Simplified MITM Modeling for Permutations: New (Quantum) Attacks

André Schrottenloher[1] and Marc Stevens[2]

[1] Inria, Univ Rennes, CNRS, IRISA
[2] Cryptology Group, CWI

Hash functions and preimages

Small-range hash function or **compression function**:

$$H \ : \{0,1\}^{n+m} \to \{0,1\}^{n}$$

Preimage resistance is one of the security goals:

- For any target t, finding a **preimage** of t (x such that $H(x) = t$) should take time 2^n
- If we can find a preimage in less than 2^n time, we have an **attack**

Most compression functions are built from **permutations** using a **feedforward** (XOR input and output).

Example

Example: Haraka-512 (v2)

$$\text{Haraka-512} : \begin{cases} \{0,1\}^{512} \to \{0,1\}^{256} \\ x \mapsto \text{trunc}_{256}(x \oplus P(x)) \end{cases}$$

defined using a **permutation** P on 512 bits.

Finding a preimage of t by Haraka-512

$\Updownarrow$

Finding x such that $\text{trunc}_{256}(x \oplus P(x)) = t$

$\Updownarrow$

Finding x such that $\text{trunc}_{256}(x) \oplus t = \text{trunc}_{256}(P(x))$

$\Updownarrow$

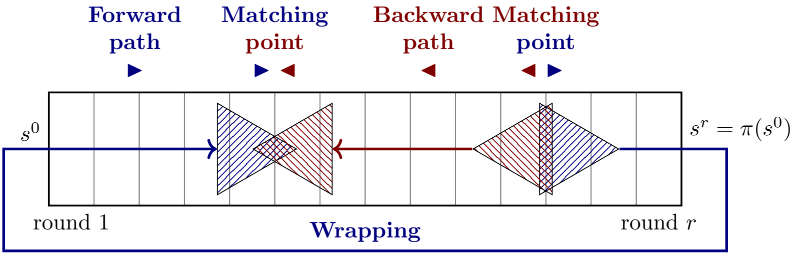
Finding x such that x and $P(x)$ have 256 bits of linear relation.

The MITM Paradigm

- We search for x with some relation between x and $P(x)$
- We can take all possible x and check the relations: that's the generic attack

However, P is made of multiple rounds, and the internal state can be cut in multiple parts.

The MITM Paradigm (ctd.)



- guess **a subset of the internal state** and compute forwards: that's the **forward path** ▶
- guess **an independent subset of the internal state** and compute backwards: that's the **backward path** ◀
- use the **matching points** between forward & backward to sieve the pairs and find solutions

The MITM Paradigm (ctd.)

How it started:

- Forward / backward, then match

How it's going:

- Splice-and-cut
- Guess-and-determine
- Initial structure
- 3-subset MITM
- Nonlinearly constrained neutral words **[DHS+21]**
- Superposition MITM **[BGST22]**
- ...

Automatic search of MITM attacks

- **Search space**: "all possible **forward** ▶ / **backward** ◀ paths"
- **Objective function**: "the attack complexity"

We **minimize** the **objective** on the **search space**: that gives the best MITM attack.

- **[BDG+21]** use a MILP model
- The **search space** is constrained by a complex set of **local rules**
- Subsequent works **[DHS+21,BGST22]** added more techniques, but also, more rules

MILP

Minimize $x_1 + 2x_2 + 6x_3$ under the constraints:

$$\begin{cases} y_1 + x_1 + x_2 - 2x_3 \leq 0 & y_1 \geq 0 \\ x_3 + 5x_2 \geq 0 & x_2 \text{ is integer} \quad \ldots \end{cases}$$

Bao, Dong, Guo, Li, Shi, Sun, Wang, "Automatic search of Meet-in-the-middle preimage attacks on AES-like hashing", EUROCRYPT 2021

Automatic search of MITM attacks (ctd.)

We use a different modeling strategy than **[BDG+21]**:

- We target **permutations** only, but more permutations than before
- We use MILP, but a different model (very **simple**)
- Our objective includes **quantum attacks**
- Applications to AES, **Haraka**, Grøstl, Spongent, Simpira, Sparkle

Quantum MITM attacks

- Haraka has been explicitly designed for post-quantum hash-based signatures (e.g. SPHINCS+)
- We need to look at its **classical and quantum** preimage security

- Generic classical preimage in time 2^{256} (exhaust. search)
- Generic quantum preimage in time 2^{128} (Grover search)

- Haraka-512 is broken: there is a MITM preimage attack in time $2^{240} < 2^{256}$ **[BDG+21]**
- But it could still be quantumly safe

- (New) Haraka-512 is **quantumly broken:** there is a **quantum** MITM preimage attack in time $< 2^{128}$

Outline

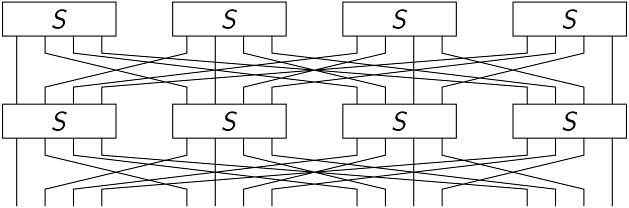
The search space: cells

SPN permutations and Present

An SPN permutation is like an SPN cipher without a key.

- State: b **cells** of w bits each
- Round: **S-Box layer** (S) and **linear layer** (P)

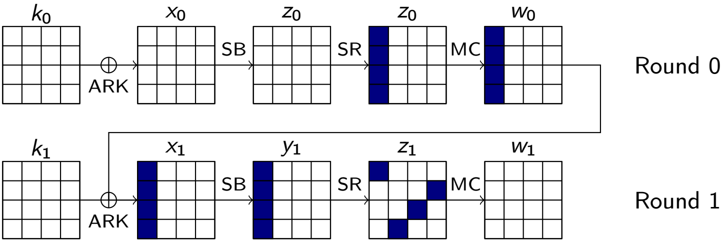
Present is a block cipher with 16 cells of 4 bits ($= 64$ bits). The linear layer permutes the bits.



(That's a small **Present** with 4 cells).

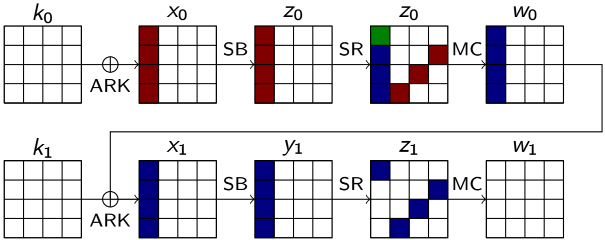
The case of AES

- **AES** is a block cipher with an 4×4 state of 8-bit S-Boxes (bytes).
- **The linear layer** permutes the bytes (Shiftrows) and mixes the columns (MixColumns).

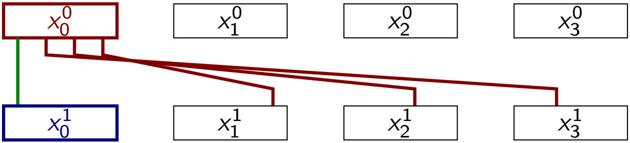


The Super S-Box

If we put together MC and SB, it creates a **Super S-Box** acting on $4 \times 8 = 32$ bits. This:



Looks like this:

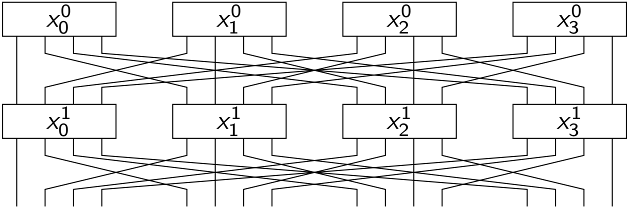


In our abstraction, **AES is a disguised Present.**

Cell-based representation

- Remove any round constants
- Consider each S-Box as an arbitrary function S_j^i
- Replace each S-Box S_j^i by a **cell** x_j^i
- Each cell has a list of **possible assignments**: the table of S_j^i

- Only linear relations between cells remain
- We must find an **assignment** to **all cells** that satisfies **all linear relations**

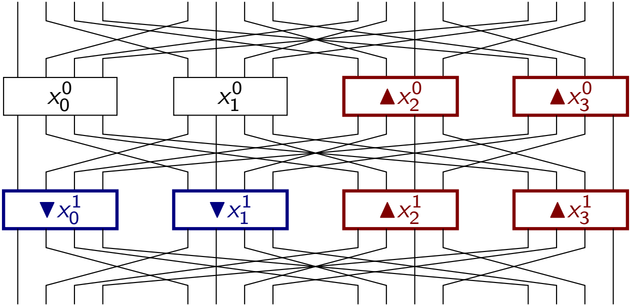


The search space

A MITM attack path is defined by 3 sets of cells:

- X_F: **forwards ▶**
- X_B: **backwards ◀**
- Merged: $X_M = \mathrm{X_F} \cup \mathrm{X_B}$

This is our **search space**: the possible choices of X_F and X_B.



The objective: complexity

The attack algorithm

$\mathcal{R}[X] :=$ all assignments of cells in X that satisfy the linear relations.

The attack does:

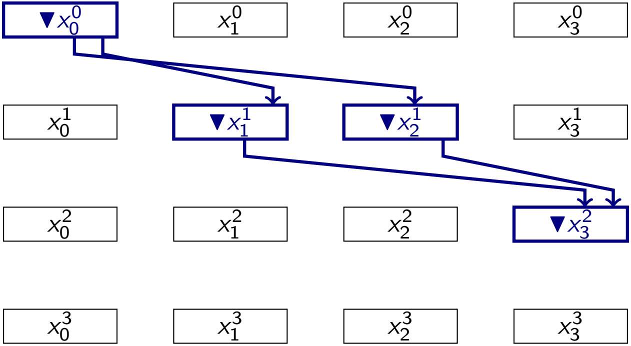
1. Compute $\mathcal{R}[\mathsf{X_F}]$ (forward path)
2. Compute $\mathcal{R}[\mathsf{X_B}]$ (backward path)
3. Compute $\mathcal{R}[X_M]$
   - For each assignment in $\mathcal{R}[X_M]$, check if this is a solution

The forward list $\mathcal{R}[X_F]$

We go **forwards**. Round by round, we guess the missing bits: 4 at rd. 0, 3 + 3 at rd. 1, 2 at rd. 2

- So, we have a valid assignment in time 1, memoryless.
- $\mathcal{R}[X_F]$ can be computed in $|\mathcal{R}[X_F]|$:

$$\text{(in } \log_2\text{)} \sum \text{weights of cells} - \sum \text{weights of edges} = 4 \times 4 - 4 = 12$$

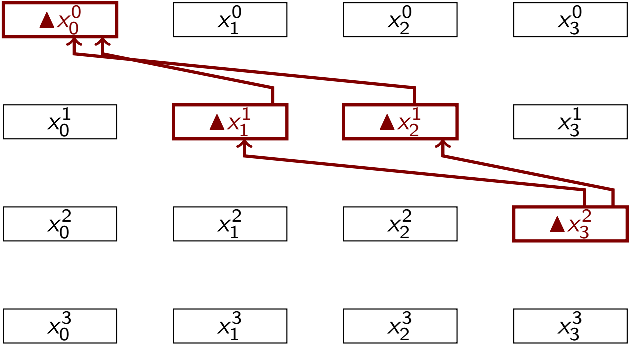


The **backward** list $\mathcal{R}[X_B]$

We go **backwards**. Round by round, we guess the missing bits: 4 at rd. 3, 3 + 3 at rd. 2, 2 at rd. 1

- So, we have a valid assignment in time 1, memoryless.
- $\mathcal{R}[X_B]$ can be computed in $|\mathcal{R}[X_B]|$:

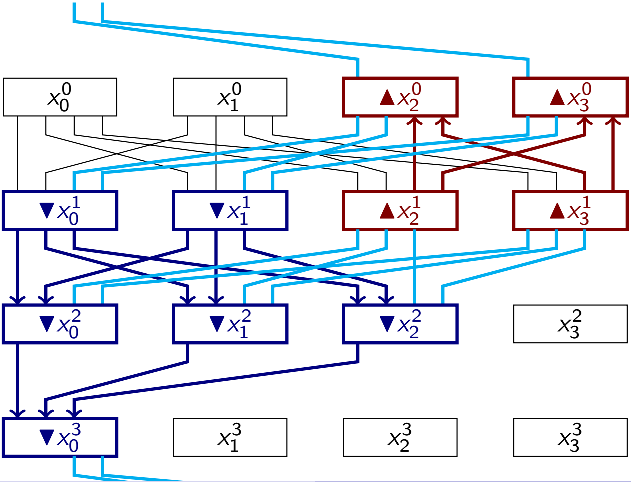
$$\text{(in } \log_2) \sum \text{weights of cells} - \sum \text{weights of edges} = 4 \times 4 - 4 = 12$$



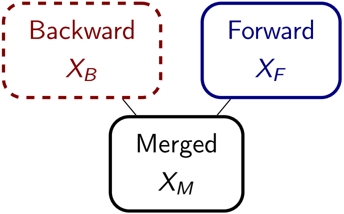
The merged list $\mathcal{R}[\mathsf{X_F} \cup \mathsf{X_B}]$

$$|\mathcal{R}[X_M]| = |\mathcal{R}[\mathsf{X_F}]| \times |\mathcal{R}[\mathsf{X_B}]|/(2^{\text{new edges}})$$

Forward: 15 bits (3.75 cells); **Backward:** 12 bits (3 cells)
Merged: $15 + 12 - 12 = 15$ bits (3.75 cells).



Classical / quantum merging



- Build the smallest list (e.g., **forward**)
- Sort it
- Go through the **backward** list and search for matches
- Test any produced partial solution

Attack complexity

- Classical time:

$$|\mathcal{R}[X_F]| + \max(|\mathcal{R}[X_B]|, |\mathcal{R}[X_M]|)$$

- Classical memory:

$$|\mathcal{R}[X_F]|$$

- Quantum time:

$$|\mathcal{R}[X_F]| + \sqrt{\max\left(|\mathcal{R}[X_B]|, |\mathcal{R}[X_M]|\right)}$$

- Quantum memory:

$$|\mathcal{R}[X_F]|$$

- the complexities depend on $|\mathcal{R}[X_F]|, |\mathcal{R}[X_B]|, |\mathcal{R}[X_M]|$
- the complexities depend only on X_F and X_B

MILP strategy

Search space: boolean variables for $\mathsf{X_F}$ and $\mathsf{X_B}$

$\Downarrow$

Deduce the quantities $\log_2 |\mathcal{R}[\mathsf{X_F}]|$, $\log_2 |\mathcal{R}[\mathsf{X_B}]|$, $\log_2 |\mathcal{R}[X_M]|$ by linear inequalities

$\Downarrow$

Deduce the time and memory complexities (classical and quantum, in $\log_2$) of an attack based on $\mathsf{X_F}$ and $\mathsf{X_B}$. **This is the objective function.**

Technical details

1. Reducing the memory

- Matching points of the form $\leftarrow\rightarrow$ can be turned into **global guesses** $\leftrightarrow$
- This precomputes some matches and reduces the list sizes

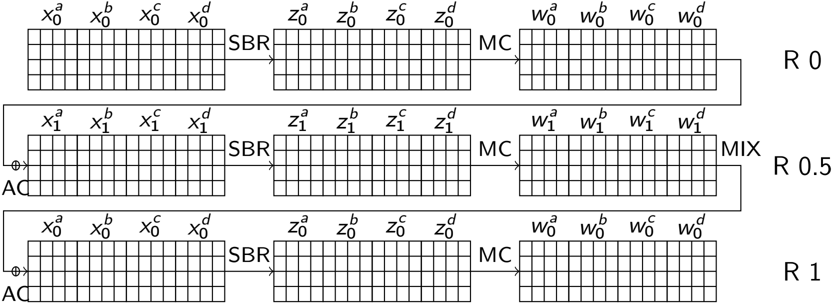
2. AES MixColumns

- In the AES case, the box is actually a (linear, MDS) MixColumns operation
- We can "match through MixColumns" to reduce $|\mathcal{R}[X_M]|$
- This can be modeled easily

Attacking Haraka-512

Haraka-512 v2

- 512-bit to 256-bit hash function: $x \mapsto \mathrm{trunc}_{256}(P_{512}(x) \oplus x)$
- AES-based
- Finding a preimage is a MITM problem on P_{512}

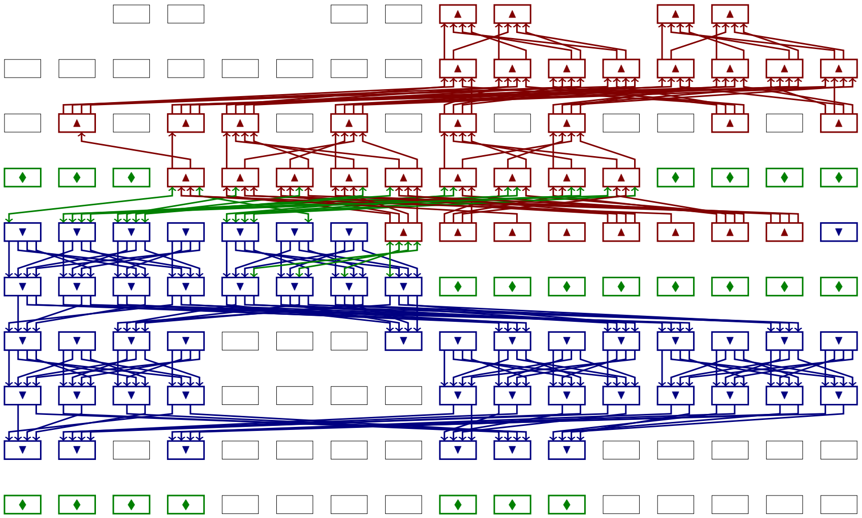


Attacking Haraka-512 v2

Ref.	Rounds	Model	Time	Memory
[BDG+21]	5.5/5	Classical	2^{240}	2^{128}
New	5.5/5	Classical	2^{240}	2^{16}
New	5.5/5	Quantum	$2^{123.34}$	2^{16}
New	**5/5**	**Classical**	2^{224}	2^{32}

Attacking Haraka-512 v2 (ctd.)

Path for 5/5 rounds:



Attacking Haraka-512 v2 (ctd.)

A **low-memory** (full) MITM preimage attack can be a **partial preimage attack** that we **repeat many times.**

$\Longrightarrow$ for Haraka-512 v2, 64-bit partial preimages in about 2^{32} time and memory.

Input x (512 bits $= 4 \times 128$ bits)

e7	d4	9f	2d	16	f6	53	65	cd	99	33	01	d5	2a	66	a1
3f	05	c4	94	7a	61	37	17	6b	8c	47	1f	1f	10	08	cc
2e	53	f6	6a	5e	83	a9	6d	f0	7a	b2	b9	69	a4	45	f4
b2	5c	0b	93	7a	ee	e2	c6	17	52	b3	74	e9	e4	79	f3

Output Haraka-512(x) (256 bits)

`4d 35 de 97 63 ba c0 f0 4c dc 64 6b d1 e6 19 15`

`00 00 00 00 cb 51 f8 2b 9d 3e 50 e1 00 00 00 00`

Conclusion

- Modeling MITM attacks can be **very simple** for permutations
- MITM attacks perform well in the quantum setting
- Ongoing work: extending our approach to the key-schedule path

Full version: ePrint 2022/189
Code: `github.com/AndreSchrottenloher/mitm-milp`

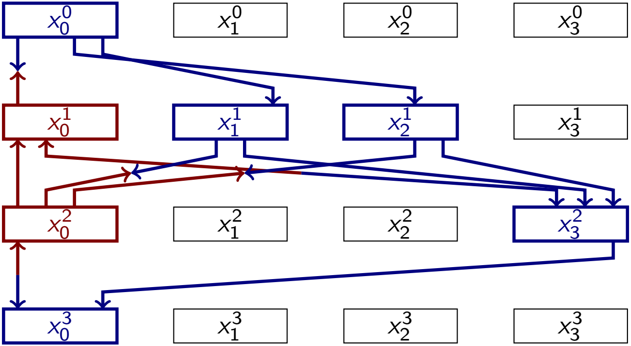
Thank you!

Technical details

When the two lists meet

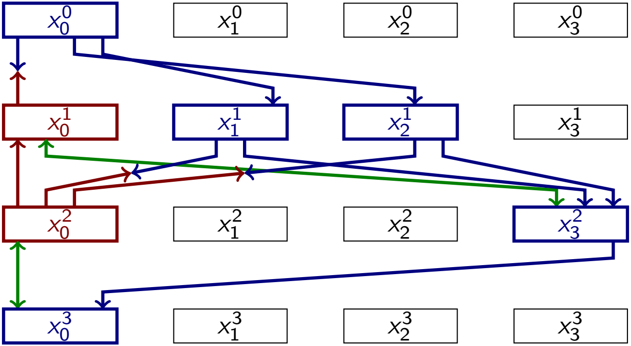
Backward and forward will meet at **matching points**. There are two types of matching points:

- $\rightarrow\leftarrow$ (forward cell up, backward cell down)
- $\leftarrow\rightarrow$ (backward cell up, forward cell down)



When the two lists meet (ctd.)

- We don't do anything special with the $\rightarrow\leftarrow$ matchings: they simply reduce the merged list size.
- But we turn the $\leftarrow\rightarrow$ into $\leftrightarrow$: we **guess globally** the value of these edges. This reduces the memory.



Global guesses and memory reduction

We guess an amount g of $\leftrightarrow$ edges, **then** we merge the **forward** and **backward** lists.

- All list sizes are reduced by g, which compensates the new loop on g.
- The classical time complexity is unchanged, but the memory complexity is reduced.
- The quantum time complexity **is changed**.

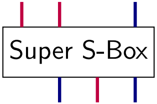
- g is still defined from X_F and X_B by linear inequalities.

AES MixColumns

Due to MC, the AES Super S-Box has the property:

If we know $c > 4$ edges in input and output to the Super S-Box, then we can match an amount of $c - 4$.

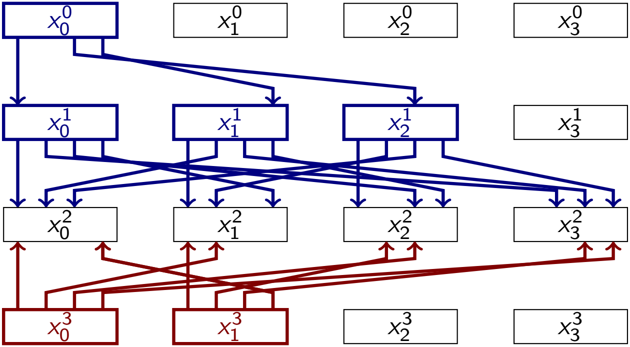
We use this for additional degrees of matching, which can also be converted into global guesses.



E.g., if we have: $(y_0, y_1, *, y_3) = MC(*, x_1, x_2, x_3)$ we can rewrite this as a system of two linear equations in $y_0, y_1, y_2, x_1, x_2, x_3$ (because MC is MDS).

AES MixColumns (ctd.)

Here is an AES-like situation of matching through MixColumns: there is 1 bytes of matching at each of $x_0^2, x_1^2, x_2^2, x_3^2$.



AES MixColumns (ctd.)

- new AES-specific rule: cells with enough $\downarrow\uparrow$ edges are **added to X_M**
- now these implicit matchings are properly counted
- finally, we can also convert these matching into guesses (like **global** $\leftrightarrow$ edges)