



Computer-assisted trail bound proving

Alireza Mehrdad

Digital Security Group, Radboud University, the Netherlands

UL Solutions

August 27, 2024

End-of-ESCADA Workshop

Introduction

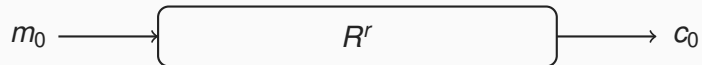
- Is a cryptographic primitive secure?

- Is a cryptographic primitive secure?
 - Cryptanalysis

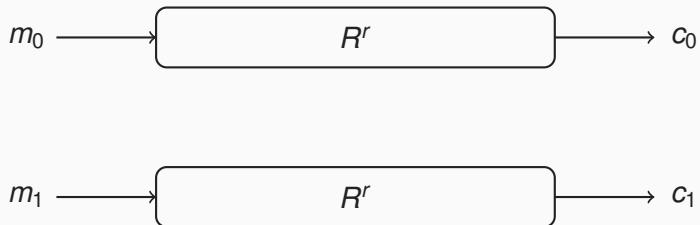
- Is a cryptographic primitive secure?
 - Cryptanalysis
 - Linear

- Is a cryptographic primitive secure?
 - Cryptanalysis
 - Linear
 - Differential

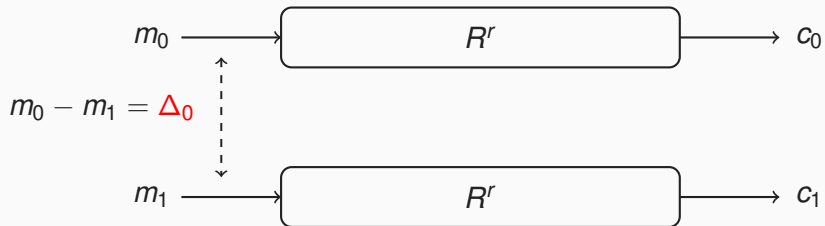
Differential cryptanalysis



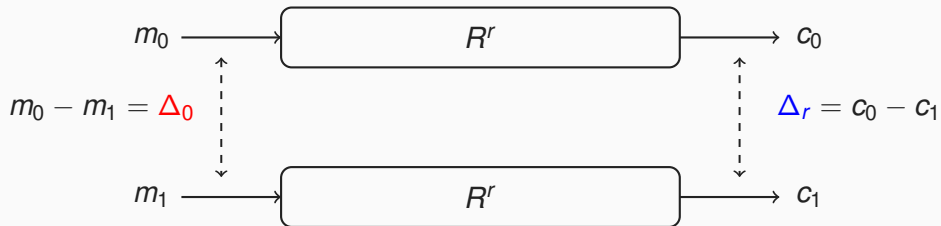
Differential cryptanalysis



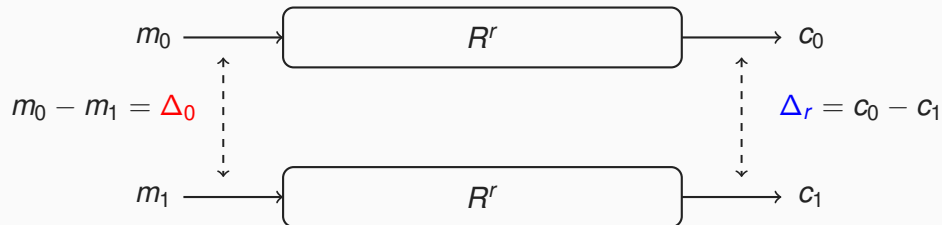
Differential cryptanalysis



Differential cryptanalysis

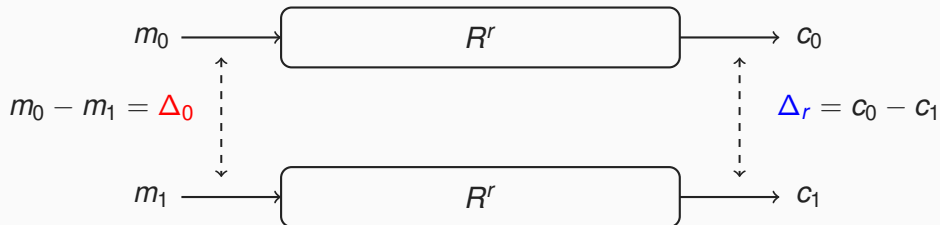


Differential cryptanalysis



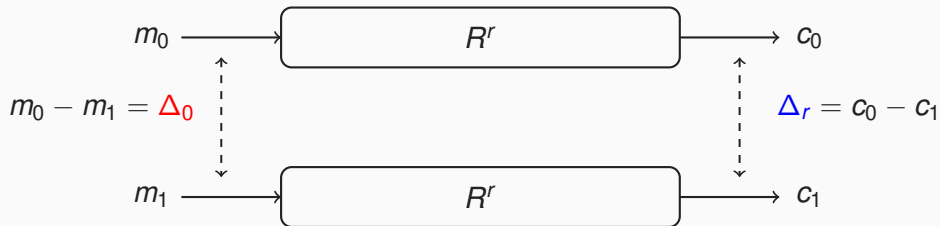
- $(\Delta_0 \rightarrow \Delta_r)$ is a differential over R^r

Differential cryptanalysis



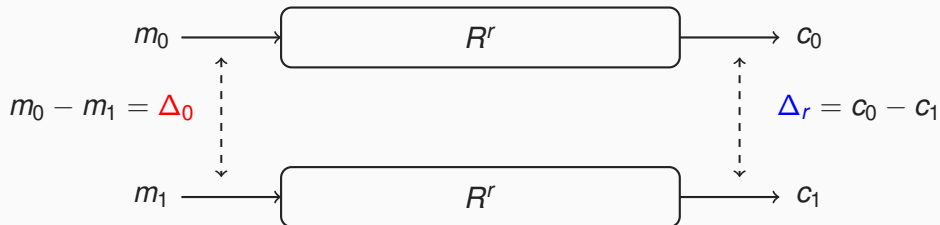
- $(\Delta_0 \rightarrow \Delta_r)$ is a differential over R^r
- $\text{DP}(\Delta_0 \rightarrow \Delta_r) = |\{m_0 \in \mathbb{F}_2^n \mid R^r(m_0) - R^r(m_0 - \Delta_0) = \Delta_r\}| \cdot 2^{-n}$

Differential cryptanalysis



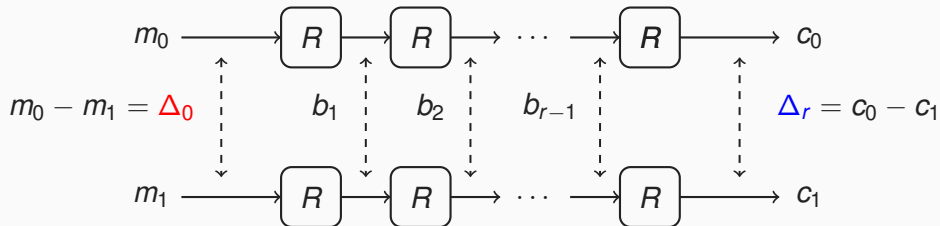
- $(\Delta_0 \rightarrow \Delta_r)$ is a differential over R^r
- $\text{DP}(\Delta_0 \rightarrow \Delta_r) = |\{m_0 \in \mathbb{F}_2^n \mid R^r(m_0) - R^r(m_0 - \Delta_0) = \Delta_r\}| \cdot 2^{-n}$
- Security: $\max \text{DP}(\Delta_0 \rightarrow \Delta_r)$

Differential cryptanalysis



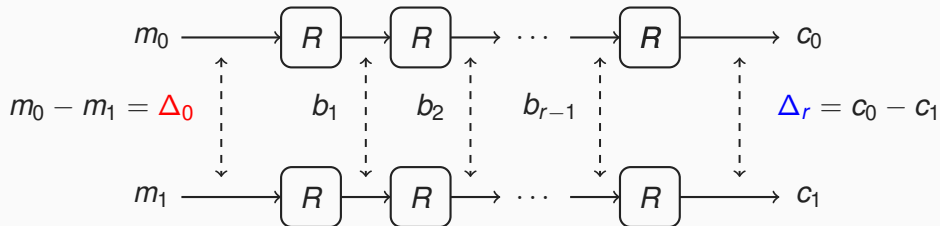
- $(\Delta_0 \rightarrow \Delta_r)$ is a differential over R^r
- $\text{DP}(\Delta_0 \rightarrow \Delta_r) = |\{m_0 \in \mathbb{F}_2^n \mid R^r(m_0) - R^r(m_0 - \Delta_0) = \Delta_r\}| \cdot 2^{-n}$
- Security: $\max \text{DP}(\Delta_0 \rightarrow \Delta_r) \rightarrow$ It is hard to determine

Differential cryptanalysis



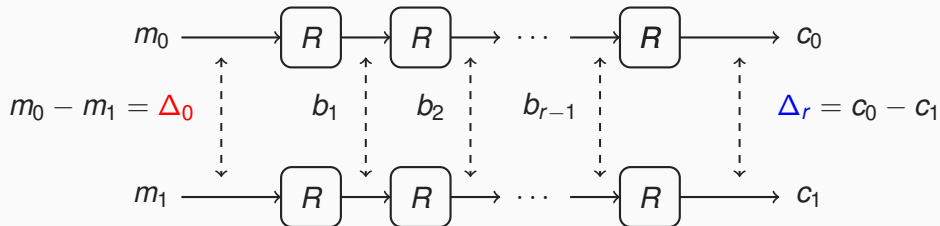
- $(\Delta_0 \rightarrow \Delta_r)$ is a differential over R^r
- $\text{DP}(\Delta_0 \rightarrow \Delta_r) = |\{m_0 \in \mathbb{F}_2^n \mid R^r(m_0) - R^r(m_0 - \Delta_0) = \Delta_r\}| \cdot 2^{-n}$
- Security: $\max \text{DP}(\Delta_0 \rightarrow \Delta_r) \rightarrow$ It is hard to determine

Differential cryptanalysis

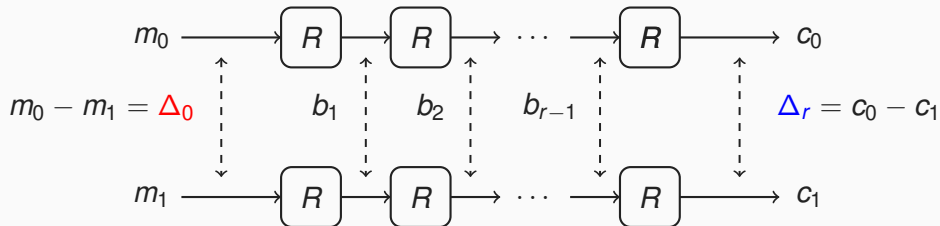


- $(\Delta_0 \rightarrow \Delta_r)$ is a differential over R^r
- $\text{DP}(\Delta_0 \rightarrow \Delta_r) = |\{m_0 \in \mathbb{F}_2^n \mid R^r(m_0) - R^r(m_0 - \Delta_0) = \Delta_r\}| \cdot 2^{-n}$
- Security: $\max \text{DP}(\Delta_0 \rightarrow \Delta_r) \rightarrow$ It is hard to determine
- $Q_r : \Delta_0 \rightarrow b_1 \rightarrow b_2 \rightarrow \dots \rightarrow b_{r-1} \rightarrow \Delta_r$ is a differential trail

Differential cryptanalysis



- $(\Delta_0 \rightarrow \Delta_r)$ is a differential over R^r
- $\text{DP}(\Delta_0 \rightarrow \Delta_r) = |\{m_0 \in \mathbb{F}_2^n \mid R^r(m_0) - R^r(m_0 - \Delta_0) = \Delta_r\}| \cdot 2^{-n}$
- Security: $\max \text{DP}(\Delta_0 \rightarrow \Delta_r) \rightarrow$ It is hard to determine
- $Q_r : \Delta_0 \rightarrow b_1 \rightarrow b_2 \rightarrow \dots \rightarrow b_{r-1} \rightarrow \Delta_r$ is a differential trail
 - $\text{DP}(Q_r) = \text{DP}(\Delta_0 \rightarrow b_1) \times \text{DP}(b_1 \rightarrow b_2) \times \dots \times \text{DP}(b_{r-1} \rightarrow \Delta_r)$



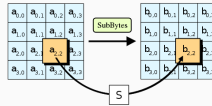
- $(\Delta_0 \rightarrow \Delta_r)$ is a differential over R^r
- $\text{DP}(\Delta_0 \rightarrow \Delta_r) = |\{m_0 \in \mathbb{F}_2^n \mid R^r(m_0) - R^r(m_0 - \Delta_0) = \Delta_r\}| \cdot 2^{-n}$
- Security: $\max \text{DP}(\Delta_0 \rightarrow \Delta_r) \rightarrow$ It is hard to determine
- $Q_r : \Delta_0 \rightarrow b_1 \rightarrow b_2 \rightarrow \dots \rightarrow b_{r-1} \rightarrow \Delta_r$ is a differential trail
 - $\text{DP}(Q_r) = \text{DP}(\Delta_0 \rightarrow b_1) \times \text{DP}(b_1 \rightarrow b_2) \times \dots \times \text{DP}(b_{r-1} \rightarrow \Delta_r)$
 - $\text{DP}(\Delta_0 \rightarrow \Delta_r) \approx \text{DP}(Q_r)$

- Aligned

- Aligned $\rightarrow$ AES

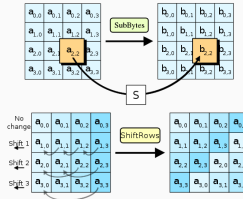
Aligned VS unaligned

- Aligned $\rightarrow$ AES



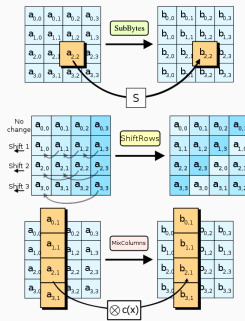
Aligned VS unaligned

- Aligned $\rightarrow$ AES



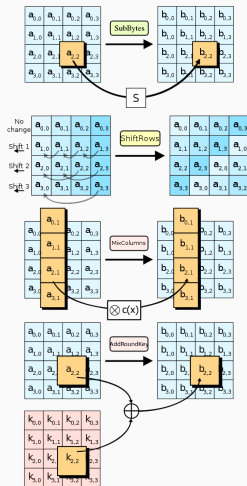
Aligned VS unaligned

- Aligned $\rightarrow$ AES



Aligned VS unaligned

- Aligned $\rightarrow$ AES



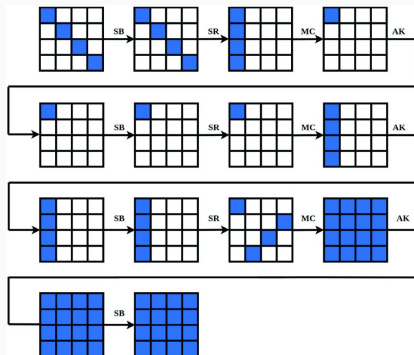
- The truncated differences can be easily constrained and followed

- The truncated differences can be easily constrained and followed
 - Compact and powerful proofs

- The truncated differences can be easily constrained and followed
 - Compact and powerful proofs $\Rightarrow$ AES

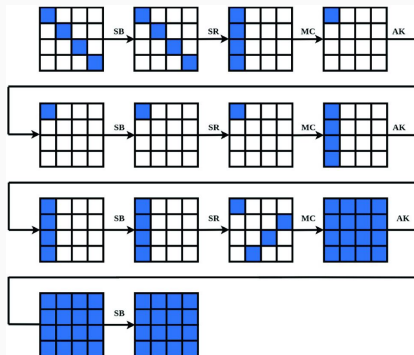
Truncated differential

- The truncated differences can be easily constrained and followed
 - Compact and powerful proofs $\Rightarrow$ AES



Truncated differential

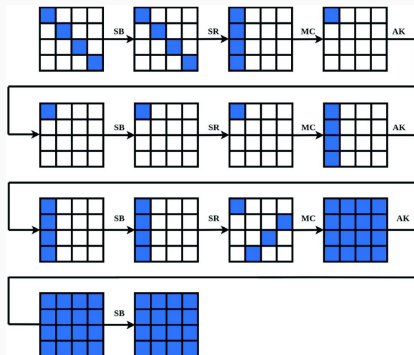
- The truncated differences can be easily constrained and followed
 - Compact and powerful proofs $\Rightarrow$ AES



- A differential trail over 4 consecutive rounds has at least 25 active S-boxes

Truncated differential

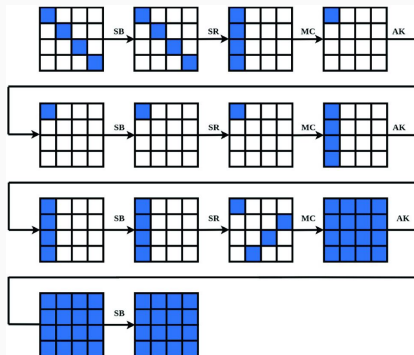
- The truncated differences can be easily constrained and followed
 - Compact and powerful proofs $\Rightarrow$ AES



- A differential trail over 4 consecutive rounds has at least 25 active S-boxes
- Maximum probability of an S-box is 2^{-6}

Truncated differential

- The truncated differences can be easily constrained and followed
 - Compact and powerful proofs $\Rightarrow$ AES



- A differential trail over 4 consecutive rounds has at least 25 active S-boxes
- Maximum probability of an S-box is $2^{-6} \Rightarrow$ DP is $(2^{-6})^{25} = 2^{-150}$

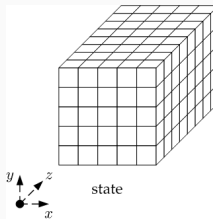
- Unaligned

- Unaligned $\rightarrow$ KECCAK

- Unaligned $\rightarrow$ KECCAK
- $R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$

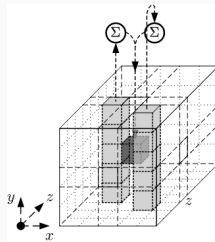
Aligned VS unaligned

- Unaligned $\rightarrow$ KECCAK
- $R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$



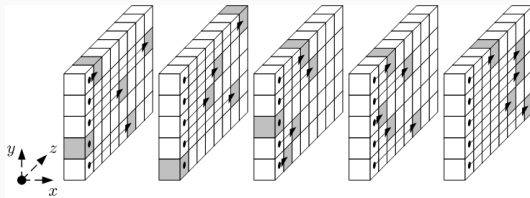
Aligned VS unaligned

- Unaligned $\rightarrow$ KECCAK
- $R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$



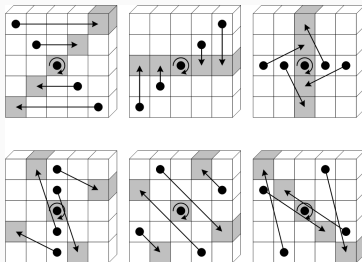
Aligned VS unaligned

- Unaligned $\rightarrow$ KECCAK
- $R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$



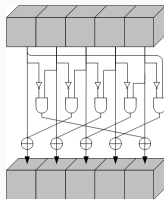
Aligned VS unaligned

- Unaligned $\rightarrow$ KECCAK
- $R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$



Aligned VS unaligned

- Unaligned $\rightarrow$ KECCAK
- $R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$



- KECCAK

Unaligned (weak alignment)

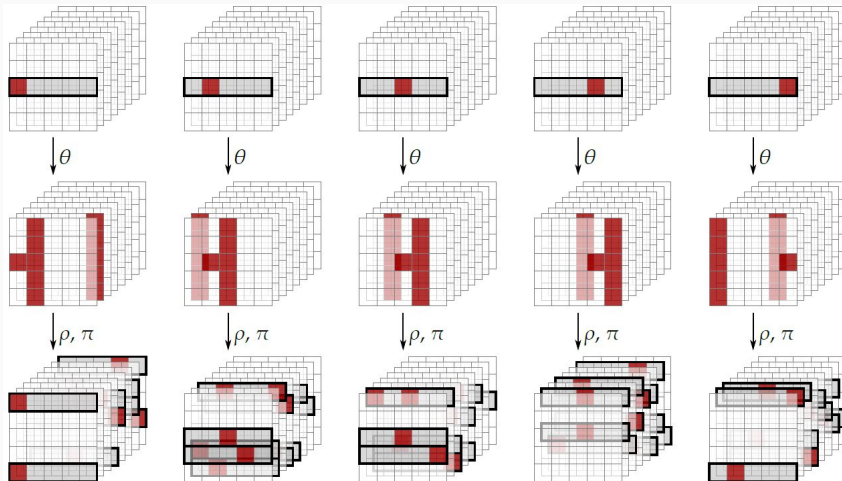
- KECCAK $\rightarrow$ hard to subdivide the input

Unaligned (weak alignment)

- KECCAK $\rightarrow$ hard to subdivide the input $\rightarrow$ differences spread extremely fast

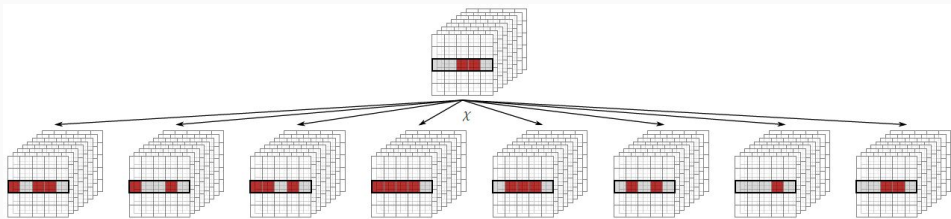
Unaligned (weak alignment)

- KECCAK $\rightarrow$ hard to subdivide the input $\rightarrow$ differences spread extremely fast



Unaligned (weak alignment)

- KECCAK $\rightarrow$ hard to subdivide the input $\rightarrow$ differences spread extremely fast



Quick summary of the introduction section

- Aligned VS unaligned

Quick summary of the introduction section

- Aligned VS unaligned
 - Aligned: a group of bits

Quick summary of the introduction section

- Aligned VS unaligned
 - Aligned: a group of bits $\rightarrow$ truncated differential

Quick summary of the introduction section

- Aligned VS unaligned
 - Aligned: a group of bits $\rightarrow$ truncated differential $\rightarrow$ compact and powerful

Quick summary of the introduction section

- Aligned VS unaligned
 - Aligned: a group of bits $\rightarrow$ truncated differential $\rightarrow$ compact and powerful
 - Unaligned: single bits

Quick summary of the introduction section

- Aligned VS unaligned
 - Aligned: a group of bits $\rightarrow$ truncated differential $\rightarrow$ compact and powerful
 - Unaligned: single bits $\rightarrow$ truncated differential is not applicable

Quick summary of the introduction section

- Aligned VS unaligned
 - Aligned: a group of bits $\rightarrow$ truncated differential $\rightarrow$ compact and powerful
 - Unaligned: single bits $\rightarrow$ truncated differential is not applicable $\rightarrow$ what can we do for unaligned?

Quick summary of the introduction section

- Aligned VS unaligned
 - Aligned: a group of bits $\rightarrow$ truncated differential $\rightarrow$ compact and powerful
 - Unaligned: single bits $\rightarrow$ truncated differential is not applicable $\rightarrow$ what can we do for unaligned?
 - Computer-assisted proofs and dedicated tools

Quick summary of the introduction section

- Aligned VS unaligned
 - Aligned: a group of bits $\rightarrow$ truncated differential $\rightarrow$ compact and powerful
 - Unaligned: single bits $\rightarrow$ truncated differential is not applicable $\rightarrow$ what can we do for unaligned?
 - Computer-assisted proofs and dedicated tools
 - Goal: computing maximum DP over r -round

Quick summary of the introduction section

- Aligned VS unaligned
 - Aligned: a group of bits $\rightarrow$ truncated differential $\rightarrow$ compact and powerful
 - Unaligned: single bits $\rightarrow$ truncated differential is not applicable $\rightarrow$ what can we do for unaligned?
 - Computer-assisted proofs and dedicated tools
 - Goal: computing maximum DP over r -round
- Weight: $w(Q) = -\log_2(\text{DP})$

Quick summary of the introduction section

- Aligned VS unaligned
 - Aligned: a group of bits $\rightarrow$ truncated differential $\rightarrow$ compact and powerful
 - Unaligned: single bits $\rightarrow$ truncated differential is not applicable $\rightarrow$ what can we do for unaligned?
 - Computer-assisted proofs and dedicated tools
 - Goal: computing maximum DP over r -round
- Weight: $w(Q) = -\log_2(\text{DP})$
- $\text{DP} = 2^{-3} \times 2^{-4} \times 2^{-5} \times 2^{-2}$

Quick summary of the introduction section

- Aligned VS unaligned
 - Aligned: a group of bits $\rightarrow$ truncated differential $\rightarrow$ compact and powerful
 - Unaligned: single bits $\rightarrow$ truncated differential is not applicable $\rightarrow$ what can we do for unaligned?
 - Computer-assisted proofs and dedicated tools
 - Goal: computing maximum DP over r -round
- Weight: $w(Q) = -\log_2(\text{DP})$
- $\text{DP} = 2^{-3} \times 2^{-4} \times 2^{-5} \times 2^{-2} = 2^{-(3+4+5+2)}$

Quick summary of the introduction section

- Aligned VS unaligned
 - Aligned: a group of bits $\rightarrow$ truncated differential $\rightarrow$ compact and powerful
 - Unaligned: single bits $\rightarrow$ truncated differential is not applicable $\rightarrow$ what can we do for unaligned?
 - Computer-assisted proofs and dedicated tools
 - Goal: computing maximum DP over r -round
- Weight: $w(Q) = -\log_2(\text{DP})$
- $\text{DP} = 2^{-3} \times 2^{-4} \times 2^{-5} \times 2^{-2} = 2^{-(3+4+5+2)} \rightarrow w = 3 + 4 + 5 + 2$

Quick summary of the introduction section

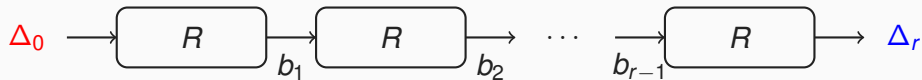
- Aligned VS unaligned
 - Aligned: a group of bits $\rightarrow$ truncated differential $\rightarrow$ compact and powerful
 - Unaligned: single bits $\rightarrow$ truncated differential is not applicable $\rightarrow$ what can we do for unaligned?
 - Computer-assisted proofs and dedicated tools
 - Goal: computing maximum DP over r -round
- Weight: $w(Q) = -\log_2(\text{DP})$
- $\text{DP} = 2^{-3} \times 2^{-4} \times 2^{-5} \times 2^{-2} = 2^{-(3+4+5+2)} \rightarrow w = 3 + 4 + 5 + 2$
- Goal: computing minimum weight over r -round of an unaligned primitive

Quick summary of the introduction section

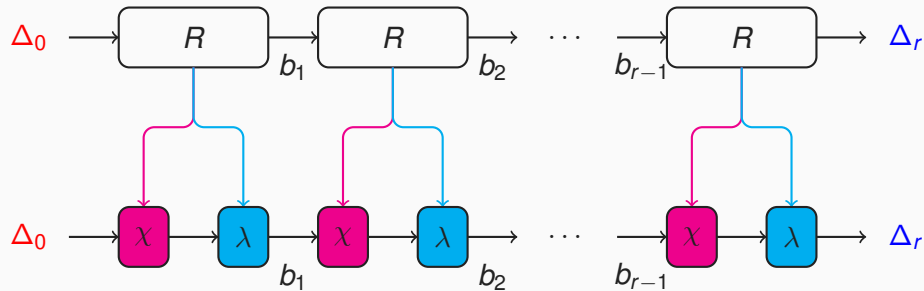
- Aligned VS unaligned
 - Aligned: a group of bits $\rightarrow$ truncated differential $\rightarrow$ compact and powerful
 - Unaligned: single bits $\rightarrow$ truncated differential is not applicable $\rightarrow$ what can we do for unaligned?
 - Computer-assisted proofs and dedicated tools
 - Goal: computing maximum DP over r -round
- Weight: $w(Q) = -\log_2(\text{DP})$
- $\text{DP} = 2^{-3} \times 2^{-4} \times 2^{-5} \times 2^{-2} = 2^{-(3+4+5+2)} \rightarrow w = 3 + 4 + 5 + 2$
- Goal: computing minimum weight over r -round of an unaligned primitive
- Generate r -round differential trails

Generating r -round Differential trails

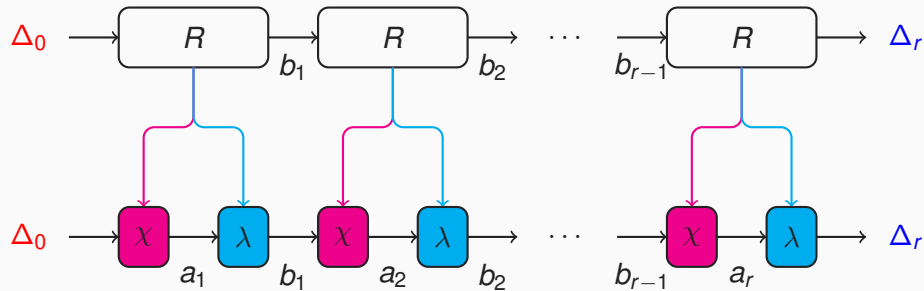
r-round differential trail



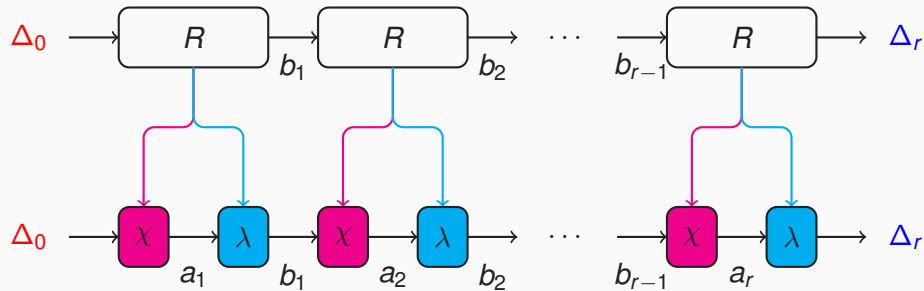
r-round differential trail



r-round differential trail

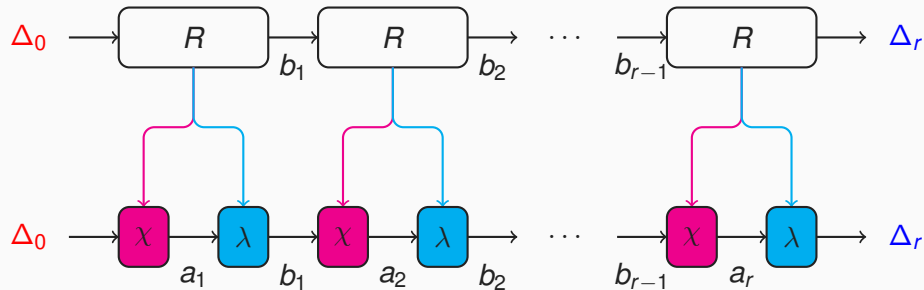


r-round differential trail



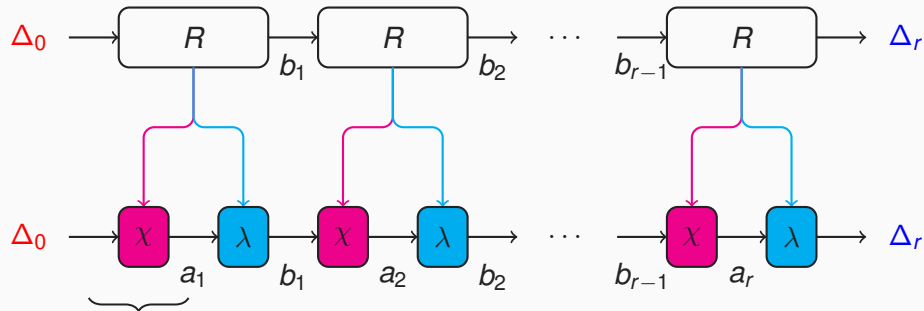
$$\text{DP}(a_i \xrightarrow{\lambda} b_i) = 1$$

r-round differential trail



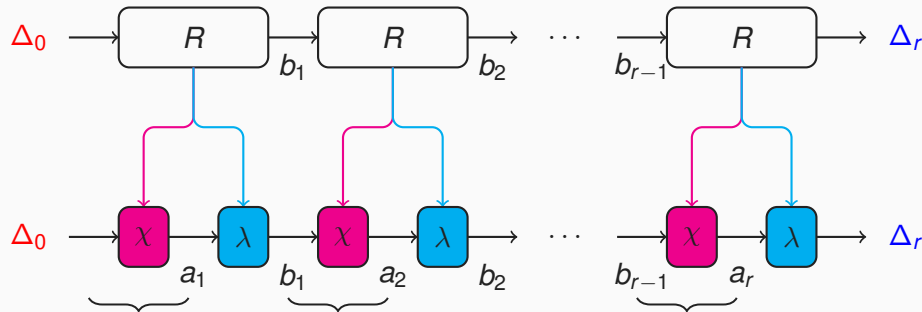
$$\text{DP}(a_i \xrightarrow{\lambda} b_i) = 1 \Rightarrow w(a_i \xrightarrow{\lambda} b_i) = 0$$

r-round differential trail



$$w(Q_r) = w(\Delta_0 \rightarrow a_1) +$$

r-round differential trail



$$w(Q_r) = w(\Delta_0 \rightarrow a_1) + \sum_{i=1}^{r-1} w(b_i \rightarrow a_{i+1})$$

Compute the minimum weight of r -round (maximum DP)

- Goal: the minimum weight of r -round trails

Compute the minimum weight of r -round (maximum DP)

- Goal: the minimum weight of r -round trails
 - We generate all r -round trails up to weight T_r

Compute the minimum weight of r-round (maximum DP)

- Goal: the minimum weight of r-round trails
 - We generate all r-round trails up to weight T_r
 - If there is any $\rightarrow$ then we found the minimum

Compute the minimum weight of r -round (maximum DP)

- Goal: the minimum weight of r -round trails
 - We generate all r -round trails up to weight T_r
 - If there is any $\rightarrow$ then we found the minimum
 - If there is nothing $\rightarrow T_r + 1$ is a lower bound on the weight

Compute the minimum weight of r -round (maximum DP)

- Goal: the minimum weight of r -round trails
 - We generate all r -round trails up to weight T_r
 - If there is any $\rightarrow$ then we found the minimum
 - If there is nothing $\rightarrow T_r + 1$ is a lower bound on the weight
 - Hard to start with r -round trails

Compute the minimum weight of r-round (maximum DP)

- Goal: the minimum weight of r-round trails
 - We generate all r-round trails up to weight T_r
 - If there is any $\rightarrow$ then we found the minimum
 - If there is nothing $\rightarrow T_r + 1$ is a lower bound on the weight
 - Hard to start with r-round trails
 - Start with smaller trails

Compute the minimum weight of r -round (maximum DP)

- Goal: the minimum weight of r -round trails
 - We generate all r -round trails up to weight T_r
 - If there is any $\rightarrow$ then we found the minimum
 - If there is nothing $\rightarrow T_r + 1$ is a lower bound on the weight
 - Hard to start with r -round trails
 - Start with smaller trails
 - For instance $(r/2)$ -round trails up to weight $T_r/2$

Compute the minimum weight of r -round (maximum DP)

- Goal: the minimum weight of r -round trails
 - We generate all r -round trails up to weight T_r
 - If there is any $\rightarrow$ then we found the minimum
 - If there is nothing $\rightarrow T_r + 1$ is a lower bound on the weight
 - Hard to start with r -round trails
 - Start with smaller trails
 - For instance $(r/2)$ -round trails up to weight $T_r/2$
 - If not possible, start with $(r/4)$ -round trails and so on

Compute the minimum weight of r -round (maximum DP)

- Goal: the minimum weight of r -round trails
 - We generate all r -round trails up to weight T_r
 - If there is any $\rightarrow$ then we found the minimum
 - If there is nothing $\rightarrow T_r + 1$ is a lower bound on the weight
 - Hard to start with r -round trails
 - Start with smaller trails
 - For instance $(r/2)$ -round trails up to weight $T_r/2$
 - If not possible, start with $(r/4)$ -round trails and so on
- Shall we start with 1-round trails?

Compute the minimum weight of r -round (maximum DP)

- Goal: the minimum weight of r -round trails
 - We generate all r -round trails up to weight T_r
 - If there is any $\rightarrow$ then we found the minimum
 - If there is nothing $\rightarrow T_r + 1$ is a lower bound on the weight
 - Hard to start with r -round trails
 - Start with smaller trails
 - For instance $(r/2)$ -round trails up to weight $T_r/2$
 - If not possible, start with $(r/4)$ -round trails and so on
- Shall we start with 1-round trails? NO

Compute the minimum weight of r -round (maximum DP)

- Goal: the minimum weight of r -round trails
 - We generate all r -round trails up to weight T_r
 - If there is any $\rightarrow$ then we found the minimum
 - If there is nothing $\rightarrow T_r + 1$ is a lower bound on the weight
 - Hard to start with r -round trails
 - Start with smaller trails
 - For instance $(r/2)$ -round trails up to weight $T_r/2$
 - If not possible, start with $(r/4)$ -round trails and so on
- Shall we start with 1-round trails? NO
 - $\#(2\text{-round trails up to weight } T_2) \ll \#(1\text{-round trails up to weight } T_2/2)$

Compute the minimum weight of r -round (maximum DP)

- Goal: the minimum weight of r -round trails
 - We generate all r -round trails up to weight T_r
 - If there is any $\rightarrow$ then we found the minimum
 - If there is nothing $\rightarrow T_r + 1$ is a lower bound on the weight
 - Hard to start with r -round trails
 - Start with smaller trails
 - For instance $(r/2)$ -round trails up to weight $T_r/2$
 - If not possible, start with $(r/4)$ -round trails and so on
- Shall we start with 1-round trails? NO
 - $\#(2\text{-round trails up to weight } T_2) \ll \#(1\text{-round trails up to weight } T_2/2)$
 - With almost equal effort, you can produce 2-round trails

Compute the minimum weight of r -round (maximum DP)

- Goal: the minimum weight of r -round trails
 - We generate all r -round trails up to weight T_r
 - If there is any $\rightarrow$ then we found the minimum
 - If there is nothing $\rightarrow T_r + 1$ is a lower bound on the weight
 - Hard to start with r -round trails
 - Start with smaller trails
 - For instance $(r/2)$ -round trails up to weight $T_r/2$
 - If not possible, start with $(r/4)$ -round trails and so on
- Shall we start with 1-round trails? NO
 - $\#(2\text{-round trails up to weight } T_2) \ll \#(1\text{-round trails up to weight } T_2/2)$
 - With almost equal effort, you can produce 2-round trails
 - Why?

2-round trails and trail cores in an unaligned primitive

- 2-round trails: $b_i \xrightarrow{r_1} b_{i+1} \xrightarrow{r_2} b_{i+2}$

2-round trails and trail cores in an unaligned primitive

- 2-round trails: $b_i \xrightarrow{r_1} b_{i+1} \xrightarrow{r_2} b_{i+2}$
- $b_i \xrightarrow{\chi} a_{i+1} \xrightarrow{\lambda} b_{i+1} \xrightarrow{\chi} a_{i+2} \xrightarrow{\lambda} b_{i+2}$

2-round trails and trail cores in an unaligned primitive

- 2-round trails: $b_i \xrightarrow{r_1} b_{i+1} \xrightarrow{r_2} b_{i+2}$
- $b_i \xrightarrow{\chi} a_{i+1} \xrightarrow{\lambda} b_{i+1} \xrightarrow{\chi} a_{i+2} \xrightarrow{\lambda} b_{i+2}$
- Since χ has degree 2, the weight is determined by just its input

2-round trails and trail cores in an unaligned primitive

- 2-round trails: $b_i \xrightarrow{r_1} b_{i+1} \xrightarrow{r_2} b_{i+2}$
- $b_i \xrightarrow{\chi} a_{i+1} \xrightarrow{\lambda} b_{i+1} \xrightarrow{\chi} a_{i+2} \xrightarrow{\lambda} b_{i+2}$
- Since χ has degree 2, the weight is determined by just its input
 - $w(Q_2) = w(b_i \xrightarrow{\chi} a_{i+1}) + w(b_{i+1} \xrightarrow{\chi} a_{i+2})$

2-round trails and trail cores in an unaligned primitive

- 2-round trails: $b_i \xrightarrow{r_1} b_{i+1} \xrightarrow{r_2} b_{i+2}$
- $b_i \xrightarrow{\chi} a_{i+1} \xrightarrow{\lambda} b_{i+1} \xrightarrow{\chi} a_{i+2} \xrightarrow{\lambda} b_{i+2}$
- Since χ has degree 2, the weight is determined by just its input
 - $w(Q_2) = w(b_i \xrightarrow{\chi} a_{i+1}) + w(b_{i+1} \xrightarrow{\chi} a_{i+2}) = w(b_i) + w(b_{i+1})$

2-round trails and trail cores in an unaligned primitive

- 2-round trails: $b_i \xrightarrow{r_1} b_{i+1} \xrightarrow{r_2} b_{i+2}$
- $b_i \xrightarrow{\chi} a_{i+1} \xrightarrow{\lambda} b_{i+1} \xrightarrow{\chi} a_{i+2} \xrightarrow{\lambda} b_{i+2}$
- Since χ has degree 2, the weight is determined by just its input
 - $w(Q_2) = w(b_i \xrightarrow{\chi} a_{i+1}) + w(b_{i+1} \xrightarrow{\chi} a_{i+2}) = w(b_i) + w(b_{i+1})$
- Instead of whole 2-rounds up to a certain weight, we generate the following

2-round trails and trail cores in an unaligned primitive

- 2-round trails: $b_i \xrightarrow{r_1} b_{i+1} \xrightarrow{r_2} b_{i+2}$
- $b_i \xrightarrow{\chi} a_{i+1} \xrightarrow{\lambda} b_{i+1} \xrightarrow{\chi} a_{i+2} \xrightarrow{\lambda} b_{i+2}$
- Since χ has degree 2, the weight is determined by just its input
 - $w(Q_2) = w(b_i \xrightarrow{\chi} a_{i+1}) + w(b_{i+1} \xrightarrow{\chi} a_{i+2}) = w(b_i) + w(b_{i+1})$
- Instead of whole 2-rounds up to a certain weight, we generate the following
 - $b_i \xrightarrow{\chi} a_{i+1} \xrightarrow{\lambda} b_{i+1} \xrightarrow{\chi}$

2-round trails and trail cores in an unaligned primitive

- 2-round trails: $b_i \xrightarrow{r_1} b_{i+1} \xrightarrow{r_2} b_{i+2}$
- $b_i \xrightarrow{\chi} a_{i+1} \xrightarrow{\lambda} b_{i+1} \xrightarrow{\chi} a_{i+2} \xrightarrow{\lambda} b_{i+2}$
- Since χ has degree 2, the weight is determined by just its input
 - $w(Q_2) = w(b_i \xrightarrow{\chi} a_{i+1}) + w(b_{i+1} \xrightarrow{\chi} a_{i+2}) = w(b_i) + w(b_{i+1})$
- Instead of whole 2-rounds up to a certain weight, we generate the following
 - $b_i \xrightarrow{\chi} a_{i+1} \xrightarrow{\lambda} b_{i+1} \xrightarrow{\chi}$
- If we can compute the weight of a_{i+1} 's, we can ignore b_i too

2-round trails and trail cores in an unaligned primitive

- 2-round trails: $b_i \xrightarrow{r_1} b_{i+1} \xrightarrow{r_2} b_{i+2}$
- $b_i \xrightarrow{\chi} a_{i+1} \xrightarrow{\lambda} b_{i+1} \xrightarrow{\chi} a_{i+2} \xrightarrow{\lambda} b_{i+2}$
- Since χ has degree 2, the weight is determined by just its input
 - $w(Q_2) = w(b_i \xrightarrow{\chi} a_{i+1}) + w(b_{i+1} \xrightarrow{\chi} a_{i+2}) = w(b_i) + w(b_{i+1})$
- Instead of whole 2-rounds up to a certain weight, we generate the following
 - $b_i \xrightarrow{\chi} a_{i+1} \xrightarrow{\lambda} b_{i+1} \xrightarrow{\chi}$
- If we can compute the weight of a_{i+1} 's, we can ignore b_i too
 - $\min w^{-1}(a_{i+1}) \leq w(b_i)$

2-round trails and trail cores in an unaligned primitive

- 2-round trails: $b_i \xrightarrow{r_1} b_{i+1} \xrightarrow{r_2} b_{i+2}$
- $b_i \xrightarrow{\chi} a_{i+1} \xrightarrow{\lambda} b_{i+1} \xrightarrow{\chi} a_{i+2} \xrightarrow{\lambda} b_{i+2}$
- Since χ has degree 2, the weight is determined by just its input
 - $w(Q_2) = w(b_i \xrightarrow{\chi} a_{i+1}) + w(b_{i+1} \xrightarrow{\chi} a_{i+2}) = w(b_i) + w(b_{i+1})$
- Instead of whole 2-rounds up to a certain weight, we generate the following
 - $b_i \xrightarrow{\chi} a_{i+1} \xrightarrow{\lambda} b_{i+1} \xrightarrow{\chi}$
- If we can compute the weight of a_{i+1} 's, we can ignore b_i too
 - $\min w^{-1}(a_{i+1}) \leq w(b_i)$
 - $w(Q_2) \geq \min w^{-1}(a_{i+1}) + w(b_{i+1})$

2-round trails and trail cores in an unaligned primitive

- 2-round trails: $b_i \xrightarrow{r_1} b_{i+1} \xrightarrow{r_2} b_{i+2}$
- $b_i \xrightarrow{\chi} a_{i+1} \xrightarrow{\lambda} b_{i+1} \xrightarrow{\chi} a_{i+2} \xrightarrow{\lambda} b_{i+2}$
- Since χ has degree 2, the weight is determined by just its input
 - $w(Q_2) = w(b_i \xrightarrow{\chi} a_{i+1}) + w(b_{i+1} \xrightarrow{\chi} a_{i+2}) = w(b_i) + w(b_{i+1})$
- Instead of whole 2-rounds up to a certain weight, we generate the following
 - $b_i \xrightarrow{\chi} a_{i+1} \xrightarrow{\lambda} b_{i+1} \xrightarrow{\chi}$
- If we can compute the weight of a_{i+1} 's, we can ignore b_i too
 - $\min w^{-1}(a_{i+1}) \leq w(b_i)$
 - $w(Q_2) \geq \min w^{-1}(a_{i+1}) + w(b_{i+1})$
- 2-round trail **core**: $\xrightarrow{\chi} a_{i+1} \xrightarrow{\lambda} b_{i+1} \xrightarrow{\chi}$

Generate 2-round trail cores up to weight T_2

$$\xrightarrow{x} \textcolor{red}{a}_1 \xrightarrow{\lambda} \textcolor{blue}{b}_1 \xrightarrow{x}$$

- $w(Q_2) \geq \min w^{-1}(\textcolor{red}{a}_1) + w(\textcolor{blue}{b}_1)$

Generate 2-round trail cores up to weight T_2

$$\xrightarrow{x} a_1 \xrightarrow{\lambda} b_1 \xrightarrow{x}$$

- $w(Q_2) \geq \min w^{-1}(a_1) + w(b_1)$
- Generate all differences at a_1

Generate 2-round trail cores up to weight T_2

$$\xrightarrow{x} a_1 \xrightarrow{\lambda} b_1 \xrightarrow{x}$$

- $w(Q_2) \geq \min w^{-1}(a_1) + w(b_1)$
- Generate all differences at a_1
- Compute b_1

Generate 2-round trail cores up to weight T_2

$$\xrightarrow{x} a_1 \xrightarrow{\lambda} b_1 \xrightarrow{x}$$

- $w(Q_2) \geq \min w^{-1}(a_1) + w(b_1)$
- Generate all differences at a_1
- Compute $b_1 = \lambda(a_1)$

Generate 2-round trail cores up to weight T_2

$$\xrightarrow{x} a_1 \xrightarrow{\lambda} b_1 \xrightarrow{x}$$

- $w(Q_2) \geq \min w^{-1}(a_1) + w(b_1)$
- Generate all differences at a_1
- Compute $b_1 = \lambda(a_1)$
- If its weight $\leq T_2$ then (a_1, b_1) is a valid pair

Generate 2-round trail cores up to weight T_2

$$\xrightarrow{x} a_1 \xrightarrow{\lambda} b_1 \xrightarrow{x}$$

- $w(Q_2) \geq \min w^{-1}(a_1) + w(b_1)$
- Generate all differences at a_1
- Compute $b_1 = \lambda(a_1)$
- If its weight $\leq T_2$ then (a_1, b_1) is a valid pair
- With the similar effort as building 1-round trails, we can build 2-round trail cores

- Goal: computing minimum weight over r -round of an unaligned primitive

Quick summary

- Goal: computing minimum weight over r -round of an unaligned primitive
- Starting with r -round trails is not efficient

Quick summary

- Goal: computing minimum weight over r -round of an unaligned primitive
- Starting with r -round trails is not efficient
- Start with generating all 2-round trail cores

Quick summary

- Goal: computing minimum weight over r -round of an unaligned primitive
- Starting with r -round trails is not efficient
- Start with generating all 2-round trail cores
 - Their number is much less than 1-round trails

Quick summary

- Goal: computing minimum weight over r -round of an unaligned primitive
- Starting with r -round trails is not efficient
- Start with generating all 2-round trail cores
 - Their number is much less than 1-round trails
 - Similar effort

- Goal: computing minimum weight over r -round of an unaligned primitive
- Starting with r -round trails is not efficient
- Start with generating all 2-round trail cores
 - Their number is much less than 1-round trails
 - Similar effort
- How to scan the space of all 2-round trail cores up to T_2 ?

- Goal: computing minimum weight over r -round of an unaligned primitive
- Starting with r -round trails is not efficient
- Start with generating all 2-round trail cores
 - Their number is much less than 1-round trails
 - Similar effort
- How to scan the space of all 2-round trail cores up to T_2 ?
 - Tree search approach [Mella, Daemen, Van Assche, ToSC 2016]

$$\xrightarrow{x} a_1 \xrightarrow{\lambda} b_1 \xrightarrow{x}$$

- Unit: a group of active bits at a_1 and its corresponding active bits at b_1

$$\xrightarrow{x} a_1 \xrightarrow{\lambda} b_1 \xrightarrow{x}$$

- Unit: a group of active bits at a_1 and its corresponding active bits at b_1
- The way to define these groups depends on the properties of the cipher

$$\xrightarrow{x} a_1 \xrightarrow{\lambda} b_1 \xrightarrow{x}$$

- Unit: a group of active bits at a_1 and its corresponding active bits at b_1
- The way to define these groups depends on the properties of the cipher
- Unit-list: an ordered list of units

$$\xrightarrow{x} a_1 \xrightarrow{\lambda} b_1 \xrightarrow{x}$$

- Unit: a group of active bits at a_1 and its corresponding active bits at b_1
- The way to define these groups depends on the properties of the cipher
- Unit-list: an ordered list of units
- We form a tree

$$\xrightarrow{x} a_1 \xrightarrow{\lambda} b_1 \xrightarrow{x}$$

- Unit: a group of active bits at a_1 and its corresponding active bits at b_1
- The way to define these groups depends on the properties of the cipher
- Unit-list: an ordered list of units
- We form a tree
 - Each node of the tree represents a unit-list

$$\xrightarrow{x} a_1 \xrightarrow{\lambda} b_1 \xrightarrow{x}$$

- Unit: a group of active bits at a_1 and its corresponding active bits at b_1
- The way to define these groups depends on the properties of the cipher
- Unit-list: an ordered list of units
- We form a tree
 - Each node of the tree represents a unit-list
 - Three function to form a tree

$$\xrightarrow{x} a_1 \xrightarrow{\lambda} b_1 \xrightarrow{x}$$

- Unit: a group of active bits at a_1 and its corresponding active bits at b_1
- The way to define these groups depends on the properties of the cipher
- Unit-list: an ordered list of units
- We form a tree
 - Each node of the tree represents a unit-list
 - Three function to form a tree
 - toChild(): adds a new unit to the unit-list

$$\xrightarrow{x} a_1 \xrightarrow{\lambda} b_1 \xrightarrow{x}$$

- Unit: a group of active bits at a_1 and its corresponding active bits at b_1
- The way to define these groups depends on the properties of the cipher
- Unit-list: an ordered list of units
- We form a tree
 - Each node of the tree represents a unit-list
 - Three function to form a tree
 - toChild(): adds a new unit to the unit-list
 - toSibling(): Changes the last unit

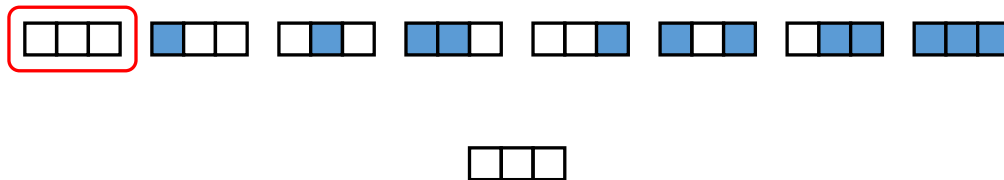
$$\xrightarrow{x} a_1 \xrightarrow{\lambda} b_1 \xrightarrow{x}$$

- Unit: a group of active bits at a_1 and its corresponding active bits at b_1
- The way to define these groups depends on the properties of the cipher
- Unit-list: an ordered list of units
- We form a tree
 - Each node of the tree represents a unit-list
 - Three function to form a tree
 - toChild(): adds a new unit to the unit-list
 - toSibling(): Changes the last unit
 - toParent(): deletes the last unit

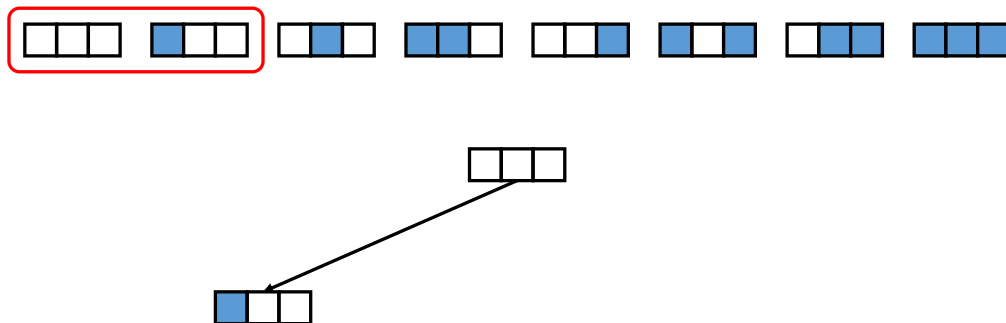
How to build a simple tree



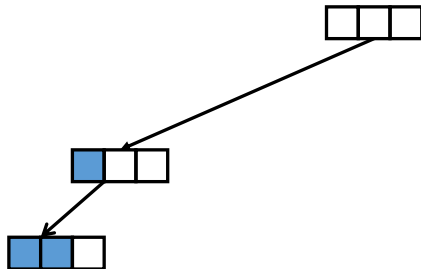
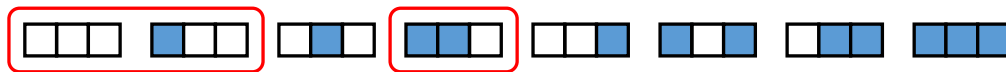
How to build a simple tree



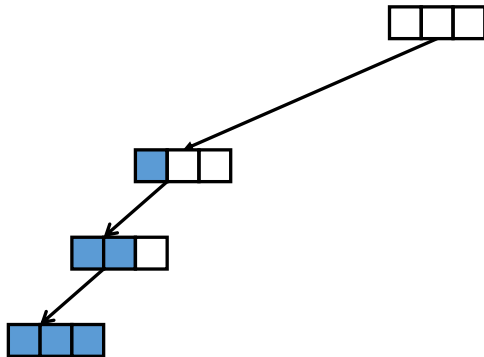
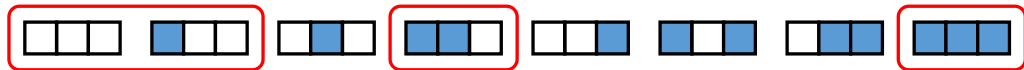
How to build a simple tree



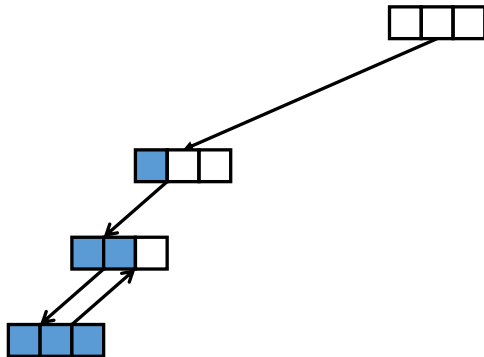
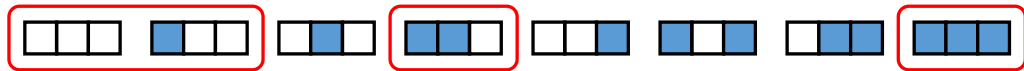
How to build a simple tree



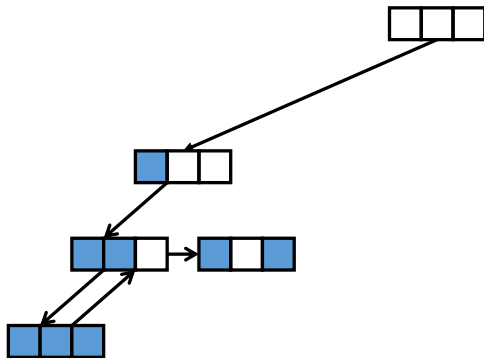
How to build a simple tree



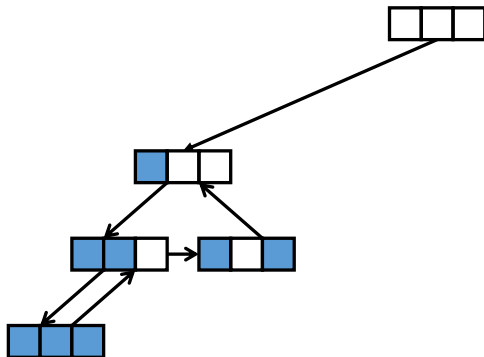
How to build a simple tree



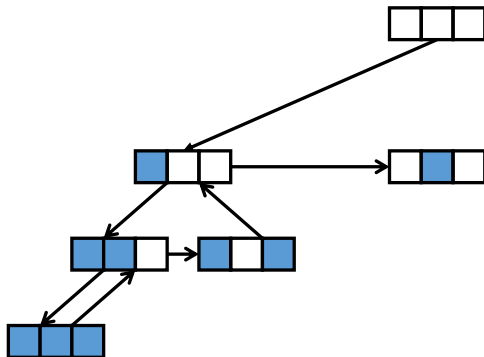
How to build a simple tree



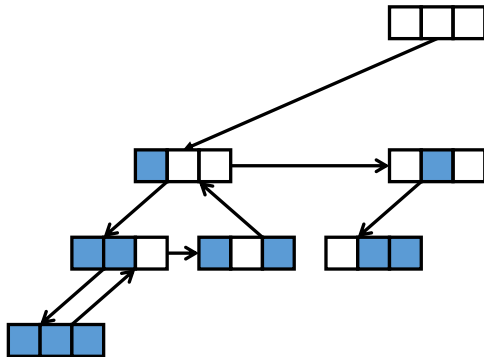
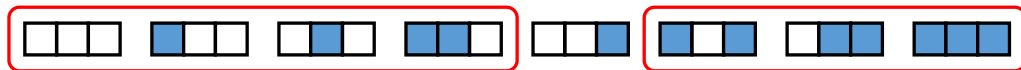
How to build a simple tree



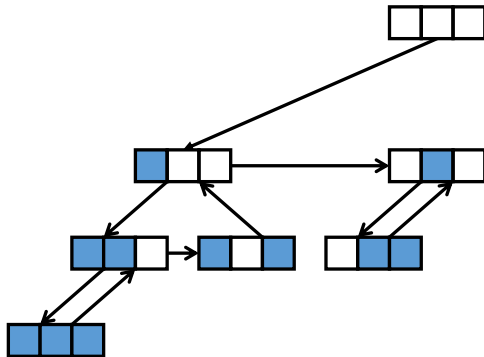
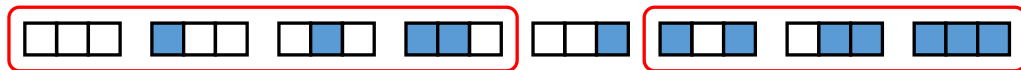
How to build a simple tree



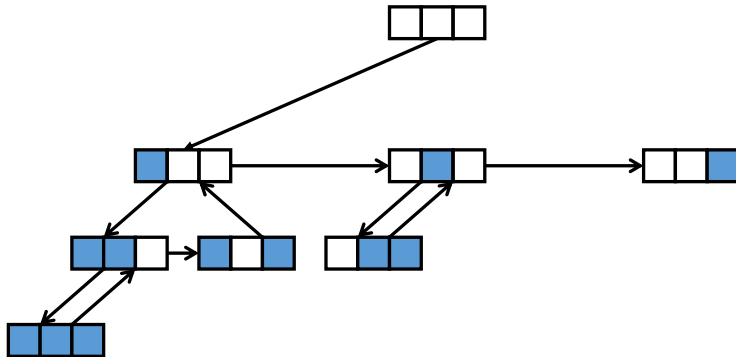
How to build a simple tree



How to build a simple tree



How to build a simple tree

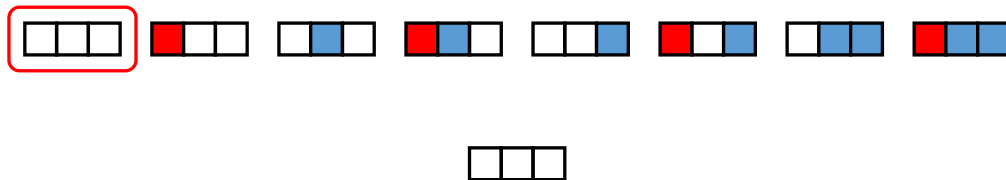




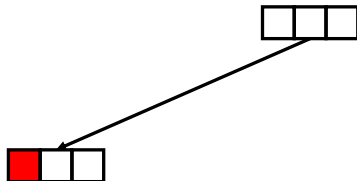
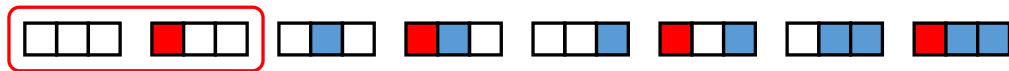
Pruning tree



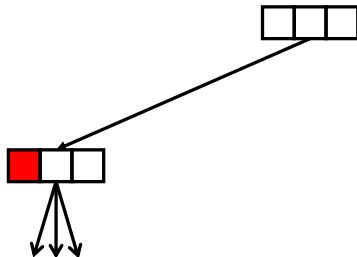
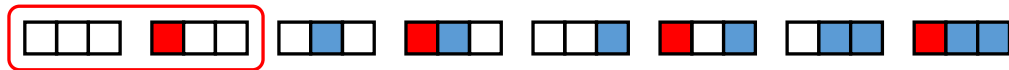
Pruning tree



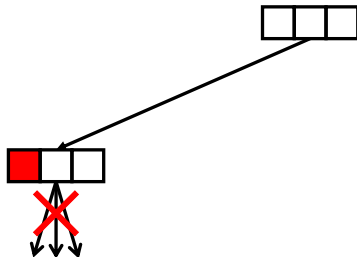
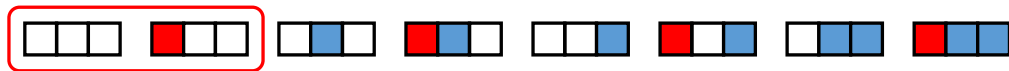
Pruning tree



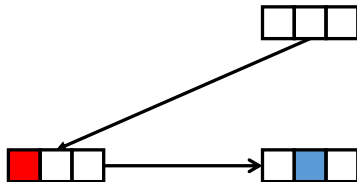
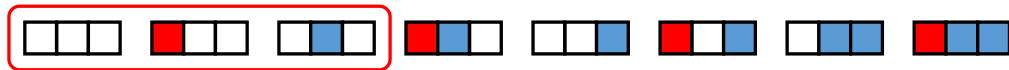
Pruning tree



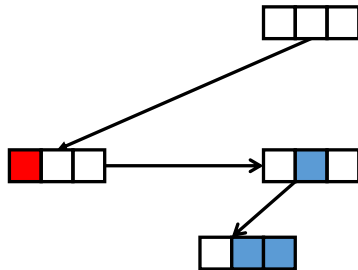
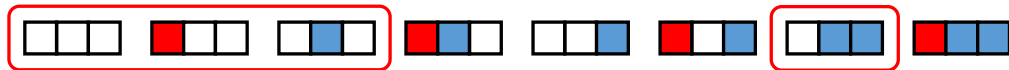
Pruning tree



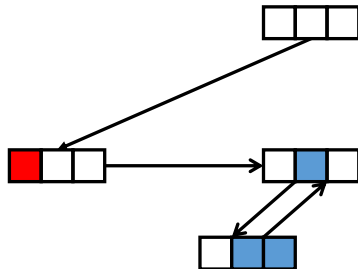
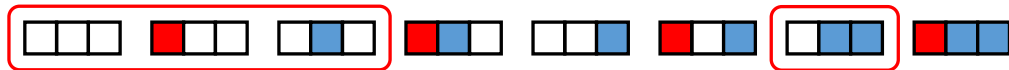
Pruning tree



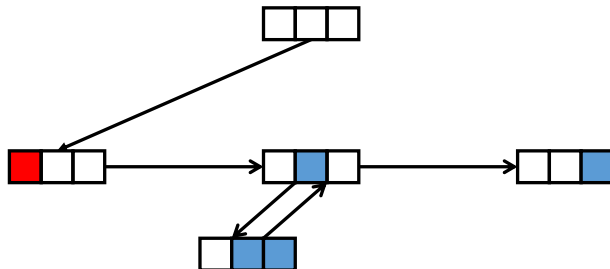
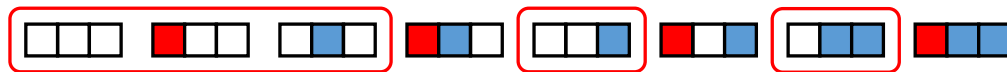
Pruning tree



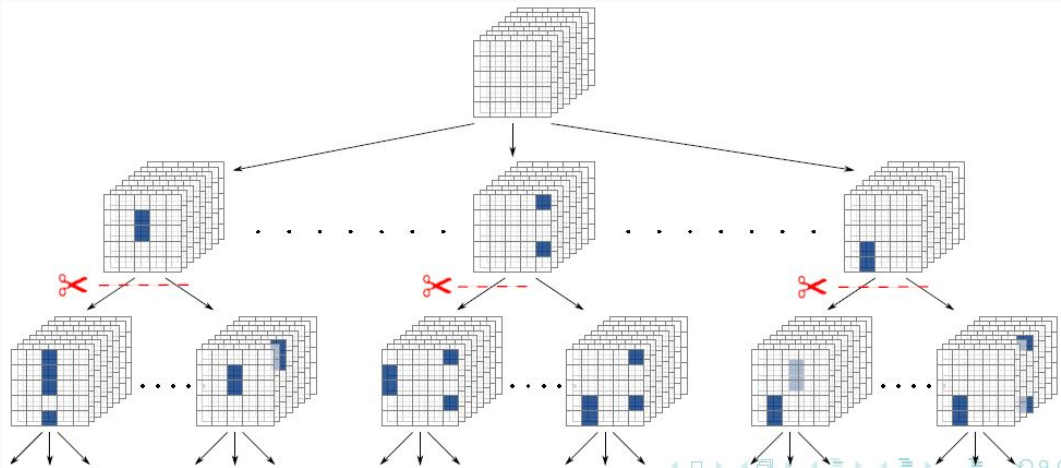
Pruning tree

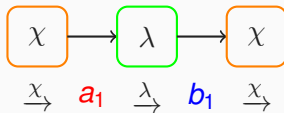


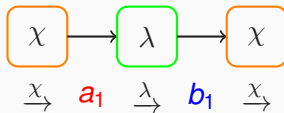
Pruning tree



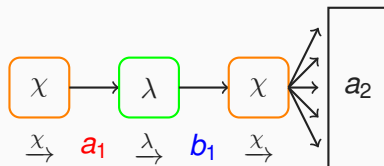
KECCAK tree search



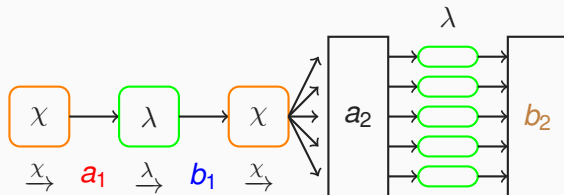




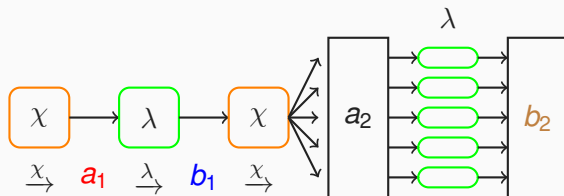
- Forward extension



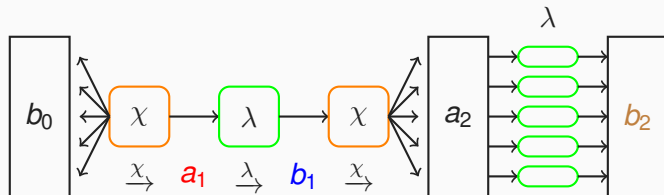
- Forward extension
 - Generating all possible a_2 's



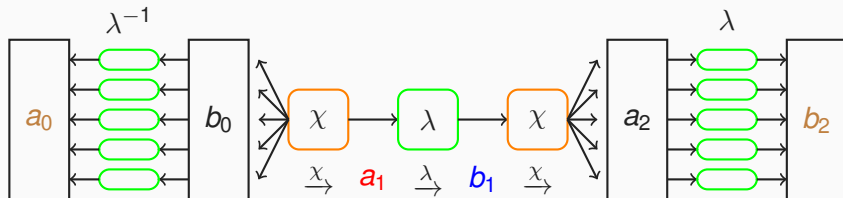
- Forward extension
 - Generating all possible a_2 's
 - Computing the corresponding b_2 for each a_2



- Forward extension
 - Generating all possible a_2 's
 - Computing the corresponding b_2 for each a_2
- Backward extension



- Forward extension
 - Generating all possible a_2 's
 - Computing the corresponding b_2 for each a_2
- Backward extension
 - Generating all possible b_0 's



- Forward extension
 - Generating all possible a_2 's
 - Computing the corresponding b_2 for each a_2
- Backward extension
 - Generating all possible b_0 's
 - Computing the corresponding a_0 for each b_0

Quick summary of generating r -round trails

- Goal: computing minimum weight over r -round of an unaligned primitive

Quick summary of generating r -round trails

- Goal: computing minimum weight over r -round of an unaligned primitive
- Start by forming a tree of all 2-round trail cores

Quick summary of generating r -round trails

- Goal: computing minimum weight over r -round of an unaligned primitive
- Start by forming a tree of all 2-round trail cores
 - Prune unwanted nodes ASAP

Quick summary of generating r -round trails

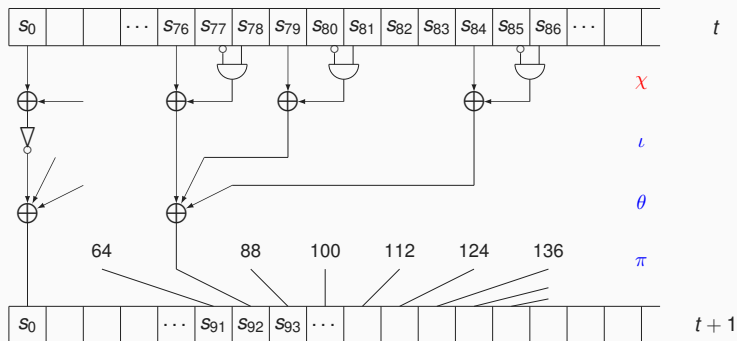
- Goal: computing minimum weight over r -round of an unaligned primitive
- Start by forming a tree of all 2-round trail cores
 - Prune unwanted nodes ASAP
- Extend 2-round trail cores to r -round

Quick summary of generating r -round trails

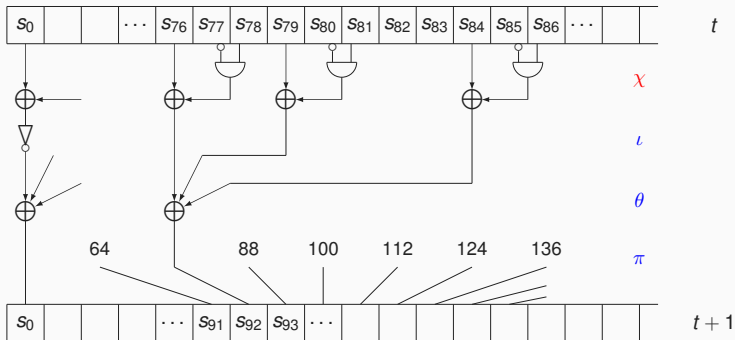
- Goal: computing minimum weight over r -round of an unaligned primitive
- Start by forming a tree of all 2-round trail cores
 - Prune unwanted nodes ASAP
- Extend 2-round trail cores to r -round
- Generate r -round differential trails

Application

The SUBTERRANEAN 2.0 round function

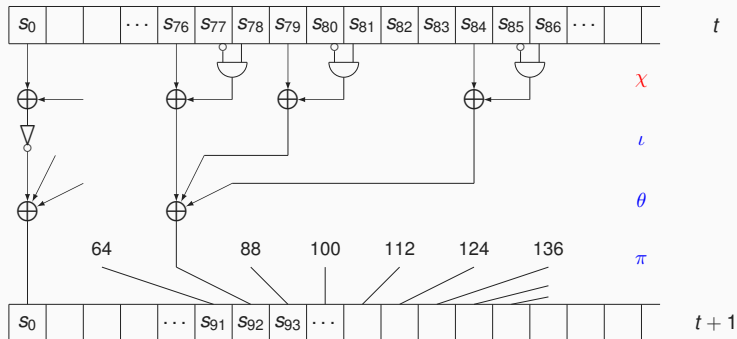


The SUBTERRANEAN 2.0 round function



$$\chi: \quad s_i \leftarrow s_i + (s_{i+1} + 1)s_{i+2}.$$

The SUBTERRANEAN 2.0 round function



$$\chi: S_i \leftarrow S_i + (S_{i+1} + 1)S_{i+2}.$$

$$\iota: S_i \leftarrow S_i + \delta_i,$$

$$\theta: S_i \leftarrow S_i + S_{i+3} + S_{i+8},$$

$$\pi: S_i \leftarrow S_{12i}.$$

- Goal: scanning 8-round trail cores in SUBTERRANEAN
 - 2-round up to 28

- Goal: scanning 8-round trail cores in SUBTERRANEAN
 - 2-round up to 28
 - 3-round up to 40

- Goal: scanning 8-round trail cores in SUBTERRANEAN
 - 2-round up to 28
 - 3-round up to 40
 - 4-round up to 57

- Goal: scanning 8-round trail cores in SUBTERRANEAN
 - 2-round up to 28
 - 3-round up to 40
 - 4-round up to 57 $\rightarrow$ no trail

- Goal: scanning 8-round trail cores in SUBTERRANEAN
 - 2-round up to 28
 - 3-round up to 40
 - 4-round up to 57 $\rightarrow$ no trail
 - The best 4-round trail we found weigh 58

- Goal: scanning 8-round trail cores in SUBTERRANEAN
 - 2-round up to 28
 - 3-round up to 40
 - 4-round up to 57 $\rightarrow$ no trail
 - The best 4-round trail we found weigh 58

state	weight	active bit positions
a_1	12	{0, 5, 8, 10, 12, 15, 16, 18, 21}
b_1	7	{65, 66, 85, 86, 87}
b_2	11	{7, 28, 134, 198, 200, 219}
b_3	28	{16, 18, 22, 39, 54, 86, 88, 107, 118, 139, 152, 173, 188, 211, 252}

- Goal: scanning 8-round trail cores in SUBTERRANEAN
 - 2-round up to 28
 - 3-round up to 40
 - 4-round up to 57 → no trail
 - The best 4-round trail we found weigh 58

state	weight	active bit positions
a_1	12	{0, 5, 8, 10, 12, 15, 16, 18, 21}
b_1	7	{65, 66, 85, 86, 87}
b_2	11	{7, 28, 134, 198, 200, 219}
b_3	28	{16, 18, 22, 39, 54, 86, 88, 107, 118, 139, 152, 173, 188, 211, 252}

# rounds:	1	2	3	4	5	6	7	8
lower bound:	2	8	25	58	?	?	?	?

8-round trail cores

- An 8-round trail core can be divided into two 4-round

8-round trail cores

- An 8-round trail core can be divided into two 4-round
- $w(Q_8) = w(Q_4) + w(Q'_4)$

8-round trail cores

- An 8-round trail core can be divided into two 4-round
- $w(Q_8) = w(Q_4) + w(Q'_4)$
- There is no 4-round trail up to 57

8-round trail cores

- An 8-round trail core can be divided into two 4-round
- $w(Q_8) = w(Q_4) + w(Q'_4)$
- There is no 4-round trail up to 57
- Worst case: two compatible 4-rounds with weight 58

8-round trail cores

- An 8-round trail core can be divided into two 4-round
- $w(Q_8) = w(Q_4) + w(Q'_4)$
- There is no 4-round trail up to 57
- Worst case: two compatible 4-rounds with weight 58
- $w(Q_8) = w(Q_4) + w(Q'_4) \geq 58 + 58 = 116$

8-round trail cores

- An 8-round trail core can be divided into two 4-round
- $w(Q_8) = w(Q_4) + w(Q'_4)$
- There is no 4-round trail up to 57
- Worst case: two compatible 4-rounds with weight 58
- $w(Q_8) = w(Q_4) + w(Q'_4) \geq 58 + 58 = 116$

# rounds:	1	2	3	4	5	6	7	8
lower bound:	2	8	25	58	62	78	83	≥ 116

Lower bound on the weight of differential trail cores

Thanks for your attention!

Question?