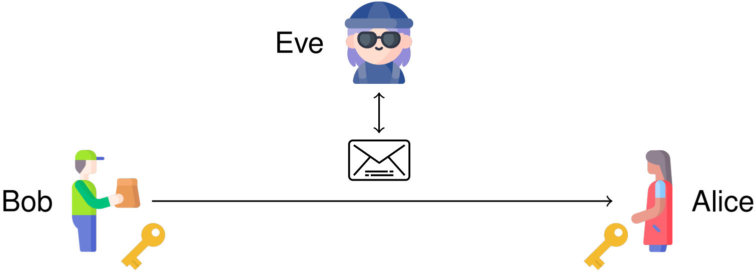
Analysis of AES-based and Arithmetization-Oriented Symmetric Cryptography Primitives

Augustin Bariant

Inria, Paris, France
ANSSI, Paris, France

PhD Defense

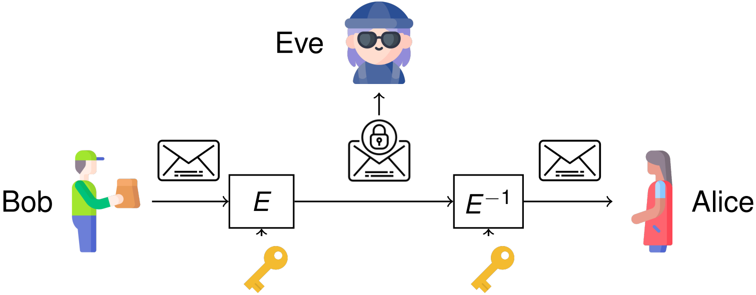
Symmetric cryptography: the basics



Symmetric cryptography

- Insecure communication channel.
- Eve controls the channel.
- Shared secret key.

Symmetric cryptography: the basics



Symmetric cryptography

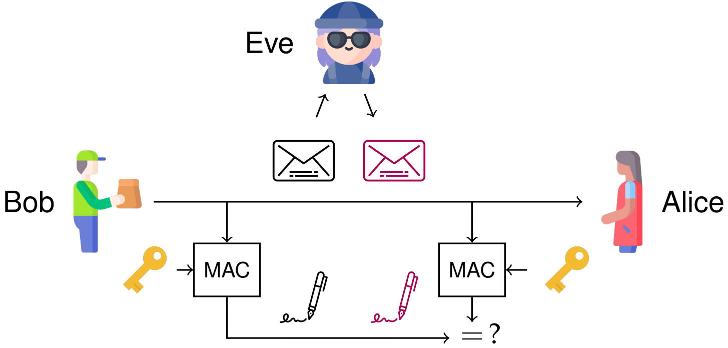
- Insecure communication channel.
- Eve controls the channel.
- Shared secret key.

Requirement: confidentiality

Eve shouldn't be able to recover the messages.

- Encryption $E_K : P \to C$.
- C gives no information on P.

Symmetric cryptography: the basics



Symmetric cryptography

- Insecure communication channel.
- Eve controls the channel.
- Shared secret key.

Requirement: integrity and authenticity

Eve shouldn't be able to modify the messages.

- Message Authentification Code (MAC).
- Hard to predict the tag.

Primitives

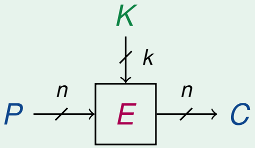
Modes of operation

General idea: split long messages in n-bit chunks ($n \approx 128$) and process each chunk.

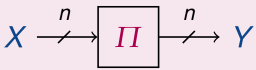
- Fixed-size primitives are used to process n-bit chunks.
- Modes provide provable confidentiality and/or authenticity.
- (Almost) only need to study the primitives.

Examples of such primitives:

Block ciphers



Public permutations

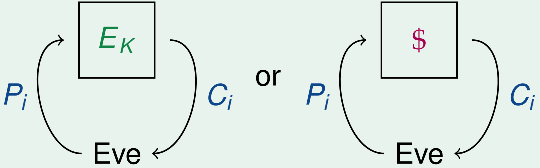


Cryptanalysis: finding the best attacks

Definition (cryptographic attack)

An algorithm that breaks a security claim more efficently that generic attacks.

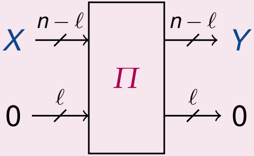
Distinguisher (against block ciphers)



Goal: distinguish oracles E_K and random \$.

- Generic attack: brute-force K ($\approx 2^k$).

CICO attacks (against public permutations)



Goal: Find X, Y s.t. $\Pi(X \| 0) = (Y \| 0)$.

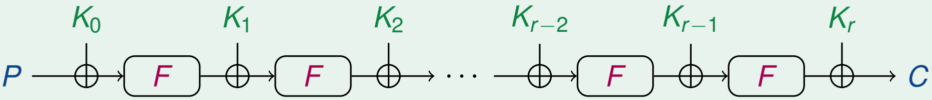
- Generic attack: brute-force X ($\approx 2^\ell$).

Primitive security rarely provable $\implies$ cryptanalysis is needed.

Iterated constructions for primitives

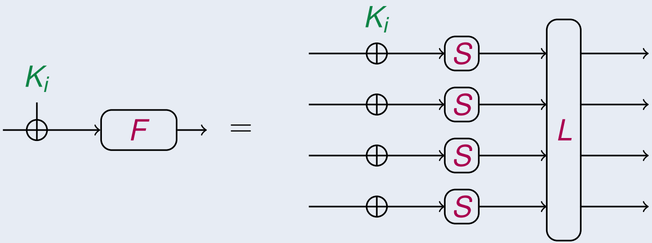
Popular design strategy: iterate a round function F and add subkeys/constants.

Key-alternating block ciphers



- Replace K_i by constants for a permutation.

Substitution-Permutation Networks (SPNs)



- S non-linear: confusion.
- L linear: diffusion.

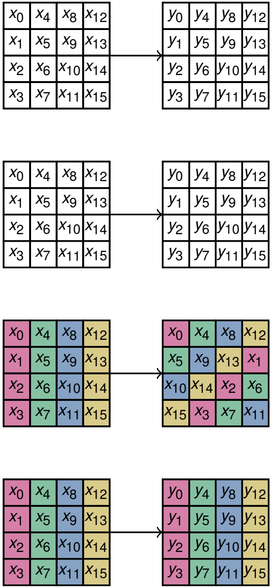
The AES and AES-based primitives

The AES block cipher: [DR, NIST'97]

- Standardized by the NIST in 2001.
- State of 4x4 bytes.
- Round function:

$$AK \rightarrow SB \rightarrow SR \rightarrow MC$$

- AES-128: 10 rounds.
- Security well-understood.



AddKey (AK):

- $y_i \leftarrow x_i \oplus rk_i$

SubBytes (SB):

- $y_i \leftarrow S(x_i)$

ShiftRows (SR):

- $\mathrm{Row}_i \leftarrow \mathrm{Row}_i \lll i$

MixColumns (MC):

- M: 4x4 matrix (MDS)
- $\mathrm{Col}_i \leftarrow M \times \mathrm{Col}_i$

The AES and AES-based primitives

The AES block cipher: [DR, NIST'97]

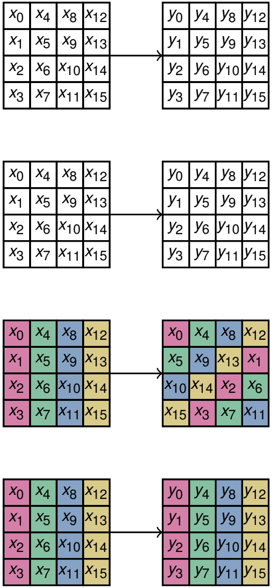
- Standardized by the NIST in 2001.
- State of 4x4 bytes.
- Round function:

$$AK \rightarrow SB \rightarrow SR \rightarrow MC$$

- AES-128: 10 rounds.
- Security well-understood.

AES-based primitives:

- Re-use the AES round function.
- Easier security analysis.
- Good performances (AES-NI).
- Ex: Deoxys-BC, Kiasu-BC, TNT-AES, Rocca ...



AddKey (AK):

- $y_i \leftarrow x_i \oplus rk_i$

SubBytes (SB):

- $y_i \leftarrow S(x_i)$

ShiftRows (SR):

- $\text{Row}_i \leftarrow \text{Row}_i \lll i$

MixColumns (MC):

- M: 4x4 matrix (MDS)
- $\text{Col}_i \leftarrow M \times \text{Col}_i$

Analysis of AES-based primitives: results

Cryptanalysis

- *Cryptanalysis of Forkciphers*: [B, David & Leurent, ToSC 2020]
  - Attack on full ForkAES.
  - Improved attacks on reduced ForkSkinny.
- *Truncated Boomerang Attacks and Application to AES-Based Ciphers*: [B & Leurent, EUROCRYPT 2023]
  - Best attacks on reduced Deoxys-BC and Kiasu-BC.
- *Improved Boomerang Attacks on 6-Round AES*: [B, Dunkelman, Keller, Leurent & Mollimard, EPRINT 2024]
  - Best boomerang key-recovery attack on reduced AES.

Design

- *Fast AES-based Universal Hash Functions and MACs*: [B, Baudrin, Leurent, Pernot, Perrin & Peyrin, ToSC 2024]
  - Fastest MAC of the literature on recent CPUs.

Arithmetization-Oriented (AO) primitives

Traditional primitives	*Arithmetization-Oriented primitives*
▶ Designed for bit-oriented platforms (computers, chips, ASIC, etc.).	▶ Designed for Zero-Knowledge Proofs and Multi-Party Computation protocols.
▶ Operate on bit sequences.	▶ Operate on large finite field elements.
▶ Low resource consumption (time, etc.).	▶ Low number of field multiplications.
▶ Several decades of cryptanalysis.	▶ $\leq$ 8 years of cryptanalysis.

Cryptanalysis of AO primitives: results

- *Algebraic Attacks against Some Arithmetization-Oriented Primitives*: [B, Bouvier, Leurent & Perrin, ToSC 2022]
  - Improved attacks against reduced Poseidon, Feistel-MiMC, Rescue-Prime.
  - Multivariate attack on full Ciminion.
- *The Algebraic Freelunch: Efficient Gröbner Basis Attacks against Arithmetization-Oriented Primitives*: [B, Boeuf, Lemoine, Manterola Ayala, Øygarden, Perrin & Raddum, CRYPTO 24]
  - Attacks threatening the security of Griffin, Arion & Anemoi.
- *A Univariate Attack on a Full Ciminion Instance*. [B, SAC'24]

Outline

Differential cryptanalysis [Biham & Shamir, CRYPTO'90]

Definition (differential)

A differential $\Delta_{\text{in}} \xrightarrow[E_K]{} \Delta_{\text{out}}$ has a probability:

$$\Pr\left[\Delta_{\text{in}} \xrightarrow[E_K]{} \Delta_{\text{out}}\right] = \Pr_{\substack{P \leftarrow \$ \\ K \leftarrow \$}}\left[E_K(P) \oplus E_K(P \oplus \Delta_{\text{in}}) = \Delta_{\text{out}}\right].$$

Differential distinguishing attack

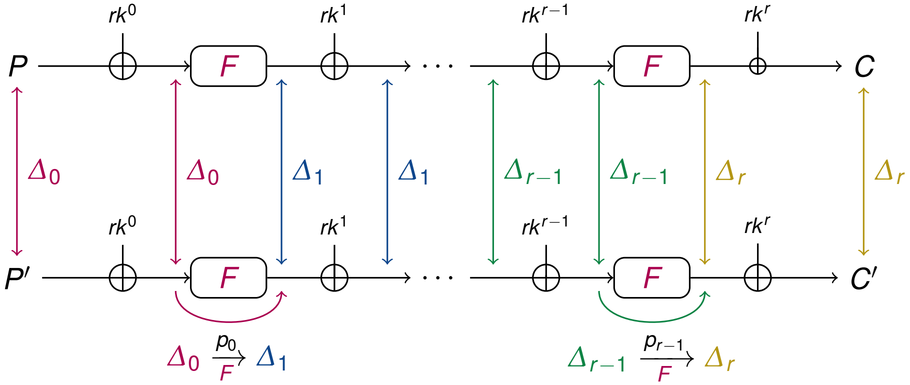
Setup: A differential $\Delta_{\text{in}} \xrightarrow[E_K]{} \Delta_{\text{out}}$ with $\Pr\left[\Delta_{\text{in}} \xrightarrow[E_K]{} \Delta_{\text{out}}\right] \gg 2^{-n}$.

Input: an oracle $O = E_K$ or $O = \$$.

- Ask the oracle for $O(P)$ and $O(P \oplus \Delta_{\text{in}})$ for many values of P.
- If the event $O(P) \oplus O(P \oplus \Delta_{\text{in}}) = \Delta_{\text{out}}$ happens frequently, $O = E_K$, else $O = \$$.

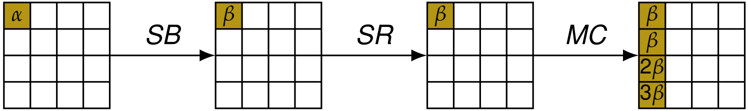
In practice, we find differentials $\Delta_{\text{in}} \xrightarrow[E_K]{} \Delta_{\text{out}}$ using differential trails.

Differential cryptanalysis [Biham & Shamir, CRYPTO'90]

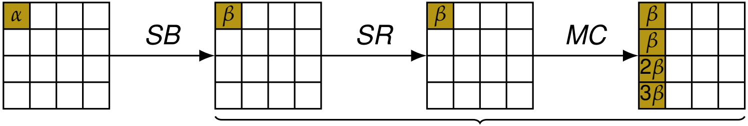


- $p_i = \Pr_{X \leftarrow \$}[F(X) \oplus F(X \oplus \Delta_i) = \Delta_{i+1}]$.
- $\Pr[\Delta_0 \xrightarrow[F]{} \Delta_1 \xrightarrow[F]{} \dots \xrightarrow[F]{} \Delta_r] = \prod p_i$. (Markov cipher & independent subkeys)
- $\Pr\left[\Delta_0 \xrightarrow[E_K]{} \Delta_r\right] \geq \prod p_i$.

Security of the AES against differential cryptanalysis

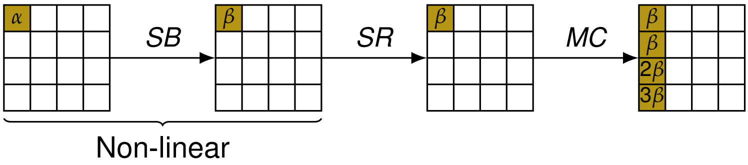


Security of the AES against differential cryptanalysis



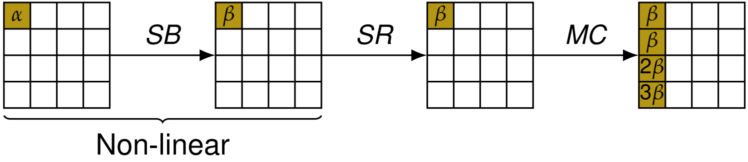
Linear $\implies$ Probability 1 differential.

Security of the AES against differential cryptanalysis



- Each S-box operates independently.
  - Inactive S-box $\rightarrow$ probability 1.
  - Active S-box $\rightarrow$ probability at most 2^{-6} (AES S-box property).

Security of the AES against differential cryptanalysis

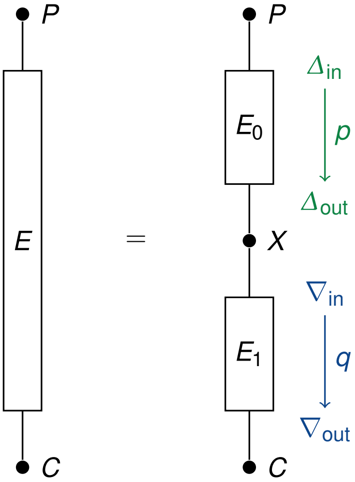


- Each S-box operates independently.
  - Inactive S-box → probability 1.
  - Active S-box → probability at most 2^{-6} (AES S-box property).
- Trail probability at most 2^{-6k} with k active S-boxes.
  - On AES, we can prove the minimal number of active S-boxes in a trail:

Number of rounds	1	2	3	4	5	6	7	8
Min. nb. of active S-boxes	1	5	9	25	26	30	34	50
Max. diff. trail probability	2^{-6}	2^{-30}	2^{-54}	2^{-150}	2^{-156}	2^{-180}	2^{-204}	2^{-300}

Outline

The boomerang attack [Wagner, FSE‘99]



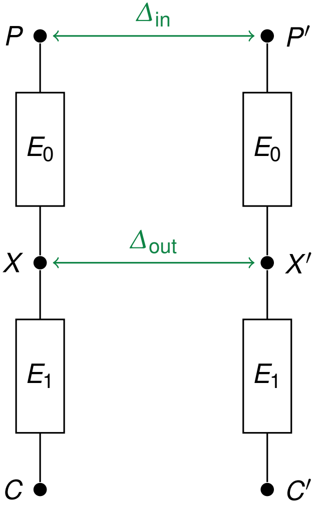
- Setup for the attack:
  - $E = E_1 \circ E_0$
  - $\Delta_{\text{in}} \xrightarrow[E_0]{p} \Delta_{\text{out}}$
  - $\nabla_{\text{in}} \xrightarrow[E_1]{q} \nabla_{\text{out}}$
- Two short differentials instead of a long one.
- Interesting when no long differential exist (ex: AES).

The boomerang attack

$$P \bullet \xleftrightarrow{\Delta_{\text{in}}} \bullet P'$$

- Select a random P.
- Select P' s.t. $P \oplus P' = \Delta_{\text{in}}$.

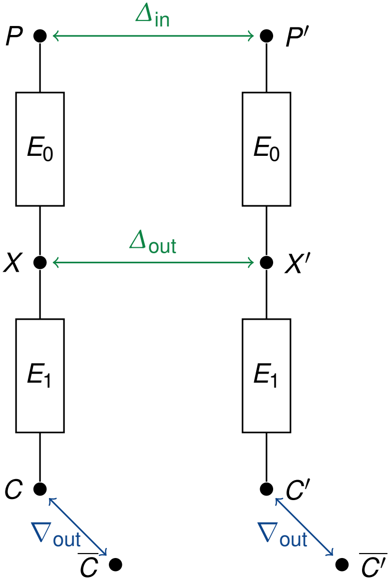
The boomerang attack



$$\Delta_{\text{in}} \xrightarrow[E_0]{p} \Delta_{\text{out}}$$

- Select a random P.
- Select P' s.t. $P \oplus P' = \Delta_{\text{in}}$.
- $\Pr[X \oplus X' = \Delta_{\text{out}}] = p$.

The boomerang attack

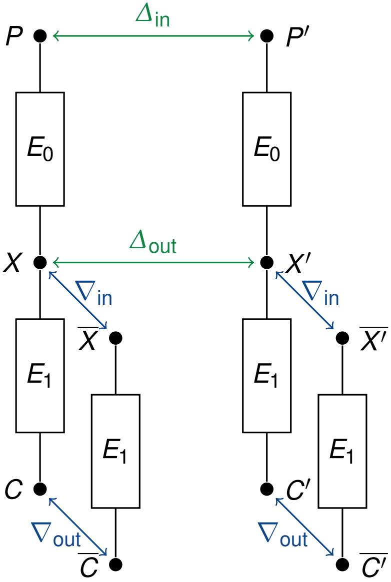


$$\Delta_{\text{in}} \xrightarrow[E_0]{p} \Delta_{\text{out}}$$

$$\nabla_{\text{in}} \xrightarrow[E_1]{q} \nabla_{\text{out}}$$

- Select a random P.
- Select P' s.t. $P \oplus P' = \Delta_{\text{in}}$.
- $\Pr[X \oplus X' = \Delta_{\text{out}}] = p$.
- Select $(\overline{C}, \overline{C'})$ s.t. $C \oplus \overline{C} = C' \oplus \overline{C'} = \nabla_{\text{out}}$.

The boomerang attack

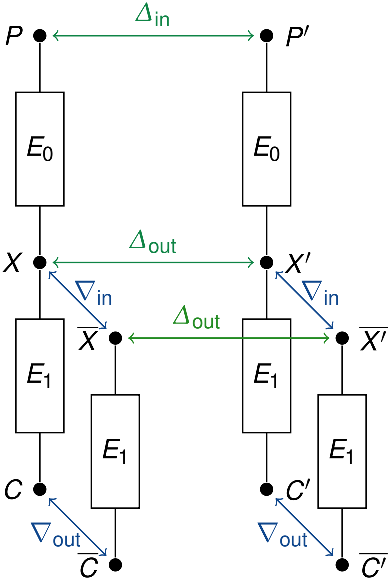


$$\Delta_{\text{in}} \xrightarrow[E_0]{p} \Delta_{\text{out}}$$

$$\nabla_{\text{in}} \xrightarrow[E_1]{q} \nabla_{\text{out}}$$

- Select a random P.
- Select P' s.t. $P \oplus P' = \Delta_{\text{in}}$.
- $\Pr[X \oplus X' = \Delta_{\text{out}}] = p$.
- Select $(\overline{C}, \overline{C'})$ s.t. $C \oplus \overline{C} = C' \oplus \overline{C'} = \nabla_{\text{out}}$.
- $\Pr[X \oplus \overline{X} = \nabla_{\text{in}}] = q$.
- $\Pr[X' \oplus \overline{X'} = \nabla_{\text{in}}] = q$.

The boomerang attack

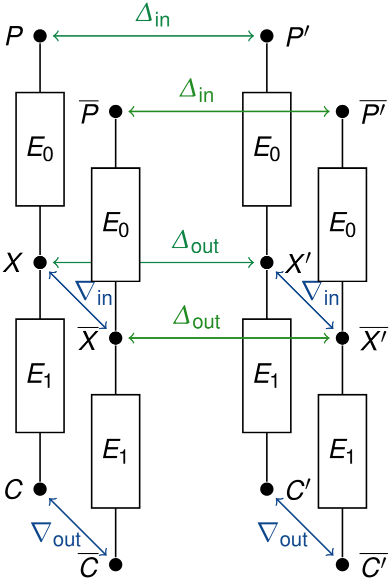


$$\Delta_{\text{in}} \xrightarrow[E_0]{p} \Delta_{\text{out}}$$

$$\nabla_{\text{in}} \xrightarrow[E_1]{q} \nabla_{\text{out}}$$

- Select a random P.
- Select P' s.t. $P \oplus P' = \Delta_{\text{in}}$.
- $\Pr[X \oplus X' = \Delta_{\text{out}}] = p$.
- Select $(\overline{C}, \overline{C'})$ s.t. $C \oplus \overline{C} = C' \oplus \overline{C'} = \nabla_{\text{out}}$.
- $\Pr[X \oplus \overline{X} = \nabla_{\text{in}}] = q$.
- $\Pr[X' \oplus \overline{X'} = \nabla_{\text{in}}] = q$.
- If this holds, then $\overline{X} \oplus \overline{X'} = \Delta_{\text{out}}$.

The boomerang attack

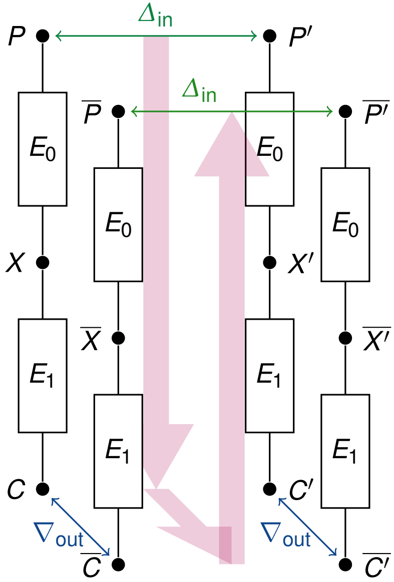


$$\Delta_{\text{in}} \xrightarrow[E_0]{p} \Delta_{\text{out}}$$

$$\nabla_{\text{in}} \xrightarrow[E_1]{q} \nabla_{\text{out}}$$

- Select a random P.
- Select P' s.t. $P \oplus P' = \Delta_{\text{in}}$.
- $\Pr[X \oplus X' = \Delta_{\text{out}}] = p$.
- Select $(\overline{C}, \overline{C'})$ s.t. $C \oplus \overline{C} = C' \oplus \overline{C'} = \nabla_{\text{out}}$.
- $\Pr[X \oplus \overline{X} = \nabla_{\text{in}}] = q$.
- $\Pr[X' \oplus \overline{X'} = \nabla_{\text{in}}] = q$.
- If this holds, then $\overline{X} \oplus \overline{X'} = \Delta_{\text{out}}$.
- $\Pr[\overline{P} \oplus \overline{P'} = \Delta_{\text{in}}] = p$.

The boomerang attack



$$\Delta_{in} \xrightarrow[E_0]{p} \Delta_{out}$$

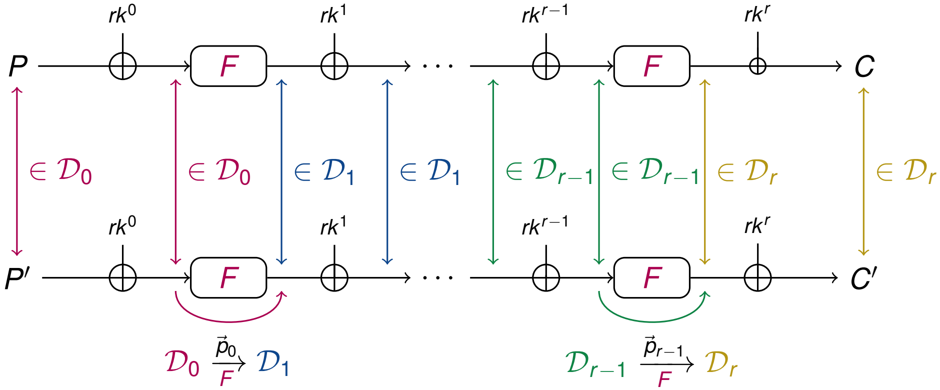
$$\nabla_{in} \xrightarrow[E_1]{q} \nabla_{out}$$

- Select a random P.
- Select P' s.t. $P \oplus P' = \Delta_{in}$.
- $\Pr[X \oplus X' = \Delta_{out}] = p$.
- Select $(\overline{C}, \overline{C'})$ s.t. $C \oplus \overline{C} = C' \oplus \overline{C'} = \nabla_{out}$.
- $\Pr[X \oplus \overline{X} = \nabla_{in}] = q$.
- $\Pr[X' \oplus \overline{X'} = \nabla_{in}] = q$.
- If this holds, then $\overline{X} \oplus \overline{X'} = \Delta_{out}$.
- $\Pr[\overline{P} \oplus \overline{P'} = \Delta_{in}] = p$.

Total boomerang probability: $p^2 q^2$.

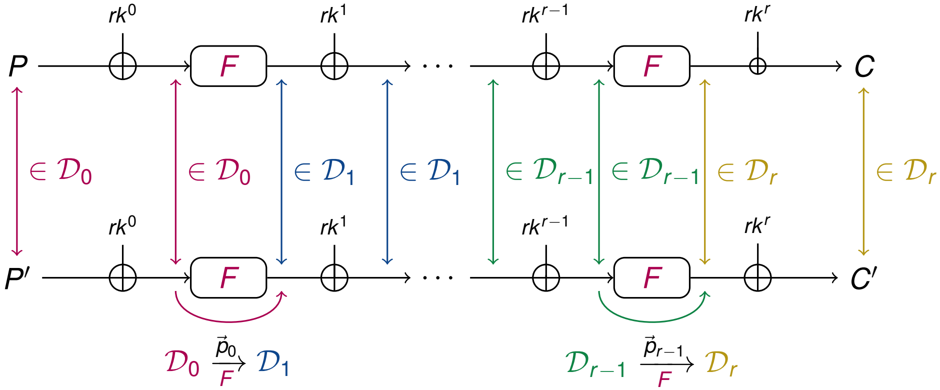
$p^2 q^2 \gg 2^{-n} \rightarrow$ Distinguisher

Truncated differential cryptanalysis [Knudsen, FSE'95]



- $\mathcal{D}_i$ are sets of differences.
- Trail probability $\vec{p} \approx \prod \vec{p}_i$.

Truncated differential cryptanalysis [Knudsen, FSE'95]

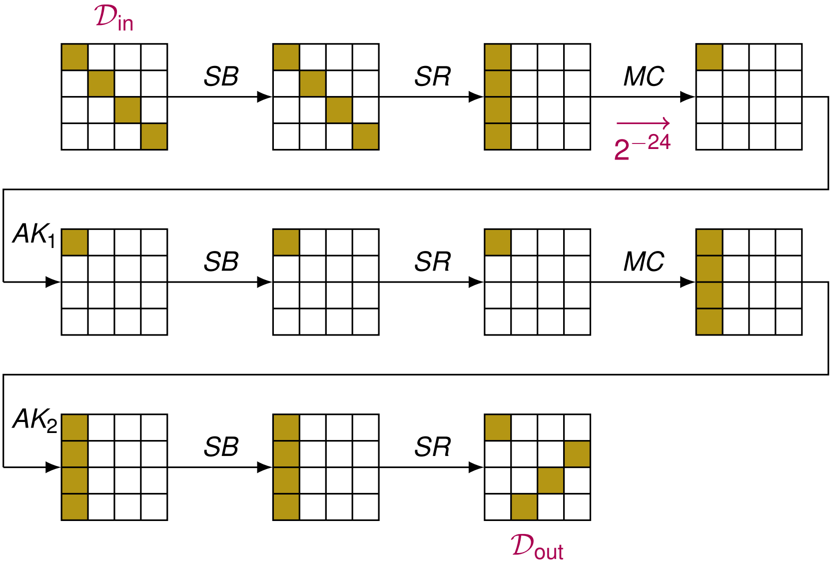


- $\mathcal{D}_i$ are sets of differences.
- Trail probability $\vec{p} \approx \prod \vec{p}_i$.

Structures (if $\mathcal{D}_0$ is a vectorial subspace)

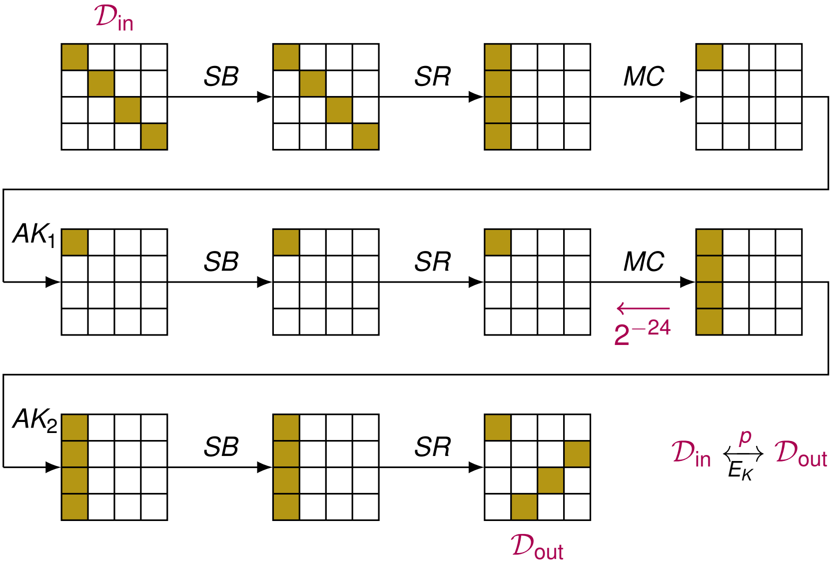
- Encrypt an affine space $P \oplus \mathcal{D}_0$.
- Look for $C, C' \in E(P \oplus \mathcal{D}_0)$ s.t. $C \oplus C' \in \mathcal{D}_r$.
- $|\mathcal{D}_0|$ encryptions but $|\mathcal{D}_0|^2/2$ pairs.

Truncated differential trail on 3-round AES



- $|\mathcal{D}_{\mathrm{in}}| = 2^{32}$.
- $|\mathcal{D}_{\mathrm{out}}| = 2^{32}$.
- $\vec{p} = 2^{-24}$.

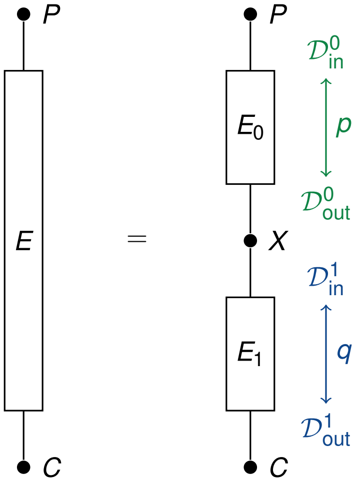
Truncated differential trail on 3-round AES



- $|\mathcal{D}_{\text{in}}| = 2^{32}$.
- $|\mathcal{D}_{\text{out}}| = 2^{32}$.
- $\vec{p} = 2^{-24}$.
- $\overleftarrow{p} = 2^{-24}$.

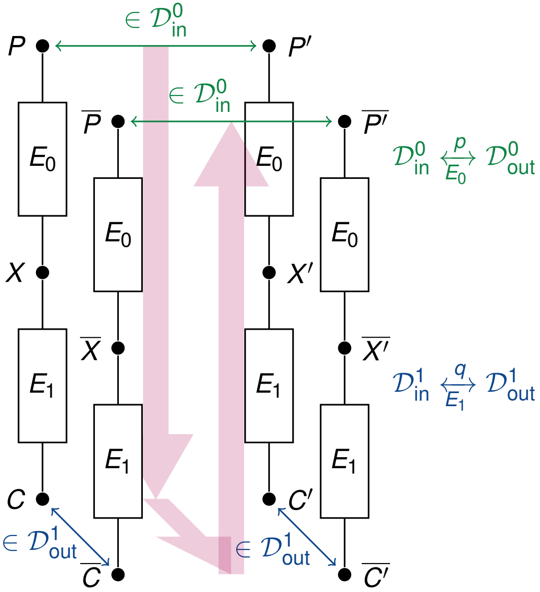
$$\mathcal{D}_{\text{in}} \overset{p}{\underset{E_K}{\longleftrightarrow}} \mathcal{D}_{\text{out}} = \begin{cases} \mathcal{D}_{\text{in}} \overset{\vec{p}}{\underset{E_K}{\longrightarrow}} \mathcal{D}_{\text{out}} \\ \mathcal{D}_{\text{out}} \overset{\overleftarrow{p}}{\underset{E_K^{-1}}{\longrightarrow}} \mathcal{D}_{\text{in}} \end{cases}$$

The truncated boomerang attack [B & Leurent, EUROCRYPT'22]



- Setup for the attack:
  - $E = E_1 \circ E_0$
  - $\mathcal{D}^0_{\text{in}} \underset{E_0}{\overset{p}{\longleftrightarrow}} \mathcal{D}^0_{\text{out}}$
  - $\mathcal{D}^1_{\text{in}} \underset{E_1}{\overset{q}{\longleftrightarrow}} \mathcal{D}^1_{\text{out}}$

The Truncated Boomerang Framework

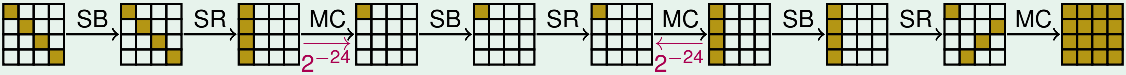


1. Select a random P_0:
   - Encrypt a structure $P_0 \oplus \mathcal{D}_{\text{in}}^0$.
2. For each $C \in E(P_0 \oplus \mathcal{D}_{\text{in}}^0)$:
   - Decrypt a structure $C \oplus \mathcal{D}_{\text{out}}^1$.
3. Look for $\overline{P}, \overline{P'} \in E^{-1}(E(P_0 \oplus \mathcal{D}_{\text{in}}^0) \oplus \mathcal{D}_{\text{out}}^1)$ s.t. $\overline{P} \oplus \overline{P'} \in \mathcal{D}_{\text{in}}^0$.
4. If needed, repeat with a different P_0.

- Boomerang switch probability: $r \geq |\mathcal{D}_{\text{in}}^1|^{-1}$.
- Total probability: $p_b = \vec{p} \cdot \overleftarrow{q}^2 \cdot r \cdot \overleftarrow{p}$.
- Random probability: $p_\$ = |\mathcal{D}_{\text{in}}^0| \cdot 2^{-n}$.
- Total structure size: $|\mathcal{D}_{\text{in}}^0||\mathcal{D}_{\text{out}}^1|$.

Example: 6-round AES distinguisher

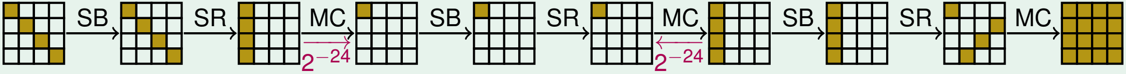
3-round AES truncated trail for E_0 and E_1



- $\overrightarrow{q} = \overleftarrow{q} = \overrightarrow{p} = \overleftarrow{p} = 2^{-24}$
- $r = 2^{-32}$
- $p_b = \overrightarrow{p} \cdot \overleftarrow{p} \cdot \overleftarrow{q}^2 \cdot r = 2^{-128}$
- $|\mathcal{D}^0_{\mathrm{out}}| = |\mathcal{D}^0_{\mathrm{in}}| = |\mathcal{D}^1_{\mathrm{out}}| = |\mathcal{D}^1_{\mathrm{in}}| = 2^{32}$
- $p_\$ = |\mathcal{D}^0_{\mathrm{in}}| \cdot 2^{-n} = 2^{-96}$

Example: 6-round AES distinguisher

3-round AES truncated trail for E_0 and E_1



- $\vec{q} = \overleftarrow{q} = \vec{p} = \overleftarrow{p} = 2^{-24}$
- $r = 2^{-32}$
- $p_b = \vec{p} \cdot \overleftarrow{p} \cdot \overleftarrow{q}^2 \cdot r = 2^{-128}$
- $|\mathcal{D}^0_{\text{out}}| = |\mathcal{D}^0_{\text{in}}| = |\mathcal{D}^1_{\text{out}}| = |\mathcal{D}^1_{\text{in}}| = 2^{32}$
- $p_\$ = |\mathcal{D}^0_{\text{in}}| \cdot 2^{-n} = 2^{-96}$

Distinguisher

Throw $Q = 2^{160}$ quartets to an oracle O using structures of size $|\mathcal{D}^0_{\text{in}}||\mathcal{D}^1_{\text{out}}| = 2^{64}$:

q quartets satisfy $\overline{P} \oplus \overline{P'} \in \mathcal{D}^0_{\text{in}}$:

- If $q \approx Qp_\$ = 2^{64} \rightarrow O = \$$.
- If $q \approx Q(p_\$ + p_b) = 2^{64} + 2^{32} \rightarrow O =$ 6R AES.

$$T = D \approx \frac{Q}{|\mathcal{D}^0_{\text{in}}||\mathcal{D}^1_{\text{out}}|} = 2^{96}.$$

Generic formulas

Distinguisher formula

- Signal-to-noise $\sigma = \frac{p_b}{p_\$}$.
- $Q = \max(1, 1/\sigma)/p_b$ quartets.

$$T = D = \frac{2Q}{|\mathcal{D}^0_{\text{in}}| \cdot |\mathcal{D}^1_{\text{out}}|}$$

Key-recovery formula

Our approach: same trail, each quartet suggests ℓ candidates for κ bits of key.

- Updated signal-to-noise ratio: $\tilde{\sigma} = \frac{p_b}{p_\$} \frac{2^\kappa}{\ell}$.
- $Q = \max(1, 1/\tilde{\sigma})/p_b$ quartets.

$$D = \frac{2Q}{|\mathcal{D}^0_{\text{in}}| \cdot |\mathcal{D}^1_{\text{out}}|}$$

Applications

Straightforward truncated boomerang attacks: [B & Leurent, EUROCRYPT'22]

- Kiasu-BC: improved 8-round key-recovery attack.
- Deoxys-BC: improved key-recovery attacks in different instances.
  - Minimize the complexity formulas in a MILP model.
- TNT-AES: marginal distinguisher.
- 6-round AES: new distinguishing, key-recovery (with or without secret S-box) attacks.

Follow-up work: [B, Dunkelman, Keller, Leurent & Mollimard, EPRINT 2024]

- 6-round AES: improved boomerang key-recovery attacks.
  - Best key-recovery boomerang attack on AES in the literature.
  - Combination with the retracing boomerang attack.

Outline

Design of Message Authentification Codes (MACs)

- Built from Universal Hash Functions (UHF)s.
- A UHF is a family of functions:

$$H_K : A \to B \text{ for } K \in \mathcal{K}.$$

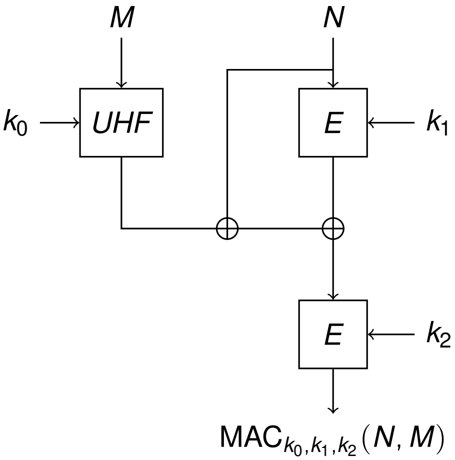
Definition (ε-AU UHFs)

$H_K : A \to B$ for $K \in \mathcal{K}$ is ε-almost-universal if:

$$\forall m \neq m' \in A,\ |\{K \in \mathcal{K} : H_K(m) = H_K(m')\}| \leq \varepsilon|\mathcal{K}|.$$

Ex: ECWDM.

[Cogliati & Seurin, CRYPTO'16]



Design of AES-based constructions

AES New Instructions (AES-NI)

- Instruction set proposed by Intel in 2008.
- 1 AESENC instruction = 1 AES round:

$$SB \rightarrow SR \rightarrow MC \rightarrow AK\,.$$

- Speed comparable to a 128-bit XOR/ADD instruction on modern processors.
- Exploited in new designs for exceptional performance.

Definition (Rate of an AES-based UHF/MAC)

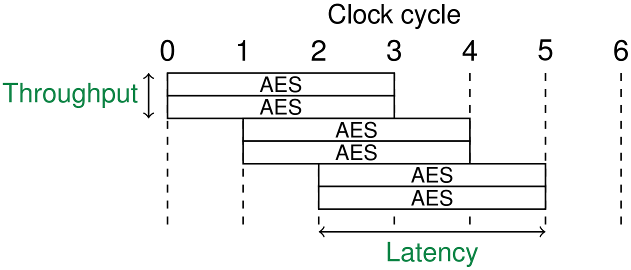
The rate is the number of AES-NI instructions per 128-bit message block.

- Rate 4: PelicanMAC, PC-MAC, AEGIS-128L. [DR:EPRINT‘05. MT:FSE‘06. WC:SAC‘13]
- Rate 3: Tiaoxin-346 (AD only). [Nikolić, CAESAR‘14]
- Rate 2: Jean-Nikolić, Rocca (AD only). [JN:FSE‘16. SLNKI:FSE‘22]

Scheduling of AES-NI instructions

On modern processors:

- Throughput: 2 AES per cycle.
- Latency: 3-4 cycles.



Theoretical bound

Rate-r constructions require $\geq \frac{r}{2}$ cycles per 128 bits of message.

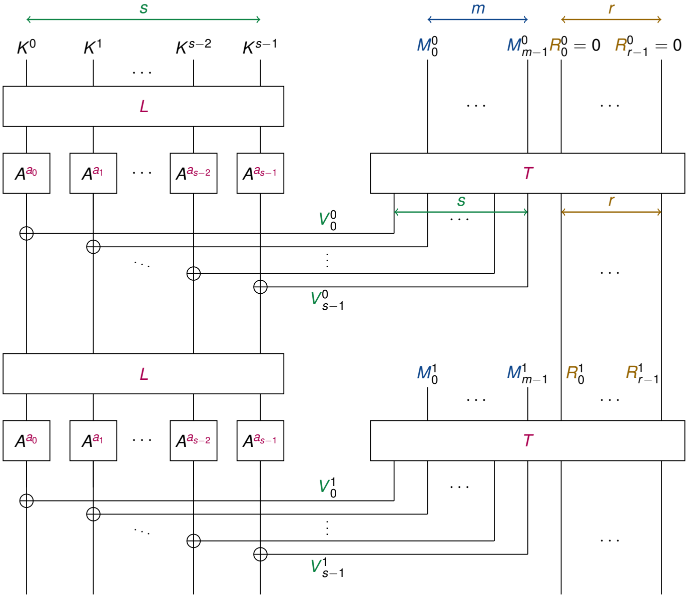
- Observation: existing rate-2 UHFs are slower than this bound (bad parallelization).

Our approach

Design a parallelization-oriented rate-2 AES-based UHF, and convert it to a MAC.

- Goal: reach the bound of 1 cycle/128-bit (= 0.0625 cycles/byte).

Our framework of UHF candidates



- Inspired by SPNs and tweakable block ciphers.
- L and T: binary matrices.
- Lots of varying parameters.

Procedure for finding fast ε-AU candidates

Procedure: generate many random candidates of the framework. For each of them:

- Check the security with MILP.
- Check the performance.
- Keep candidates that are secure and performant.

Security check

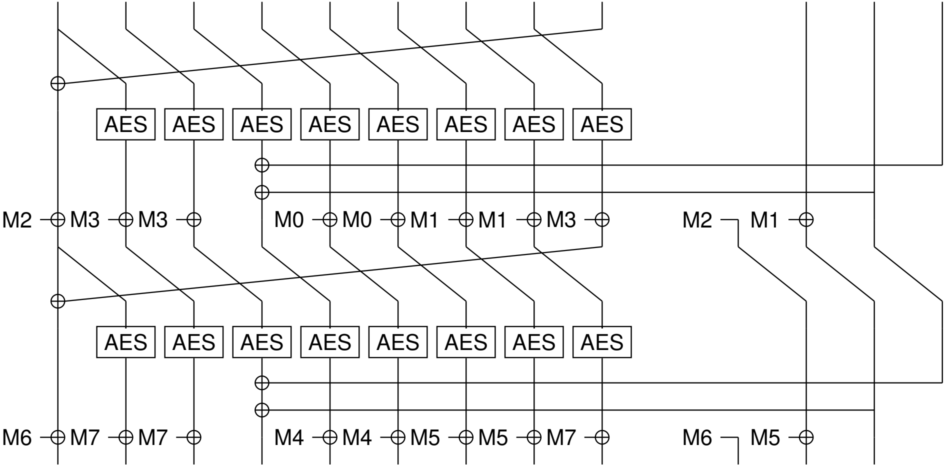
Heuristic: A candidate is ε-AU if no high probability differential $\Delta_M \to 0$ exists.

- Find the best differential trail leading to a collision with MILP.
- Secure if the number of active S-boxes is ≥ 22 (trail probability $\leq 2^{-22\times 6} = 2^{-132}$).

Performance check

- Automatically generate a C implementation and compile.
- Benchmark the candidate on the fly.

Round function of LeMac's UHF



Security	*Performance*
$\geq$ 26 active S-boxes.	Rate 2 with good parallelization.

Performance comparison

Cipher	Rate	Theoretical bound	Speed (cycles per byte)					
			Intel Ice Lake			**AMD Zen3**		
			1kB	16kB	256kB	1kB	16kB	256kB
GCM (AD only)	-	-	0.699	0.311	0.286	0.794	0.470	0.451
AEGIS128L (AD only)	4	0.125	0.416	0.208	0.195	0.358	0.183	0.173
Tiaoxin-346 v2 (AD only)	3	0.094	0.328	0.131	0.121	0.311	0.121	0.109
Rocca (AD only)	2	0.063	0.528	0.171	0.149	0.393	0.139	0.124
Jean-Nikolić	2	0.063	0.307	0.126	0.113	0.312	0.111	0.098
LeMac	2	0.063	0.289	0.082	0.068	0.309	0.085	0.072

LeMac: fastest existing MAC on modern processors.

Outline

Arithmetization-Oriented primitives

Reminders

- AO primitives operate on $\mathbb{F}_q$.
- $\mathbb{F}_q$ is too large to be exhausted by an attacker.
- AO primitives use a low number of field multiplications.

They must be designed to resist algebraic attacks.

Algebraic attacks

The polynomial solving attack is composed of two steps:

Modeling

Represent the primitive with a polynomial system $\mathcal{P}$.

- A solution to $\mathcal{P}$ leads to the key or to a CICO solution.
- Highly primitive-dependant.
- Not trivial to find the best modeling.

$$\mathcal{P} = \begin{cases} P_1(X_1, \ldots X_n) = 0 \\ \vdots \\ P_n(X_1, \ldots X_n) = 0 \end{cases}$$

Solving

Find $(X_1, \ldots X_n) \in \mathbb{F}_q^n$ which solves $\mathcal{P}$.

- Use state-of-the-art algorithms from computer algebra.
- Generic complexity formulas.
- Recover the key or a CICO solution.

Solving polynomial systems: the univariate case

One univariate equation of degree d in $\mathbb{F}_q$:

$$\mathcal{P} = \left\{ P(X) = 0. \right.$$

- Solving complexity quasi-linear in d: $\mathcal{O}(d \log(q) \log(d) \log(\log(d)))$ operations.

Remarks

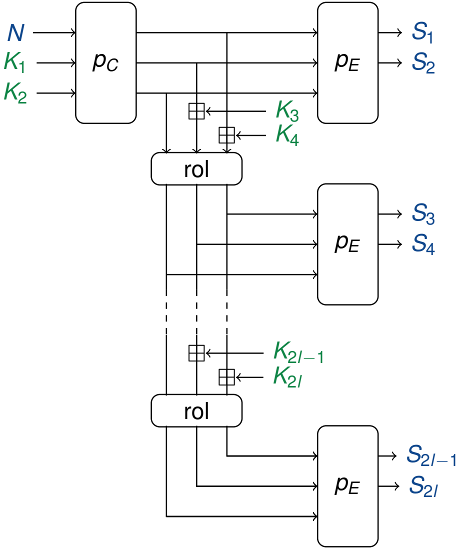
- Cheaper than factorisation ($\mathcal{O}(d^{1.815})$).
- Cheaper than multivariate solving.

Ciminion (limited data variant) [Dobraunig & al, EC'21]

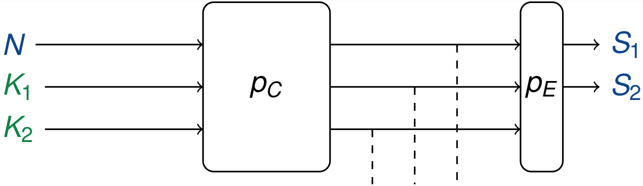
- Nonce-based stream cipher.
  - N different every query.
  - For each N, generate a series of S_i
- Secret subkeys $K_i \in \mathbb{F}_q$.
- Security based on truncated outputs.
- p_E and p_C permutations of $\mathbb{F}_q^3$.
  - p_E and p_E^{-1} of degrees $\approx q^{\frac{1}{12}}$.
  - p_C and p_C^{-1} of degrees $\approx q^{\frac{2}{3}}$.

Security claim of the designers

Attack complexity $> q$ (with $< \sqrt{q}$ data queries).



Application: the Ciminion limited data variant [B, SAC'24]

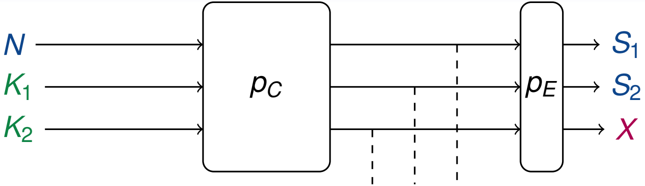


Modeling

1. The attacker queries 2 blocks S_1 and S_2 under the nonce N.

Solving

Application: the Ciminion limited data variant [B, SAC'24]

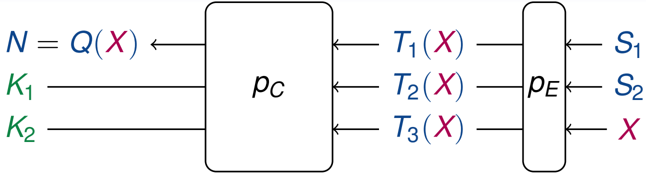


Modeling

1. The attacker queries 2 blocks S_1 and S_2 under the nonce N.

Solving

Application: the Ciminion limited data variant [B, SAC'24]



Modeling

1. The attacker queries 2 blocks S_1 and S_2 under the nonce N.
2. The attacker computes $Q(X) = p_C^{-1} \circ p_E^{-1}(S_1, S_2, X)$ of degree $\approx q^{\frac{1}{12}} q^{\frac{2}{3}} = q^{\frac{3}{4}}$.
   - Evaluate $p_C^{-1} \circ p_E^{-1}$ on $\mathbb{F}_q[X]^3$ instead of $\mathbb{F}_q^3$.

Solving

1. The attacker computes the roots of $Q(X) - N$ in $\tilde{\mathcal{O}}(q^{\frac{3}{4}})$.
2. The attacker recovers of the value for X and inverts $p_E \circ p_C$ to recover K_1, K_2.

Solving polynomial systems: the multivariate case

$$\begin{cases} P_1(X_1, \ldots X_n) = 0 \\ \vdots \\ P_n(X_1, \ldots X_n) = 0 \end{cases}$$

Definition (Ideal spanned by the polynomial system)

The ideal $\langle P_1, \ldots P_n \rangle$ is composed of all polynomials P such that:

$$\exists Q_1, \ldots Q_n \in \mathbb{F}_q[X_1, \ldots X_n], \qquad P(X_1, \ldots X_n) = \sum_{i=1}^{n} P_i Q_i.$$

General idea for solving

Find a univariate polynomial $P(X_1) \in \langle P_1, \ldots P_n \rangle$:

- In particular, $P(X_1) = 0$ if $P_i(X_1, \ldots X_n) = 0$ for all i.
- Compute the roots of $P(X_1)$ in $\mathbb{F}_q$ using univariate solving and recover values for X_1.

Solving multivariate systems with Gröbner bases

$$\begin{cases} P_1(X_1,\ldots X_n) = 0 \\ \vdots \\ P_n(X_1,\ldots X_n) = 0 \end{cases}$$

- $\deg(P_i) = d_i$.
- $\omega \leq 3$ is the matrix multiplication exponent.

Step 1: F5

Compute a *grevlex* Gröbner basis with F5, in: $\mathcal{O}\left(\binom{1+\sum_{i=1}^{n} d_i}{n}^{\omega}\right)$ [Faugère, ISSAC‘02]

- Loose complexity bound.

Step 2: Change of order

Convert it into a *lex* Gröbner basis, in: $\mathcal{O}\left(n\left(\prod_{i=1}^{n} d_i\right)^3\right)$ [FGLM, JSC‘93]

- Better complexity bounds under additional hypotheses.
- The *lex* Gröbner basis contains a univariate polynomial of the ideal.

Solving polynomial systems: applications

Algebraic attacks on arithmetization-oriented ciphers: [BBLP, ToSC'22]

- Motivation: set of challenges launched by the Ethereum Fundation.
- Contributions: improved modeling and implementation of all attacks.

Univariate attacks

- Poseidon, Feistel-MiMC: CICO attacks a reduced versions.
  - 2 rounds bypassed in the modeling of Poseidon.

Multivariate attacks

- Rescue-Prime: improved CICO attack on a reduced version.
  - 2 steps bypassed in the modeling.
- Ciminion: improved key-recovery attack.
  - New multivariate modeling of the cipher $\Longrightarrow$ full-round attack in some settings.

The FreeLunch attack [BBLMØPR, CRYPTO'24]

Modeling step

Generate a multivariate system.

→

FreeLunch step 1: modeling

Model the primitive with a well-chosen weighted monomial order:

- The system is directly a Gröbner basis.
- Negligible complexity (vs step 2).

Solving step 1: F5

$$\mathcal{O}\left(\binom{1+\sum_{i=1}^{n} d_i}{n}^{\omega}\right)$$

→

Solving step 2: Change of order

$$\tilde{\mathcal{O}}\left(d_1\left(\prod_{i=2}^{n} d_i\right)^{\omega}\right)$$

- Requires additional hypotheses.

[BNS, ISSAC'22]

→

FreeLunch step 2: finding a univariate equation

1. Compute a multiplication matrix in $\mathcal{O}(?)$.
2. Compute its characteristic polynomial in

$$\tilde{\mathcal{O}}\left(d_1\left(\prod_{i=2}^{n} d_i\right)^{\omega}\right)$$

3. Compute its roots.

The FreeLunch attack: results [BBLMØPR, CRYPTO'24]

FreeLunch step 2: complexity analysis

1. Multiplication matrix computation:
   - Complexity estimated experimentally.
   - Theoretical but loose upper bound:

$$\mathcal{O}\left(n\left(\prod_{i=1}^{n} d_i\right)^3\right) \quad \text{[FGLM, JSC'93]}$$

2. Characteristic polynomial computation:

$$\tilde{\mathcal{O}}\left(d_1\left(\prod_{i=2}^{n} d_i\right)^{\omega}\right)$$

   - Tight bound.

The FreeLunch attack: results [BBLMØPR, CRYPTO'24]

FreeLunch step 2: complexity analysis

1. Multiplication matrix computation:
   - Complexity estimated experimentally.
   - Theoretical but loose upper bound:

$$\mathcal{O}\left(n\left(\prod_{i=1}^{n} d_i\right)^3\right) \quad \text{[FGLM, JSC'93]}$$

2. Characteristic polynomial computation:

$$\tilde{\mathcal{O}}\left(d_1\left(\prod_{i=2}^{n} d_i\right)^{\omega}\right)$$

   - Tight bound.

Target	α / e	Number of branches: 2	3	4	5	6	8	≥ 12
Griffin	3	-	120	112	-	-	76	64
	5	-	141	110	-	-	81	74
Arion	3	-	128	134	114	119	98	-
	5	-	132	113	118	122	101	-
α-Arion	3	-	104	84	88	92	98	-
	5	-	83	87	91	94	101	-
Anemoi	3	118	-	-	-	-	-	-

Theoretical complexity of 2 ($\log_2$).

Work in progress: Can we find a better theoretical bound for 1?

Conclusion: results of this thesis

- Design (AES-based primitives):
  - Conception of the fastest AES-based MAC on modern processors (LeMac).
- Cryptanalysis (AES-based primitives):
  - The truncated boomerang framework: best attacks against Kiasu-BC and Deoxys-BC.
  - Best boomerang key-recovery attacks against 6-round AES.
  - Truncated differential attacks against full ForkAES.
- Cryptanalysis (arithmetization-oriented primitives):
  - The FreeLunch attack: a new algebraic attack framework.
  - Multivariate attacks against full Ciminion, Griffin, and Arion.
  - Univariate attack against full Ciminion.

Conclusion: results of this thesis

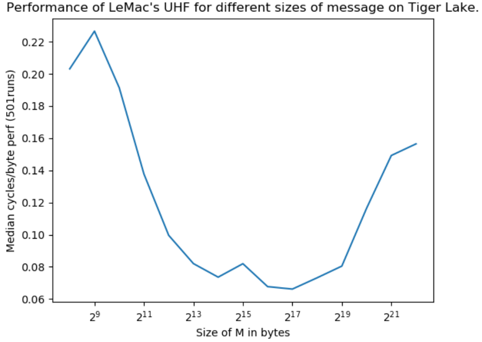
- Design (AES-based primitives):
  - Conception of the fastest AES-based MAC on modern processors (LeMac).
- Cryptanalysis (AES-based primitives):
  - The truncated boomerang framework: best attacks against Kiasu-BC and Deoxys-BC.
  - Best boomerang key-recovery attacks against 6-round AES.
  - Truncated differential attacks against full ForkAES.
- Cryptanalysis (arithmetization-oriented primitives):
  - The FreeLunch attack: a new algebraic attack framework.
  - Multivariate attacks against full Ciminion, Griffin, and Arion.
  - Univariate attack against full Ciminion.

Thank you for your attention.

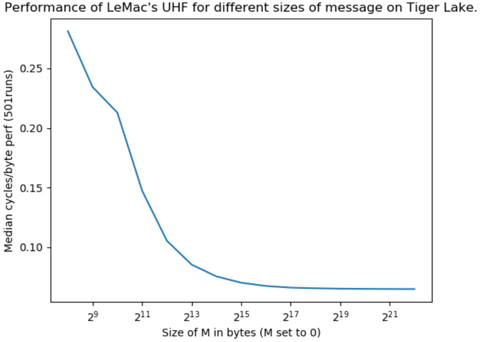
6-round AES results

	Type	Data		Time	Ref
Distinguishers	Yoyo	$2^{122.8}$	ACC	$2^{121.8}$	[RBH, AC'17]
	Exchange attack	2^{84}	ACC	2^{83}	[Bardeh, EPRINT'19]
	Truncated differential	$2^{89.4}$	CP	$2^{96.5}$	[BGL, ToSC'20]
	Truncated boomerang	2^{87}	ACC	2^{87}	[B & Leurent, EC'22]
Key-recovery	Partial-sum	2^{32}	CP	2^{48}	[FKLSSWW, FSE'00]
	Boomerang	2^{71}	ACC	2^{71}	[Biryukov, AES'04]
	Mixture	2^{26}	CP	2^{80}	[BDKRS, JoC'20]
	Retracing boomerang	2^{55}	ACC	2^{80}	[DKRS, EC'20]
	Truncated boomerang	2^{59}	ACC	2^{61}	[B & Leurent, EC'22]
	Boomerang	2^{51}	ACC	2^{68}	[BDKLM, EPRINT'24]
	Boomerang	2^{51}	ACC	2^{66}	[BDKLM, EPRINT'24]
	Boomerang	2^{57}	ACC	2^{61}	[BDKLM, EPRINT'24]
Secret S-Box KR	Square	2^{64}	CP	2^{90}	[TKKL, FSE'15]
	Truncated boomerang	2^{94}	ACC	2^{94}	[B & Leurent, EC'22]

LeMac's UHF performance for different message sizes



(Message M set to random)



(Message M set to 0)

- On Tiger-Lake: L1 cache: 48 kB, L2 cache: 1.25 MB.

Solving univariate systems: details

One equation $P(X) = 0$ of degree d in $\mathbb{F}_q$.

Observations

- We look for $r \in \mathbb{F}_q$ s.t. $P(r) = 0$.
- $r \in \mathbb{F}_q$ implies that r is solution of $X^q - X = 0$ (i.e. $r^q = r$).
- Therefore, r is a root of $R(X) = \gcd(X^q - X, P(X))$.

- Idea: efficiently compute $R(X) = \gcd(X^q - X, P(X))$.
  - Compute $Q(X) = X^q \bmod P(X)$ using fast exponentiation ($\log(q)$ steps).
  - Each step costs $\mathcal{O}(d \log(d) \log(\log(d)))$ with fast polynomial multiplication.
  - Compute $\gcd(Q(X) - X, P(X))$ in $\mathcal{O}(d \log(d)^2 \log(\log(d)))$ operations.
- $R(X)$ is of small degree, and can be factored efficiently.

Total complexity: $\mathcal{O}(d \log(q) \log(d) \log(\log(d)))$.

WIP: multiplication matrix computation for FreeLunch systems (result 1)

FreeLunch system (Gröbner Basis) $\mathcal{P} = \begin{cases} {X_1}^{d_1} & = P_1(X_1, X_2, \ldots, X_r), \\ {X_2}^{d_2} & = P_2(X_1, X_2, \ldots, X_r), \\ & \vdots \\ {X_r}^{d_r} & = P_r(X_1, X_2, \ldots, X_r), \end{cases}$

- Denote $d = \prod_{i=1}^{n} d_i$.

Theoretical bound : $\mathcal{O}(nd^3) \rightsquigarrow$ WIP results: $\mathcal{O}(nd^3/d_1)$

WIP: multiplication matrix computation for FreeLunch systems (principle)

Goal: compute the reduction of $\text{Surface}_1 = \{X_1{}^{d_1} m \mid m = X_2{}^{\alpha_2} \dots X_r{}^{\alpha_r}, 0 \leq \alpha_i \leq d_i - 1\}$.
Idea: reduce all monomials of $S = \text{Surface}_1 \cup \dots \cup \text{Surface}_r$ in ascending order.

- Store the monomial reductions in a reduction table.
- Use the reduction table to speed up following reductions.
- $|S|$ reductions, each costing d^2 lookups.

Naïvely, this leads to a bound $\mathcal{O}(|S| d^2) \approx \mathcal{O}(nd^3)$.

Our improvement: we remark that X_1 divides most of the monomials in $\overline{S} = S \setminus \text{Surface}_1$.

- Exactly a fraction $1 - \frac{1}{d_1}$ of $\overline{S}$.
- When X_1 divides a monomial $m \in \overline{S}$, reducing m costs d^2/d_1 lookups.
  - Instead of d^2 lookups.
  - Using sparsity in the multiplication matrix by X_1.
- We also remark that $|\text{Surface}_1| = d^2/d_1$.

Total complexity: $\mathcal{O}(|\overline{S}|/d_1 \times d^2 + |\overline{S}| \times d^2/d_1 + |\text{Surface}_1| d^2) \approx \mathcal{O}(nd^3/d_1)$

WIP: multiplication matrix computation for Griffin

In practical modelings, semi-triangular systems $\mathcal{P} = \begin{cases} {X_1}^{d_1} & = P_1(X_1, X_2, \ldots, X_r), \\ {X_2}^{d_2} & = P_2(X_1), \\ {X_3}^{d_3} & = P_3(X_1, X_2), \\ & \vdots \\ {X_r}^{d_r} & = P_r(X_1, X_2, \ldots, X_{r-1}), \end{cases}$

- In the case of Griffin, $\mathrm{wt}(P_i) < \mathrm{wt}(X_i)$ for $i = 2, \ldots r$.
- With this property, reductions are easy to compute:

Matrix multiplication computation complexity: $\tilde{\mathcal{O}}(d^2/d_1)$ instead of $\tilde{\mathcal{O}}(d^3/d_1)$

Remarks

- Quasi-linear in the size of the dense part of the multiplication matrix.
- Much faster than the characteristic polynomial computation.
- WIP: does it work on Arion and Anemoi?