Practical Key Recovery Attack against Secret-IV EDON-$\mathcal{R}$

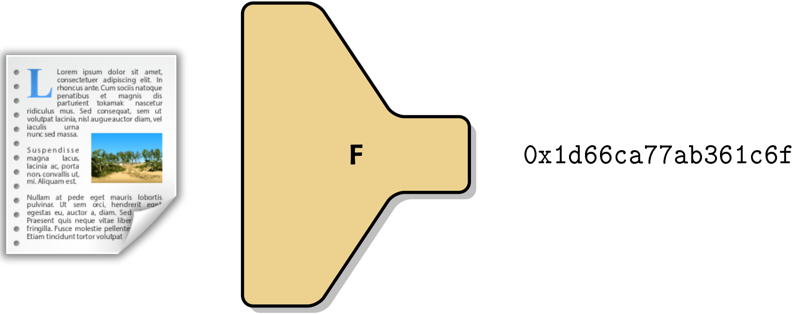
Gaëtan Leurent

École Normale Supérieure
Paris, France

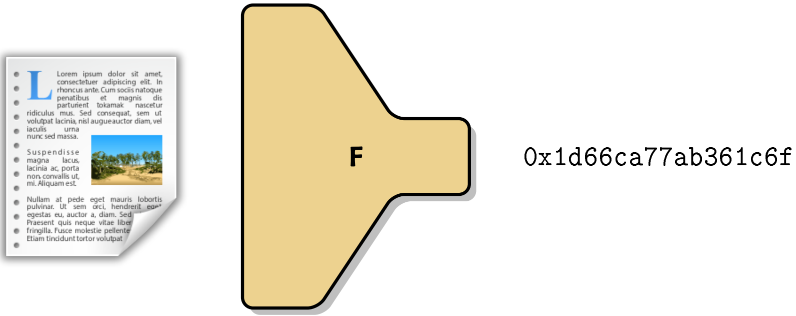
What is a hash function?

- A public function with no structural properties.
  - Cryptographic strength without keys!
  - We also expect security when used in a keyed mode...

- $F : \{0,1\}^* \to \{0,1\}^n$



- A public function with no structural properties.
  - Cryptographic strength without keys!
  - We also expect security when used in a keyed mode...
- $F : \{0,1\}^* \rightarrow \{0,1\}^n$



The SHA-3 Competition

- Similar to the AES competition
- Organized by NIST
- After devastating attacks on MD4, MD5, SHA-1, ...

- Submission dead-line was October 2008: 64 candidates
- 51 valid submissions in first round (December 2008)

- 14 in the second round (July 2009)
- 5 finalists in September 2010?
- 1 winner in 2012?

New designs

- Take into consideration recent advances in cryptanalysis
- Somewhat higher expectation that SHA-2
- Wide diversity of designs
- EDON-$\mathcal{R}$ was one of the first round candidates
  - One of the fastest.
- In this work, we study EDON-$\mathcal{R}$ used as a MAC

New designs

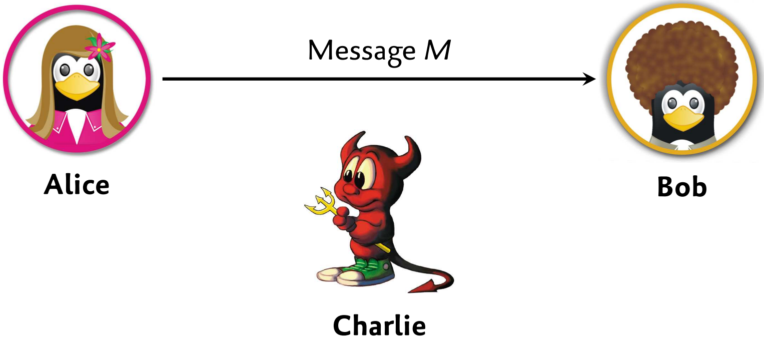
- Take into consideration recent advances in cryptanalysis
- Somewhat higher expectation that SHA-2
- Wide diversity of designs
- EDON-$\mathcal{R}$ was one of the first round candidates
  - One of the fastest.
- In this work, we study EDON-$\mathcal{R}$ used as a MAC

Danilo Gligoroski, Rune Steinsmo Ødegård, Marija Mihova, Svein Johan Knapskog, Ljupco Kocarev, Aleš Drápal, Vlastimil Klima
Cryptographic Hash Function EDON-$\mathcal{R}$
Submission to the NIST SHA-3 competition

New designs

- Take into consideration recent advances in cryptanalysis
- Somewhat higher expectation that SHA-2
- Wide diversity of designs
- EDON-$\mathcal{R}$ was one of the first round candidates
    - One of the fastest.
- In this work, we study EDON-$\mathcal{R}$ used as a MAC

Danilo Gligoroski, Rune Steinsmo Ødegård, Marija Mihova, Svein Johan Knapskog, Ljupco Kocarev, Aleš Drápal, Vlastimil Klima
Cryptographic Hash Function EDON-$\mathcal{R}$
Submission to the NIST SHA-3 competition



- Alice wants to send a message to Bob
- They share a key, and use a MAC to authenticate the message
- Bob rejects the message if $\text{MAC}_k(M) \neq t$

What is a MAC algorithm?

- Alice wants to send a message to Bob
- They share a key, and use a MAC to authenticate the message
- Bob rejects the message if $\mathrm{MAC}_k(M) \neq t$

MAC security

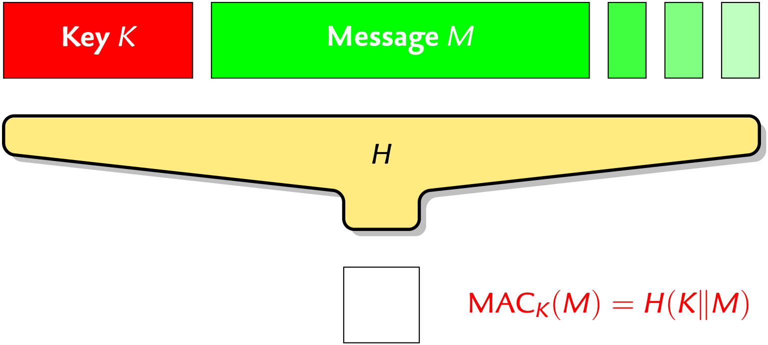
A MAC (Message Authentication Code) should provide authentication and integrity protection.

MAC security notions: chosen message attacks

The adversary has access to an oracle $M \mapsto \mathrm{MAC}_k(M)$.
He must compute a new MAC for:

- One message of his choice: existential forgery.
- A challenge message: selective forgery.
- Any message: universal forgery.

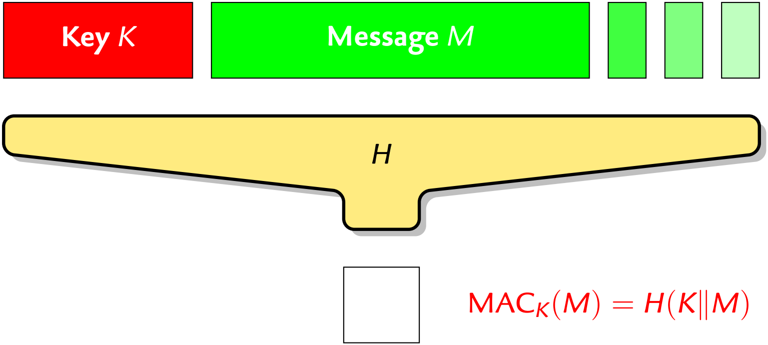
Secret-prefix MAC



$\mathrm{MAC}_K(M) = H(K \| M)$

- The hash function is expected to break the structure of the input
  - This is the secret-prefix construction
- MD5, SHA-1 cannot be used in this way...
  - But new designs *should* try to be safe in this mode

Secret-prefix MAC



$\mathsf{MAC}_K(M) = H(K \| M)$

- The hash function is expected to break the structure of the input
  - This is the secret-prefix construction
- MD5, SHA-1 cannot be used in this way...
  - But new designs *should* try to be safe in this mode

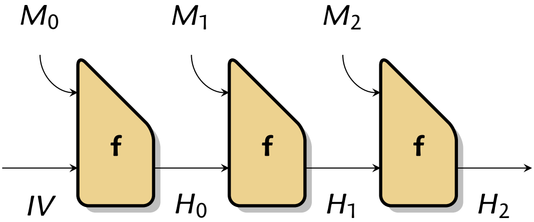
Overview of the attack

We want to extract key information from the state.

1. Use related queries to gather information about input and output of the round function $\mathcal{R}$
   - The compression function can be reduced to a small equation
2. Solve using linear algebra techniques
   - Using only two queries
   - Or using more queries to decrease the time complexity

Length extension attack

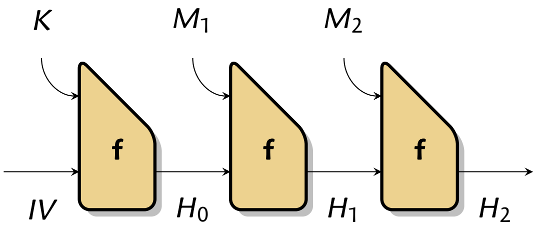
- Most hash function are based on a simple iteration.



- In secret-prefix MAC, we don't know the inner state
- But given the MAC of a known message, we can compute the MAC of some related messages (existential forgery).
- Solution: use a finalisation function.

Length extension attack

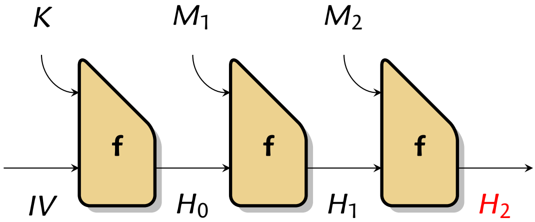
- Most hash function are based on a simple iteration.



- In secret-prefix MAC, we don't know the inner state
- But given the MAC of a known message, we can compute the MAC of some related messages (existential forgery).
- Solution: use a finalisation function.

Length extension attack

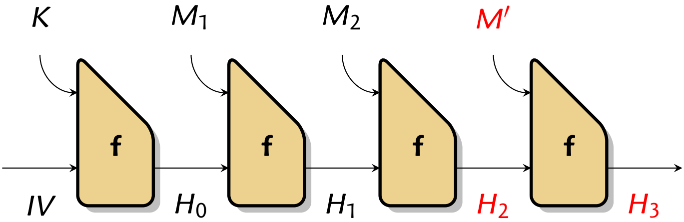
- Most hash function are based on a simple iteration.



- In secret-prefix MAC, we don't know the inner state
- But given the MAC of a known message, we can compute the MAC of some related messages (existential forgery).
- Solution: use a finalisation function.

Length extension attack

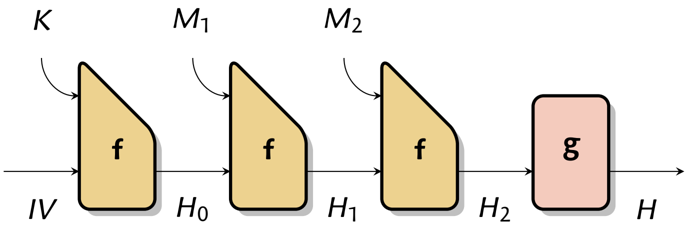
- Most hash function are based on a simple iteration.



- In secret-prefix MAC, we don't know the inner state
- But given the MAC of a known message, we can compute the MAC of some related messages (existential forgery).
- Solution: use a finalisation function.

Length extension attack

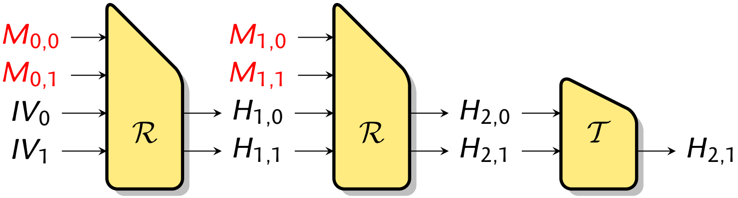
- Most hash function are based on a simple iteration.



- In secret-prefix MAC, we don't know the inner state
- But given the MAC of a known message, we can compute the MAC of some related messages (existential forgery).
- Solution: use a finalisation function.

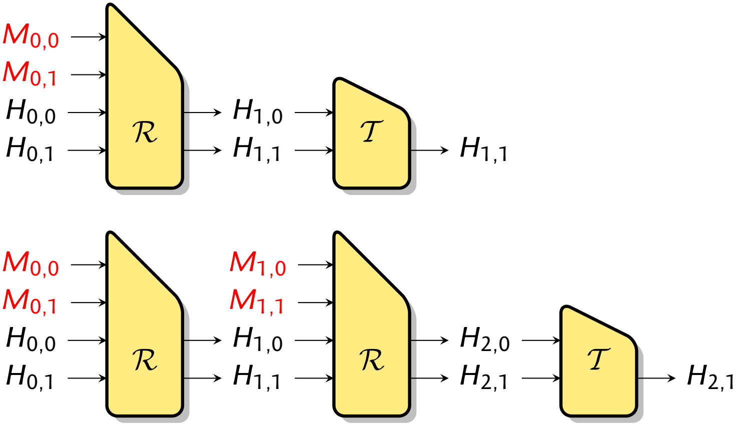
EDON-$\mathcal{R}$ *iteration*

- EDON-$\mathcal{R}$ is based on the chop-MD principle
  - The internal state is twice as big as the output
  - The finalization function is just a truncation.



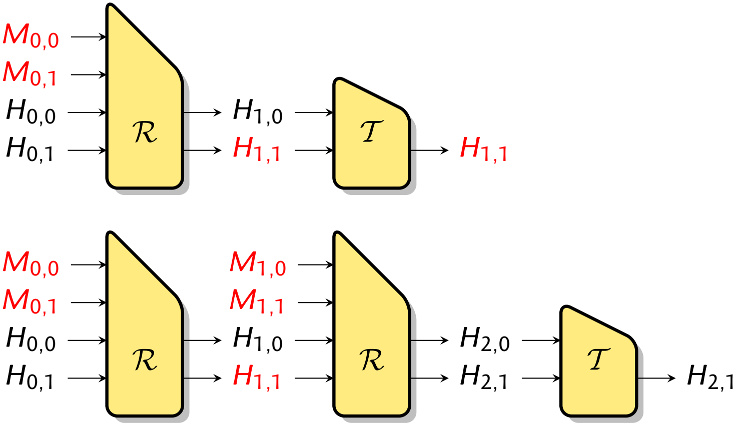
- Can we do something similar to the length-extension attack?

Using Related Queries



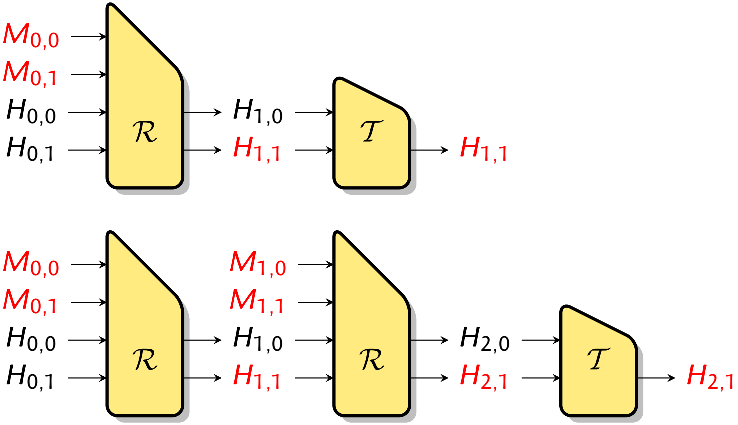
- We use two related messages: M is a prefix of M'
- $\mathrm{MAC}(M)$ gives $H_{1,1}$
- $\mathrm{MAC}(M')$ gives $H_{2,1}$

Using Related Queries



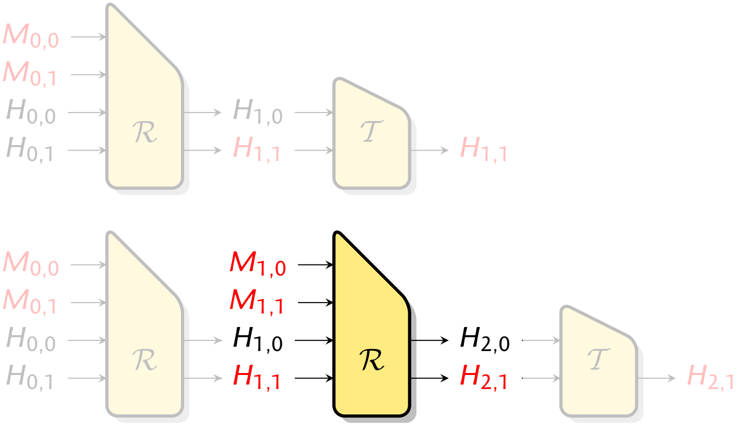
- We use two related messages: M is a prefix of M'
- $\text{MAC}(M)$ gives $H_{1,1}$
- $\text{MAC}(M')$ gives $H_{2,1}$

Using Related Queries



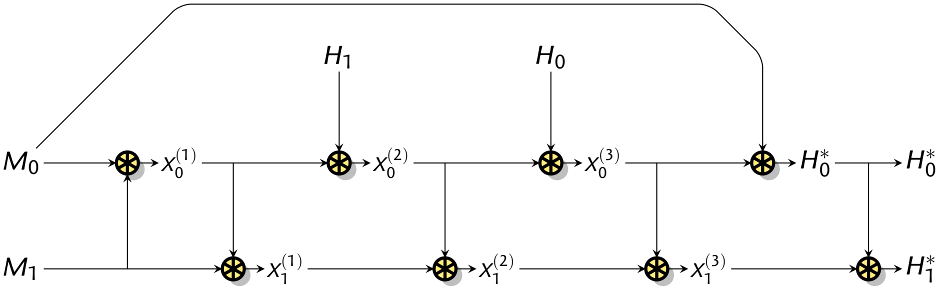
- We use two related messages: M is a prefix of M'
- $\mathrm{MAC}(M)$ gives $H_{1,1}$
- $\mathrm{MAC}(M')$ gives $H_{2,1}$

Using Related Queries



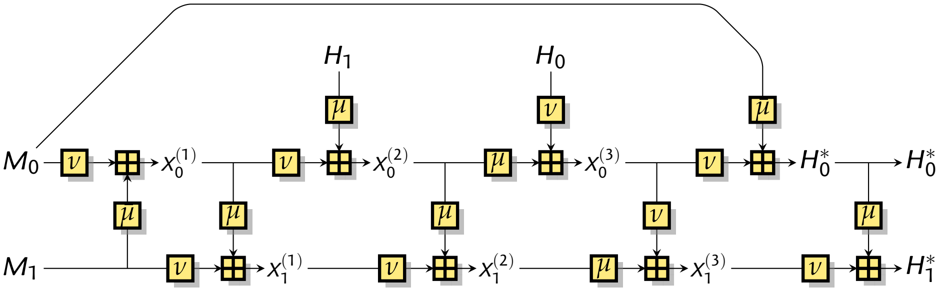
- We use two related messages: M is a prefix of M'
- $\text{MAC}(M)$ gives $H_{1,1}$
- $\text{MAC}(M')$ gives $H_{2,1}$

EDON-$\mathcal{R}$ Compression function



- Design based on a quasi-group operation $*$
- $*$ is a sum of two permutations, μ and ν

EDON-$\mathcal{R}$ Compression function



- Design based on a quasi-group operation $*$
- $*$ is a sum of two permutations, μ and ν

The Quasi-group Operation $*$

$$X * Y = \mu(X) + \nu(Y)$$

Inside μ

1. A linear step over $\mathbb{Z}_{2^w}^8$ $\qquad S_X = P_X(X)$

   $S_X^{[2]} = X^{[0]} + X^{[1]} + X^{[4]} + X^{[6]} + X^{[7]}$

2. Word-wise rotations $\qquad T_X = R_X(S_X)$

   $T_X^{[2]} = S_X^{[2]} \lll 8$

3. A linear step over $(\mathbb{F}_2^w)^8$ $\qquad \mu(X) = Q_X(T_X)$

   $\mu(X)^{[7]} = T_X^{[2]} \oplus T_X^{[3]} \oplus T_X^{[5]}$

The Quasi-group Operation $*$

$$X * Y = \mu(X) + \nu(Y)$$

Inside ν

1. A linear step over $\mathbb{Z}_{2^w}^8$ — $S_Y = P_Y(Y)$

   $S_Y^{[0]} = Y^{[0]} + Y^{[1]} + Y^{[3]} + Y^{[4]} + Y^{[5]}$

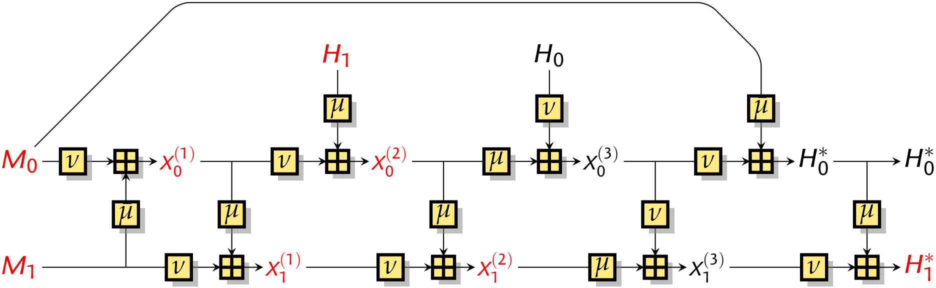
2. Word-wise rotations — $T_Y = R_Y(S_Y)$

   $T_Y^{[4]} = S_Y^{[4]} \lll 15$

3. A linear step over $(\mathbb{F}_2^w)^8$ — $\nu(Y) = Q_Y(T_Y)$

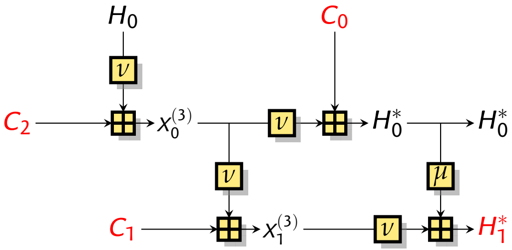
   $\nu(Y)^{[7]} = T_Y^{[4]} \oplus T_Y^{[6]} \oplus T_Y^{[7]}$

Simplify



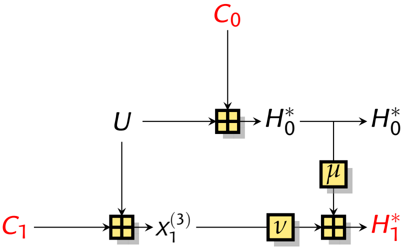
- We know M_0, M_1, H_1 and H_1^*
  - We can compute some parts
- $C_0 = \bar{\mu}(M_{1,0})$, $C_1 = \mu(X_1^{(2)})$, $C_2 = \mu(X_0^{(2)})$
- $U = \nu(X_0^{(3)})$ is the new unknown $\quad (H_0 = \nu^{-1}(\nu^{-1}(U) - C_2))$
- $H_1^* = (U + C_0) * (U + C_1)$
  - Relatively simple equation

Simplify



- We know M_0, M_1, H_1 and H_1^*
  - We can compute some parts
- $C_0 = \bar{\mu}(M_{1,0})$, $C_1 = \mu(X_1^{(2)})$, $C_2 = \mu(X_0^{(2)})$
- $U = \nu(X_0^{(3)})$ is the new unknown $(H_0 = \nu^{-1}(\nu^{-1}(U) - C_2))$
- $H_1^* = (U + C_0) * (U + C_1)$
  - Relatively simple equation

Simplify



- We know M_0, M_1, H_1 and H_1^*
  - We can compute some parts
- $C_0 = \bar{\mu}(M_{1,0})$, $C_1 = \mu(X_1^{(2)})$, $C_2 = \mu(X_0^{(2)})$
- $U = \nu(X_0^{(3)})$ is the new unknown $\qquad (H_0 = \nu^{-1}(\nu^{-1}(U) - C_2))$
- $H_1^* = (U + C_0) * (U + C_1)$
  - Relatively simple equation

$H_1^* = (U + C_0) * (U + C_1)$

The core of the attack is to solve this equation:

$$H_1^* = (U + C_0) * (U + C_1)$$

We propose two techniques:

1. A technique based on neutral words
2. We build an even simpler equation using several equations

Neutral words

- We identified a subspace of the input that does not affect the full output of $*$

$$
\begin{array}{llllllllll}
U_0 = [& 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 &] \\
U_1 = [& 2 & 2 & 2 & 2 & 2^{31}-3 & 2^{31}-3 & 0 & 2^{31}-1 &] \\
U_2 = [& 1 & 0 & 0 & 0 & 2^{31}-1 & 2^{31} & 0 & 2^{31} &]
\end{array}
$$

$$
\begin{aligned}
(X+\alpha U_0) * (Y+\beta U_0) \oplus X * Y &= [* \;\; * \;\; * \;\; * \;\; * \;\; 0 \;\; 0 \;\; 0] \\
(X+\alpha U_1) * (Y+\beta U_1) \oplus X * Y &= [* \;\; * \;\; * \;\; * \;\; * \;\; 0 \;\; * \;\; 0] \\
(X+\alpha U_2) * (Y+\beta U_2) \oplus X * Y &= [* \;\; * \;\; * \;\; * \;\; * \;\; * \;\; * \;\; 0]
\end{aligned}
$$

Using neutral words

1. Extend U_0, U_1, U_2 into a basis $U_0, \ldots, U_7$
   - Write the unknown in this basis $U = \sum_{i=0}^{7} \alpha_i U_i$

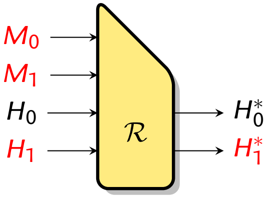
2. For each choice of $\alpha_3, \ldots \alpha_7$
   We know that $\alpha_0, \alpha_1, \alpha_2$ do not affect $H^{[7]}$
   Compute $H^{[7]}$ and skip if it is wrong.

3. Similarly, we can filter on $H^{[5]}$ after choosing α_2

4. And filter on $H^{[6]}$ after choosing α_1

Our results: Attack I

- Attack on the compression function:
  - Given M_0, M_1, H_1 and H_1^*, we can compute H_0 and H_0^*.



- We use two MAC queries to get H_1 and H_1^*.

	Queries	Time	Memory	Precomputation
EDON-$\mathcal{R}$224/256	2	2^{160}	-	-
EDON-$\mathcal{R}$384/512	2	2^{320}	-	-

Using more queries

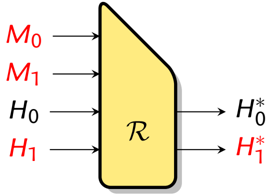
- We build two equations with *similar* C_0 constants
- We subtract the two equations
  - The μ terms *mostly* cancels out

$$
\begin{aligned}
H^{(i)} &= \mu(U + C_0^{(i)}) + \nu(U + C_1^{(i)}) \\
H^{(j)} &= \mu(U + C_0^{(j)}) + \nu(U + C_1^{(j)}) \\
H^{(i)} - H^{(j)} &\approx \nu(U + C_1^{(i)}) - \nu(U + C_1^{(j)})
\end{aligned}
$$

- We can remove the first layer of ν by linearity

Our results: Attack II

- Attack on the compression function:
  - Given M_0, M_1, H_1 and H_1^*, with a specially crafted message pair, we can build a simple equation involving H_0 and H_0^*.
  - With 10 related equations, we can recover three words of H_0.



- We need two MAC queries for each equation.

	Queries	Time	Memory	Precomputation
EDON-$\mathcal{R}$224/256	32	$\simeq 2^{30}$	-	2^{52}
EDON-$\mathcal{R}$384/512	32	$\simeq 2^{32}$	-	2^{100}

Thank you for listening!

Any questions?

Neutral words: example

$$X' = X + \alpha U_2, Y' = Y + \beta U_2$$
$$(X * Y)^{[7]} = (T_X^{[2]} \oplus T_X^{[3]} \oplus T_X^{[5]}) + (T_Y^{[4]} \oplus T_Y^{[6]} \oplus T_Y^{[7]})$$

$$T_X^{[2]} = (X^{[0]} + X^{[1]} + X^{[4]} + X^{[6]} + X^{[7]}) \lll 8$$
$$T_X^{[3]} = (X^{[2]} + X^{[3]} + X^{[5]} + X^{[6]} + X^{[7]}) \lll 13$$
$$T_X^{[5]} = (X^{[0]} + X^{[2]} + X^{[3]} + X^{[4]} + X^{[5]}) \lll 22$$
$$T_Y^{[4]} = (Y^{[0]} + Y^{[1]} + Y^{[3]} + Y^{[4]} + Y^{[5]}) \lll 15$$
$$T_Y^{[6]} = (Y^{[1]} + Y^{[2]} + Y^{[5]} + Y^{[6]} + Y^{[7]}) \lll 25$$
$$T_Y^{[7]} = (Y^{[0]} + Y^{[3]} + Y^{[4]} + Y^{[6]} + Y^{[7]}) \lll 27$$

Neutral words: example

$$X' = X + \alpha U_2, Y' = Y + \beta U_2$$
$$(X' * Y')^{[7]} = ({T'}_X^{[2]} \oplus {T'}_X^{[3]} \oplus {T'}_X^{[5]}) + ({T'}_Y^{[4]} \oplus {T'}_Y^{[6]} \oplus {T'}_Y^{[7]})$$

$$
\begin{aligned}
{T'}_X^{[2]} &= ({X'}^{[0]} + {X'}^{[1]} + {X'}^{[4]} + {X'}^{[6]} + {X'}^{[7]}) \lll 8 \\
{T'}_X^{[3]} &= ({X'}^{[2]} + {X'}^{[3]} + {X'}^{[5]} + {X'}^{[6]} + {X'}^{[7]}) \lll 13 \\
{T'}_X^{[5]} &= ({X'}^{[0]} + {X'}^{[2]} + {X'}^{[3]} + {X'}^{[4]} + {X'}^{[5]}) \lll 22 \\
{T'}_Y^{[4]} &= ({Y'}^{[0]} + {Y'}^{[1]} + {Y'}^{[3]} + {Y'}^{[4]} + {Y'}^{[5]}) \lll 15 \\
{T'}_Y^{[6]} &= ({Y'}^{[1]} + {Y'}^{[2]} + {Y'}^{[5]} + {Y'}^{[6]} + {Y'}^{[7]}) \lll 25 \\
{T'}_Y^{[7]} &= ({Y'}^{[0]} + {Y'}^{[3]} + {Y'}^{[4]} + {Y'}^{[6]} + {Y'}^{[7]}) \lll 27
\end{aligned}
$$

Neutral words: example

$$X' = X + \alpha U_2, Y' = Y + \beta U_2$$
$$(X' * Y')^{[7]} = (T'^{[2]}_X \oplus T'^{[3]}_X \oplus T'^{[5]}_X) + (T'^{[4]}_Y \oplus T'^{[6]}_Y \oplus T'^{[7]}_Y)$$

$$T'^{[2]}_X = (X^{[0]} + \alpha + X^{[1]} + X^{[4]} + \alpha(2^{31} - 1) + X^{[6]} + X^{[7]} + \alpha 2^{31}) \lll 8$$
$$T'^{[3]}_X = (X^{[2]} + X^{[3]} + X^{[5]} + \alpha 2^{31} + X^{[6]} + X^{[7]} + \alpha 2^{31}) \lll 13$$
$$T'^{[5]}_X = (X^{[0]} + \alpha + X^{[2]} + X^{[3]} + X^{[4]} + \alpha(2^{31} - 1) + X^{[5]} + \alpha 2^{31}) \lll 22$$
$$T'^{[4]}_Y = (Y^{[0]} + \beta + Y^{[1]} + Y^{[3]} + Y^{[4]} + \beta(2^{31} - 1) + Y^{[5]} + \beta 2^{31}) \lll 15$$
$$T'^{[6]}_Y = (Y^{[1]} + Y^{[2]} + Y^{[5]} + \beta 2^{31} + Y^{[6]} + Y^{[7]} + \beta 2^{31}) \lll 25$$
$$T'^{[7]}_Y = (Y^{[0]} + \beta + Y^{[3]} + Y^{[4]} + \beta(2^{31} - 1) + Y^{[6]} + Y^{[7]} + \beta 2^{31}) \lll 27$$

Neutral words: example

$$X' = X + \alpha U_2, Y' = Y + \beta U_2$$
$$(X' * Y')^{[7]} = (T'^{[2]}_X \oplus T'^{[3]}_X \oplus T'^{[5]}_X) + (T'^{[4]}_Y \oplus T'^{[6]}_Y \oplus T'^{[7]}_Y)$$

$$T'^{[2]}_X = (X^{[0]} + \cancel{\alpha} + X^{[1]} + X^{[4]} + \cancel{\alpha(2^{31} - 1)} + X^{[6]} + X^{[7]} + \cancel{\alpha 2^{31}}) \lll 8$$
$$T'^{[3]}_X = (X^{[2]} + X^{[3]} + X^{[5]} + \alpha 2^{31} + X^{[6]} + X^{[7]} + \alpha 2^{31}) \lll 13$$
$$T'^{[5]}_X = (X^{[0]} + \alpha + X^{[2]} + X^{[3]} + X^{[4]} + \alpha(2^{31} - 1) + X^{[5]} + \alpha 2^{31}) \lll 22$$
$$T'^{[4]}_Y = (Y^{[0]} + \beta + Y^{[1]} + Y^{[3]} + Y^{[4]} + \beta(2^{31} - 1) + Y^{[5]} + \beta 2^{31}) \lll 15$$
$$T'^{[6]}_Y = (Y^{[1]} + Y^{[2]} + Y^{[5]} + \beta 2^{31} + Y^{[6]} + Y^{[7]} + \beta 2^{31}) \lll 25$$
$$T'^{[7]}_Y = (Y^{[0]} + \beta + Y^{[3]} + Y^{[4]} + \beta(2^{31} - 1) + Y^{[6]} + Y^{[7]} + \beta 2^{31}) \lll 27$$