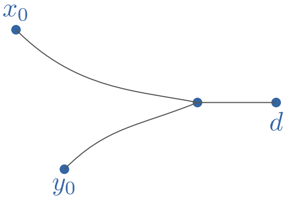


Finding Collisions: the Distinguished Point Technique

- Define a function f on a set S of size N.
- Define a distinguished subset S_0 of S.
- Build chains from random startpoints: $x_{i+1} = f(x_i)$.
- Stop chain when $x_\ell = d \in S_0$.
- Store (x_0, d, ℓ).

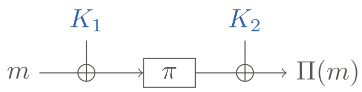


How do we detect a collision?



The Attack of Fouque et al.

The Even-Mansour Scheme:



Collision-based attack using the distinguished point technique.

1. Build chains by using functions:

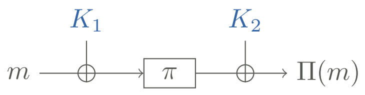
$$F(P) = P \oplus \Pi(P) \oplus \Pi(P \oplus \delta)$$

$$f(P) = P \oplus \pi(P) \oplus \pi(P \oplus \delta)$$

For two plaintexts $(P, P^{'})$ where: $P^{'} = P \oplus K_1$ or $P^{'} = P \oplus K_1 \oplus \delta$
$\Rightarrow F(P^{'}) = f(P) \oplus K_1$(resp. $\oplus \delta$)

The Attack of Fouque et al.

The Even-Mansour Scheme:



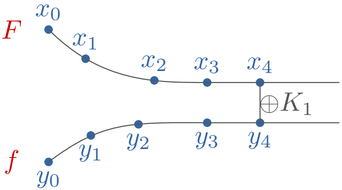
Collision-based attack using the distinguished point technique.

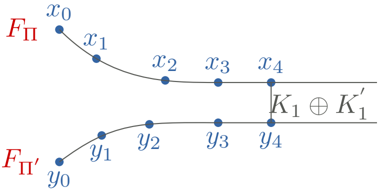
1. Build chains by using functions:

$$F(P) = P \oplus \Pi(P) \oplus \Pi(P \oplus \delta)$$

$$f(P) = P \oplus \pi(P) \oplus \pi(P \oplus \delta)$$

For two plaintexts (P, P') where: $P' = P \oplus K_1$ or $P' = P \oplus K_1 \oplus \delta$
$\Rightarrow F(P') = f(P) \oplus K_1$(resp. $\oplus \delta$)





The Attack of Fouque et al.

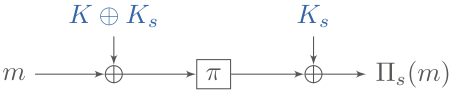
2. Construct a graph:
   - Nodes are labelled by the users and the unkeyed user.
   - If a collision is found, then add a vertex between the two nodes.
   - If a single collision between a user and the unkeyed user is found, then we learn all keys in the connected component.

Analysis of the attack:

For $2^{n/3}$ users, $2^{n/3}$ queries/user, $2^{n/3}$ unkeyed queries
$\rightarrow$ recover a constant fraction of $2^{n/3}$ keys

Key-Recovery Attacks Against Chaskey

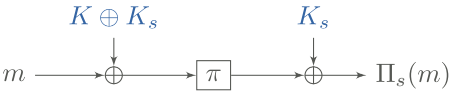
Use single-block messages:



Build chains by using a function based on Chaskey and then search for collisions between them.

Key-Recovery Attacks Against Chaskey

Use single-block messages:



Build chains by using a function based on Chaskey and then search for collisions between them.

Define functions:

$$\Pi_s(M) = K_s \oplus \pi(M \oplus (K_s \oplus K))$$

$$F_{\Pi_s}(M) = \Pi_s(M) \oplus \Pi_s(M \oplus \delta) \oplus M$$

$$f_\pi(M) = M \oplus \pi(M) \oplus \pi(M \oplus \delta)$$

Attack in the Single-User Setting

For a complete block M and an incomplete $M^{'}$:

1. Create chains constructed by using both:

$$F_{\Pi_1}(M) = \Pi_1(M) \oplus \Pi_1(M \oplus \delta) \oplus M$$

$$F_{\Pi_2}(M^{'}) = \Pi_2(M^{'}) \oplus \Pi_2(M^{'} \oplus \delta) \oplus M^{'}$$

2. Store all endpoints and search for collisions between the two different types of chains:

$$F_{\Pi_1}(M) = F_{\Pi_2}(M^{'})$$

3. If a collision is found, recover:

$$\boxed{K_1 \oplus K \oplus K_2 \oplus K = (\alpha + \alpha^2)K}$$

Attack in the Single-User Setting

For a complete block M and an incomplete $M^{'}$:

1. Create chains constructed by using both:

$$F_{\Pi_1}(M) = \Pi_1(M) \oplus \Pi_1(M \oplus \delta) \oplus M$$

$$F_{\Pi_2}(M^{'}) = \Pi_2(M^{'}) \oplus \Pi_2(M^{'} \oplus \delta) \oplus M^{'}$$

2. Store all endpoints and search for collisions between the two different types of chains:

$$F_{\Pi_1}(M) = F_{\Pi_2}(M^{'})$$

3. If a collision is found, recover:

$$\boxed{K_1 \oplus K \oplus K_2 \oplus K = (\alpha + \alpha^2)K}$$

$\Rightarrow$ Collision between two chains of length 2^{64}.

The Multi-User Setting

L different users are all using Chaskey based on the same public permutation π.

- Each U_i chooses $K^{(i)}$ and generates $K_s^{(i)}$ with $s \in \{1, 2\}$.
- **Define functions:**

$$\Pi_s^{(i)}(M) = K_s^{(i)} \oplus \pi(M \oplus (K_s^{(i)} \oplus K^{(i)}))$$

$$F_{\Pi_s^{(i)}}(M) = \Pi_s^{(i)}(M) \oplus \Pi_s^{(i)}(M \oplus \delta) \oplus M$$

$$F_\pi(M) = M \oplus \pi(M) \oplus \pi(M \oplus \delta)$$

Attacks in the Multi-User Setting: Simple Application

1. Build chains for every user and the unkeyed user.
2. Store the endpoints and search for collisions between chains:
   - Collision between two chains $F_{\Pi_1}(M)$ and $F_{\Pi_2}(M^{'})$ of user i:

$$K_1^{(i)} \oplus K^{(i)} \oplus K_2^{(i)} \oplus K^{(i)} = (\alpha + \alpha^2)K^{(i)}$$

   - Collision between similar chains of different users i and j:

$$K_1^{(i)} \oplus K^{(i)} \oplus K_1^{(j)} \oplus K^{(j)} = (1 + \alpha)(K^{(i)} \oplus K^{(j)})$$

$$K_2^{(i)} \oplus K^{(i)} \oplus K_2^{(j)} \oplus K^{(j)} = (1 + \alpha^2)(K^{(i)} \oplus K^{(j)})$$

   - Collision between different chains of different users i and j (cross collision):

$$K_1^{(i)} \oplus K^{(i)} \oplus K_2^{(j)} \oplus K^{(j)} = (1 + \alpha)K^{(i)} \oplus (1 + \alpha^2)K^{(j)}$$

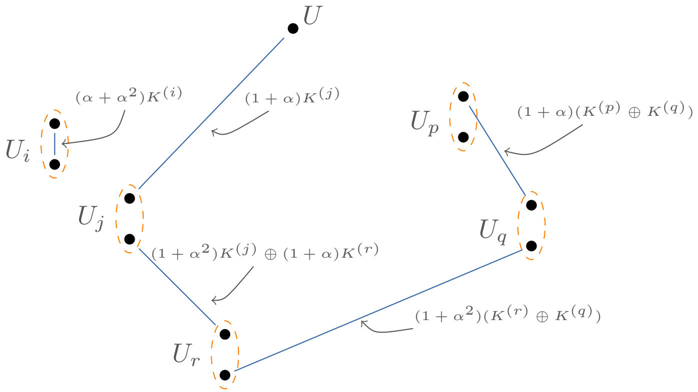
   - Collision between a chain of a keyed user i and the unkeyed user:

$$K_1^{(i)} \oplus K^{(i)} = (1 + \alpha)K^{(i)}$$

$$K_2^{(i)} \oplus K^{(i)} = (1 + \alpha^2)K^{(j)}$$

Attacks in the Multi-User Setting: Simple Application

3. Build a graph:
   - Each user is represented by two nodes.
   - If a collision is found, put a vertex between the corresponding nodes.



4. Solve the system and recover the users' keys.

Attacks in the Multi-User Setting: Simple Application

Analysis of the attack

For 2^{43} users, 2^{43} queries/user, 2^{43} unkeyed queries
$\Rightarrow$ recover almost all keys

Variant of Previous Attack: Cross Collisions

1. For two users U_i and U_j, build chains using functions F_{Π_1} and F_{Π_2}.

2. For each chain, store the endpoints and search for a cross collision:

$$K_1^{(i)} \oplus K^{(i)} \oplus K_2^{(j)} \oplus K^{(j)} = (1+\alpha)K^{(i)} \oplus (1+\alpha^2)K^{(j)}.$$

3. If a cross collision is detected, recover from the corresponding outputs:

$$K_1^{(i)} \oplus K_2^{(j)} = \alpha K^{(i)} \oplus \alpha^2 K^{(j)}.$$

4. Solve the system:

$$\begin{aligned} (1+\alpha)K^{(i)} \oplus (1+\alpha^2)K^{(j)} &= \Delta_1 \\ \alpha K^{(i)} \oplus \alpha^2 K^{(j)} &= \Delta_2 \end{aligned}$$

and recover $K^{(i)}$ and $K^{(j)}$.

Variant of Previous Attack: Cross Collisions

Analysis of the attack

- Similar techniques as basic attack.
- Recover keys of two users with only one cross collision between them.

For 2^{32} users, 2^{32} queries per user
$\Rightarrow$ recover keys of two users

Conclusion

New key-recovery attacks on Chaskey in the single and multi-user setting:

1. Single-user setting: recover key with complexity 2^{64} (respects claimed security).
2. Multi-user setting: recover almost all keys in a group of 2^{43} users.
3. Multi-user setting: recover 2 keys in a group of 2^{32} users.

Conclusion

New key-recovery attacks on Chaskey in the single and multi-user setting:

1. Single-user setting: recover key with complexity 2^{64} (respects claimed security).
2. Multi-user setting: recover almost all keys in a group of 2^{43} users.
3. Multi-user setting: recover 2 keys in a group of 2^{32} users.

Thank you for your attention!