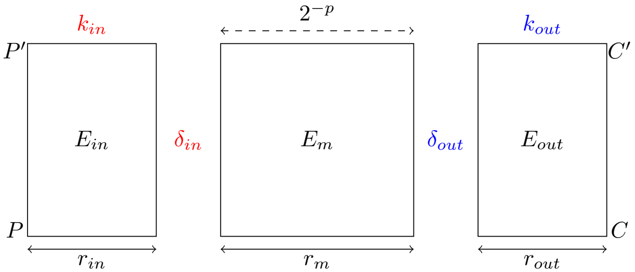
The State-test Technique on Families of Differential Attacks

Dounia M'Foukh[1], María Naya-Plasencia[1], Patrick Neumann[1]

[1]Inria Paris

Key-recovery attack

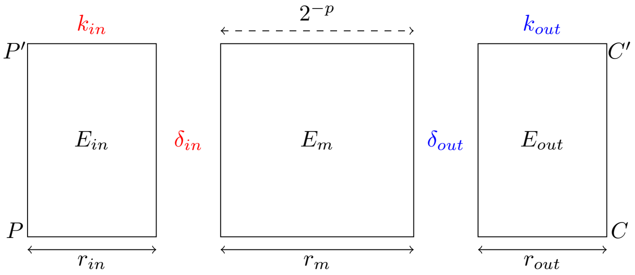
Let $E = E_{out} \circ E_m \circ E_{in}$ be a block cipher.



- Find candidate triplets $(P, P', k_{in} \cup k_{out})$ that imply δ_{in} and δ_{out}.
- The time complexity depends in part on the size of k_{in} and k_{out}.

Key-recovery attack

Let $E = E_{out} \circ E_m \circ E_{in}$ be a block cipher.



- Find candidate triplets $(P, P', k_{in} \cup k_{out})$ that imply δ_{in} and δ_{out}.
- The time complexity depends in part on the size of k_{in} and k_{out}.

$\rightsquigarrow$ We want to **minimize** the size of k_{in} and k_{out}

Table of contents

Context

State-test technique

History: Introduced in Meet-in-the-Middle and Impossible Differential attacks [DSP07,BNS14].

Context

State-test technique

History: Introduced in Meet-in-the-Middle and Impossible Differential attacks [DSP07,BNS14].
Goal: Lower the time complexity of the key recovery part by guessing internal state bits instead of key bits thus **compacting** the information.

Context

State-test technique

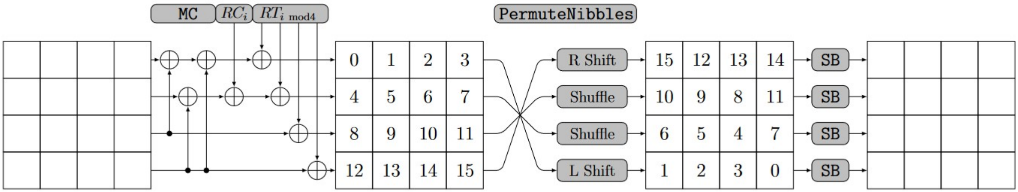
History: Introduced in Meet-in-the-Middle and Impossible Differential attacks [DSP07,BNS14].
Goal: Lower the time complexity of the key recovery part by guessing internal state bits instead of key bits thus **compacting** the information.

Where can it be applied?

1. Impossible Differential.
2. Differential Meet-in-the-Middle [AKM+24].
3. Classical Differential :
   - Case without a counter: Need to solve a system of equations efficiently.
   - Case with a counter: How take into account the information on the key in the counter ?
4. Differential Linear.

Description of CRAFT

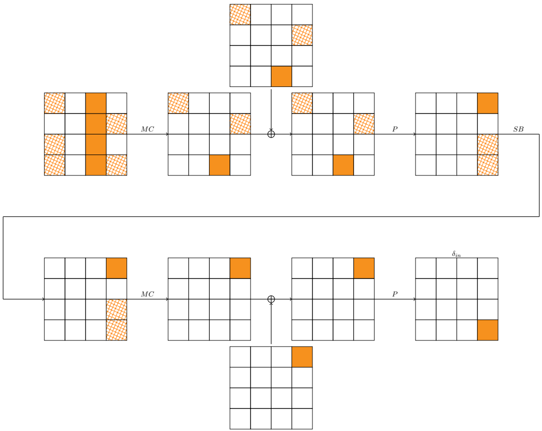
CRAFT [BLMR19], published in ToSC in 2019, is a lightweight tweakable block cipher operating on a 64-bit block, a 128-bit key $(K_0||K_1)$, and a 64-bit tweak T.



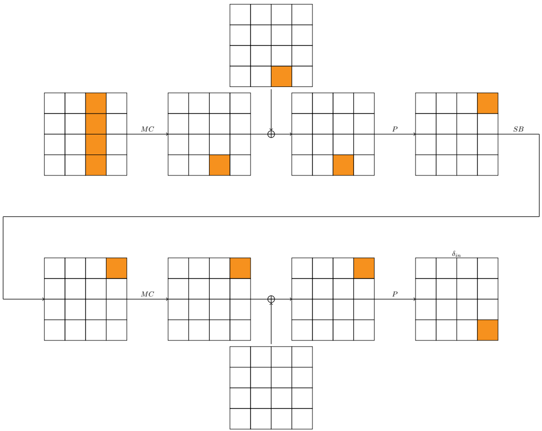
And the MixColumn operation is a matrix multiplication by :

$$M = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

A quick example

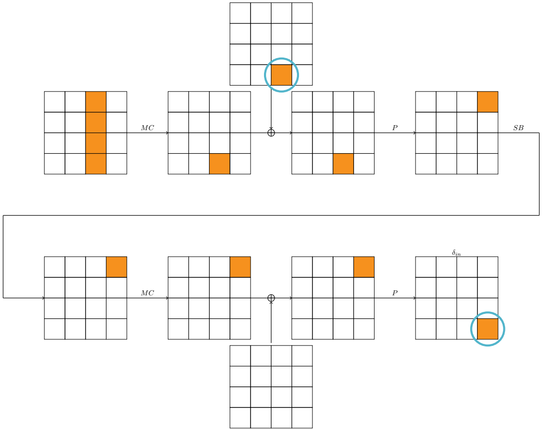


A quick example



- Gives non-linear equations over the key bits,
- Reduces the size of k_{in} and k_{out}.

A quick example

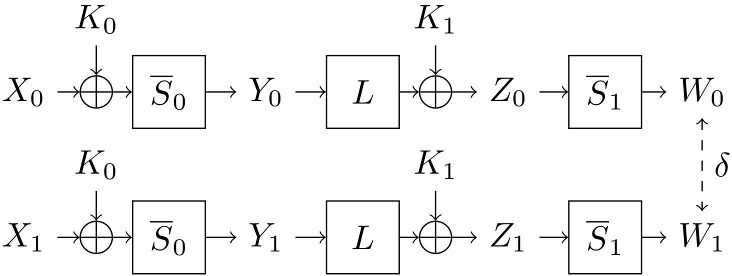


- Gives non-linear equations over the key bits,
- Reduces the size of k_{in} and k_{out}.

Framework

When is the state-test useful?

For a SPN with a linear layer denoted L.



- Γ_{mask} : set of all masks such that $\langle \gamma, Z_i \rangle$ is an active bit with respect to δ.
- Γ_{diff} : set of differences of $Z_0 \oplus Z_1$ giving a difference δ.

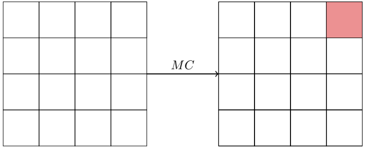
$$\Longrightarrow L^T(\Gamma_{mask}) \not\subset L^{-1}(\Gamma_{diff}).$$

Back to our example

For CRAFT, the linear layer L is a matrix multiplication by :

$$M^T = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix} \text{ and } M = M^{-1} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Transitions through a Mixcolumn operation:



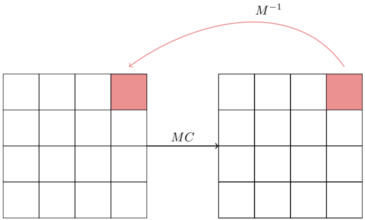
- ■ : words with differences.
- ▧ : words needed to compute the value of the active word.

Back to our example

For CRAFT, the linear layer L is a matrix multiplication by :

$$M^T = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix} \text{ and } M = M^{-1} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Transitions through a Mixcolumn operation:



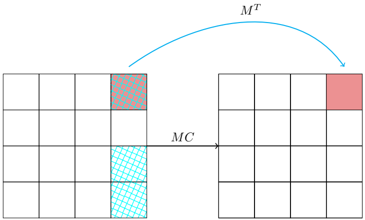
- ■ : words with differences.
- ⊠ : words needed to compute the value of the active word.

Back to our example

For CRAFT, the linear layer L is a matrix multiplication by :

$$M^T = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix} \text{ and } M = M^{-1} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Transitions through a Mixcolumn operation:



- ■ : words with differences.
- ⊠ : words needed to compute the value of the active word.

1. Introduction to the state-test technique
2. Hint on how to deal with the state-test equations
3. An application on the block cipher PRIDE

How to exploit the information: First case

If the number of triplets $(P, P', k_{in} \cup k_{out})$ is lower than the size of $k_{in} \cup k_{out}$.

- For each triplet, we recover a system of equations over the unknown key bits.
- Main idea: Find, in parallel, partial solutions of the system to then find an efficient way to solve the whole system.

$\rightsquigarrow$ **List merging** problems.

Example: The a_i are constants depending on the plaintexts/ciphertexts and known key bits.

$$\begin{cases} a_1 = S(k_1 + S(a_2 + k_0)) + S(S(a_3 + k_0 + S(k_2 + a_4)) \\ a_5 = S(a_2 + k_0 + S(a_3 + k_1)) + S(k_2 + a_4) \\ a_6 = S(k_1 + S(k_2 + a_3 + S(k_3))) + S(a_7 + k_0). \end{cases}$$

How to exploit the information: First case

If the number of triplets $(P, P', k_{in} \cup k_{out})$ is lower than the size of $k_{in} \cup k_{out}$.

- For each triplet, we recover a system of equations over the unknown key bits.
- Main idea: Find, in parallel, partial solutions of the system to then find an efficient way to solve the whole system.

$\rightsquigarrow$ **List merging** problems.

Example: The a_i are constants depending on the plaintexts/ciphertexts and known key bits.

$$\begin{cases} a_1 = S(k_1 + S(a_2 + k_0)) + S(S(a_3 + k_0 + S(k_2 + a_4)) \\ a_5 = S(a_2 + k_0 + S(a_3 + k_1)) + S(k_2 + a_4) \\ a_6 = S(k_1 + S(k_2 + a_3 + S(k_3))) + S(a_7 + k_0). \end{cases}$$

How to exploit the information: First case

If the number of triplets $(P, P', k_{in} \cup k_{out})$ is lower than the size of $k_{in} \cup k_{out}$.

- For each triplet, we recover a system of equations over the unknown key bits.
- Main idea: Find, in parallel, partial solutions of the system to then find an efficient way to solve the whole system.

$\rightsquigarrow$ **List merging** problems.

Example: The a_i are constants depending on the plaintexts/ciphertexts and known key bits.

$$\begin{cases} a_5 = S(a_2 + k_0 + S(a_3 + k_1)) + S(k_2 + a_4) \\ a_6 = S(k_1 + S(k_2 + a_3 + S(k_3))) + S(a_7 + k_0). \end{cases}$$

How to exploit the information: First case

If the number of triplets $(P, P', k_{in} \cup k_{out})$ is lower than the size of $k_{in} \cup k_{out}$.

- For each triplet, we recover a system of equations over the unknown key bits.
- Main idea: Find, in parallel, partial solutions of the system to then find an efficient way to solve the whole system.

$\rightsquigarrow$ **List merging** problems.

Example: The a_i are constants depending on the plaintexts/ciphertexts and known key bits.

$$\begin{cases} a_1 = S(k_1 + S(a_2 + k_0)) + S(S(a_3 + k_0 + S(k_2 + a_4)) \\ a_5 = S(a_2 + k_0 + S(a_3 + k_1)) + S(k_2 + a_4) \\ a_6 = S(k_1 + S(k_2 + a_3 + S(k_3))) + S(a_7 + k_0). \end{cases}$$

How to exploit the information: Second case

Greedy Substitution :

- Rewrite some of the equations such as to isolate some of the unknowns as function of others;
- **Best case :** guess the remaining unknowns and solve the rest of the equations;
- **Otherwise :** Pre-compute the solutions of the system for the remaining equations.

Example: The a_i are constants depending on the plaintexts/ciphertexts and known key bits.

$$\begin{cases} a_1 = S(k_1 + S(a_2 + k_0)) + S(S(a_3 + k_0 + S(k_2 + a_4)) \\ k_2 = S^{-1}(a_5 + S(a_2 + k_0 + S(a_3 + k_1))) + a_4 \\ a_6 = S(k_1 + S(k_2 + a_3 + S(k_3))) + S(a_7 + k_0). \end{cases}$$

How to exploit the information: Second case

Greedy Substitution :

- Rewrite some of the equations such as to isolate some of the unknowns as function of others;
- **Best case :** guess the remaining unknowns and solve the rest of the equations;
- **Otherwise :** Pre-compute the solutions of the system for the remaining equations.

Example: The a_i are constants depending on the plaintexts/ciphertexts and known key bits.

$$\begin{cases} a_1 = S(k_1 + S(a_2 + k_0)) + S(S(a_3 + k_0 + \\ \quad + S(S^{-1}(a_5 + S(a_2 + k_0 + S(a_3 + k_1))))) \\ k_2 = S^{-1}(a_5 + S(a_2 + k_0 + S(a_3 + k_1))) + a_4 \\ a_6 = S(k_1 + S(S^{-1}(a_5 + S(a_2 + k_0 + S(a_3 + k_1))) \\ \quad + a_4 + a_3 + S(k_3))) + S(a_7 + k_0). \end{cases}$$

How to exploit the information: Second case

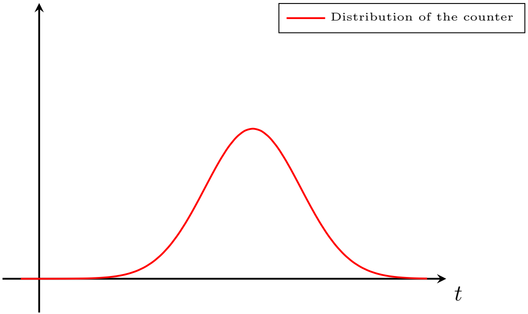
Greedy Substitution :

- Rewrite some of the equations such as to isolate some of the unknowns as function of others;
- **Best case :** guess the remaining unknowns and solve the rest of the equations;
- **Otherwise :** Pre-compute the solutions of the system for the remaining equations.

Example: The a_i are constants depending on the plaintexts/ciphertexts and known key bits.

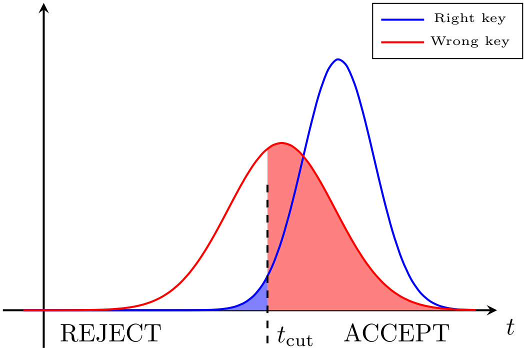
$$\begin{cases} a_1 = S(k_1 + S(a_2 + k_0)) + S(S(a_3 + k_0 + \\ \quad + S(S^{-1}(a_5 + S(a_2 + k_0 + S(a_3 + k_1))))) \\ k_2 = S^{-1}(a_5 + S(a_2 + k_0 + S(a_3 + k_1))) + a_4 \\ k_3 = S^{-1}(S^{-1}(a_6 + S(a_7 + k_0)) + k_1 + a_4 + a_3 \\ \quad + S(S^{-1}(a_5 + S(a_2 + k_0 + S(a_3 + k_1))))). \end{cases}$$

Statistical attack



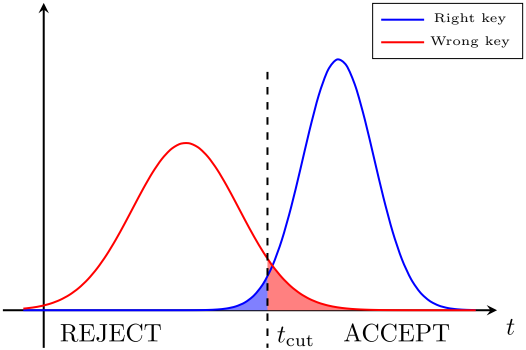
- Increment the counter of a candidate key blackn it appears.
- The counters follow a binomial distribution.

Statistical attack



- Increment the counter of a candidate key blackn it appears.
- The counters follow a binomial distribution.
- Large ratio between the size of $k_{in} \cup k_{out}$ and the number of candidate triplets $\rightsquigarrow$ bigger gap between the distribution of the right key from the wrong keys.

Statistical attack



- Increment the counter of a candidate key blackn it appears.
- The counters follow a binomial distribution.
- Large ratio between the size of $k_{in} \cup k_{out}$ and the number of candidate triplets $\rightsquigarrow$ bigger gap between the distribution of the right key from the wrong keys.

How to exploit the information : Third case

If we need to use a counter.

- Fix part of the plaintext $\rightsquigarrow$ Form disjoint partition of the key

$$X = (P_1 \oplus k_1) \cdot (P_2 \oplus k_2) \oplus k_3$$

	$X = 0$	$X = 1$
$(P_1, P_2) = (0,0)$	$\{(1,0,0),(0,1,0),(0,0,0),(1,1,1)\}$	$\{(1,1,0),(0,1,1),(0,0,1),(1,0,1)\}$
$(P_1, P_2) = (0,1)$	$\{(1,0,1),(0,1,0),(0,0,0),(1,1,0)\}$	$\{(1,0,0),(0,1,1),(0,0,1),(1,1,1)\}$
$(P_1, P_2) = (1,0)$	$\{(0,1,1),(0,0,0),(1,0,0),(1,1,0)\}$	$\{(1,0,1),(1,1,1),(0,0,1),(0,1,1)\}$
$(P_1, P_2) = (1,1)$	$\{(0,1,0),(0,0,1),(1,0,0),(1,1,0)\}$	$\{(1,0,1),(1,1,1),(0,0,0),(0,1,1)\}$

If we fix (P_1, P_2) and X is known then k_3 is uniquely determined by the value of k_1 and k_2.

$\rightsquigarrow$ We can take into account the value of X in the counter.

How to exploit the information : Third case

If we need to use a counter.

- Fix part of the plaintext $\rightsquigarrow$ Form disjoint partition of the key

$$X = (P_1 \oplus k_1) \cdot (P_2 \oplus k_2) \oplus k_3$$

	$X = 0$	$X = 1$
$(P_1, P_2) =$ (0,0)	$\{(1,0,0),(0,1,0),(0,0,0),(1,1,1)\}$	$\{(1,1,0),(0,1,1),(0,0,1),(1,0,1)\}$
$(P_1, P_2) =$(0,1)	$\{(1,0,1),(0,1,0),(0,0,0),(1,1,0)\}$	$\{(1,0,0),(0,1,1),(0,0,1),(1,1,1)\}$
$(P_1, P_2) =$(1,0)	$\{(0,1,1),(0,0,0),(1,0,0),(1,1,0)\}$	$\{(1,0,1),(1,1,1),(0,0,1),(0,1,1)\}$
$(P_1, P_2) =$ (1,1)	$\{(0,1,0),(0,0,1),(1,0,0),(1,1,0)\}$	$\{(1,0,1),(1,1,1),(0,0,0),(0,1,1)\}$

If we fix (P_1, P_2) and X is known then k_3 is uniquely determined by the value of k_1 and k_2.

$\rightsquigarrow$ We can take into account the value of X in the counter.

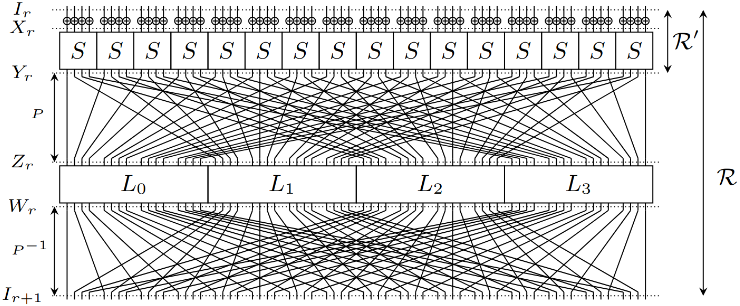
- If we can not fix part of the plaintext/ciphertext $\rightsquigarrow$ the rest of the equations define an over-determined system.

1. Introduction to the state-test technique
2. Hint on how to deal with the state-test equations
3. An application on the block cipher PRIDE

An application

The block cipher PRIDE

PRIDE is a 20-round lightweight block cipher introduced at Crypto 2014 in [ADK+14]. The cipher operate on block of size 64-bit with a 128-bit key $k = k_0||k_1$.
Security claim: Product of Time and Data complexities must be lower than 2^{128}.



Previous best known attack

In [LR18], the authors proposed a differential attack on 18-round PRIDE with the following parameters and properties.

Complexities : Time - Data - Memory

$$\mathscr{T} = 2^{63.3}, \mathscr{D} = 2^{61} \text{ and } \mathscr{M} = 2^{35}.$$

Previous best known attack

In [LR18], the authors proposed a differential attack on 18-round PRIDE with the following parameters and properties.

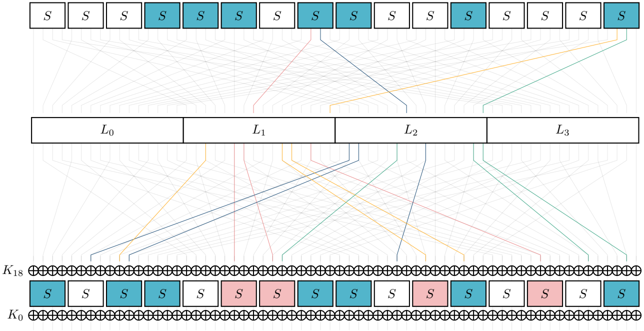
Complexities : Time - Data - Memory

$$\mathscr{T} = 2^{63.3}, \mathscr{D} = 2^{61} \text{ and } \mathscr{M} = 2^{35}.$$

The distinguisher

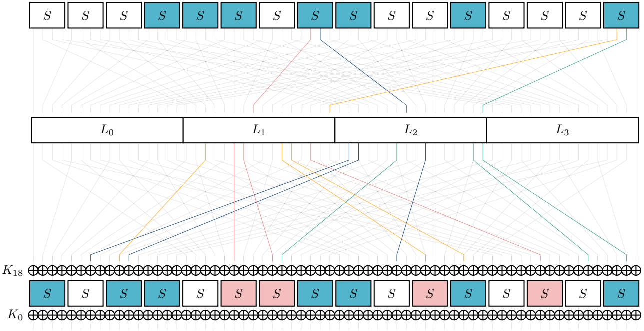
1-round differential characteristic iterated on 14 rounds : 0000 0008 0000 0008 of probability 2^{-56}.

Steps of the attack



- **Original attack:** The differential path involves 76 bits of the key for 2^{74} candidate triplets.
- **Our attack:** The differential path involves 70 bits of information on the key and 67 bits are taken into account in the counter for 2^{66} candidate triplets.

Steps of the attack



- **Original attack:** The differential path involves 76 bits of the key for 2^{74} candidate triplets.
- **Our attack:** The differential path involves 70 bits of information on the key and 67 bits are taken into account in the counter for 2^{66} candidate triplets.

  With 3 remaining bits, check if the over-determined systems of equations are consistent $\rightarrow$ determine 23 bits of the key.

$\rightsquigarrow$ Less key candidates to consider.

Some results

Cipher	Rounds	Time	Memory	Data		Attack	Ref.
CRAFT	21	$2^{106.53}$	2^{100}	$2^{60.99}$		ID	[HSE23]
	21	$2^{116}\dagger$	2^{68}	2^{56}		TD-MITM	[AKM+24]
	22	$2^{125}\dagger$	2^{72}	2^{58}		TD-MITM	[AKM+24]
	23	$2^{125}\dagger$	2^{101}	2^{60}		TD-MITM	[AKM+24]
	23	$2^{111.46}\dagger$	2^{120}	$2^{60.99}$		D	[SYC+24]
	24	$2^{125.79}$	2^{106}	2^{62}		TD-MITM	this work
	25	$2^{125.71}$	2^{106}	2^{64}		TD-MITM	this work
PRIDE	18	$2^{63.3}$	2^{35}	2^{61}		D	[LR17]
	18	$2^{57.83}$	2^{56}	2^{61}		D	this work
Serpent	12	$2^{233.55}$	$2^{127.92}$	$2^{127.92}$	CP	DL	[BCD+22]
	12	$2^{242.93}$	$2^{118.40}$	$2^{118.40}$	CP	DL	[BCD+22]
	12	2^{240}	$2^{118.40}$	$2^{118.40}$	CP	DL	this work

D	Differential	TD-MITM	Truncated Differential MitM
DL	Differential-Linear	ID	Impossible Differential
CP	Chosen Plaintext	KP	Known Plaintext

Table 1: Summary of best known differential attacks on CRAFT, Serpent and PRIDE
†: Time complexity not measured in cipher evaluations

Conclusion

Conclusion

- Generic way to apply an already known technique.
- Can be applied in different families of differential attacks $\rightarrow$ generalization to different context.
- Generic approach to solve big systems of equations.

Open questions and future works

- Implement the technique in a tool.
- Is there a better way to solve the equations ?

Conclusion

Conclusion

- Generic way to apply an already known technique.
- Can be applied in different families of differential attacks $\rightarrow$ generalization to different context.
- Generic approach to solve big systems of equations.

Open questions and future works

- Implement the technique in a tool.
- Is there a better way to solve the equations ?

Thank you for your attention !