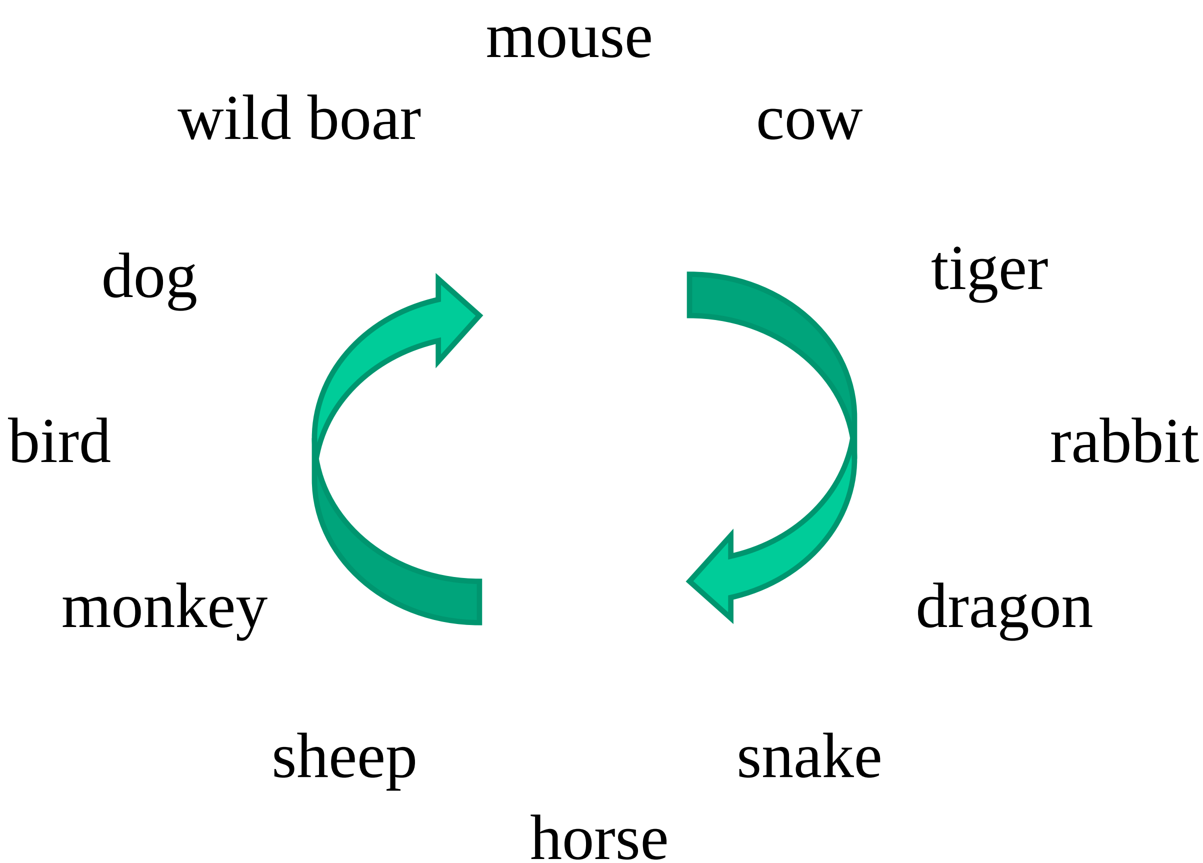
Finding Preimages of Tiger Up To 23 Steps

Lei Wang[1], and **Yu Sasaki**[1,2]

1. The University of Electro-Communications
2. NTT Corporation

08/Feb./2010 @ FSE2010 Seoul, Korea

Symbolic animal of the year



Symbolic animal of the year

mouse

wild boar

cow

dog

tiger

bird

rabbit

monkey

dragon

sheep

snake

horse

This year is a **TIGER** year !!

Outline

- Motivation
- Tiger hash function
- Pseudo-preimage attack on 23-step Tiger
- Conclusion

Recent progress in preimage attacks

- Since 2008, meet-in-the-middle preimage attacks have been developed for various MD4-based hash functions.
- Problems: weak message expansion (Reordering message index in each round)
- At CRYPTO'09, Aoki and Sasaki proposed an attack framework for linear message expansion.

Is the attack applied to non-MD4-based hashs?

Design strategy of Tiger

	Tiger	MD4-family
Key schedule function	Non-linear expansion	Linear expansion
Non-linearity of step function	S-box	Bitwise Boolean function
Number of steps	24 (small)	At least 48 (large)
Word size	64 bits	32 bits
Shift/Rotation	Bit shift	Bit rotation

- Tiger's strategy:
  - strong and heavy computations
  - small number of rounds
- Can these prevent MitM preimage attacks?

Comparison with previous work

◆ Preimage attack on Tiger (24steps for full specification)

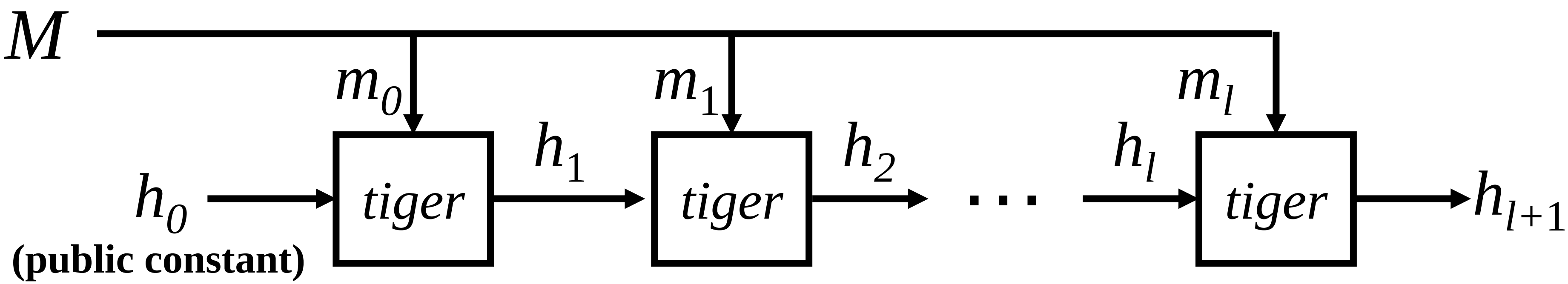
	#steps	*complexity*	*memory*	*note*
Indesteege *et al.*	13	$2^{128.5}$	*Negl.*	WeWoRC2007
Isobe *et al.*	16	2^{161}	2^{32}	FSE2009
Mendel	17	2^{185}	2^{160}	Africacrypt2009
Ours	**23**	**1.4×2^{189}** **$2^{187.5}$**	**2^{22}** **2^{22}**	**Preimages** **2nd Preimages**
Guo *et al.*	24 (full)	$2^{184.3}$	$2^{16.7}$	ePrint 2010

Outline

- Motivation
- Tiger hash function
- Pseudo-preimage attack on 23-step Tiger
- Conclusion

Tiger

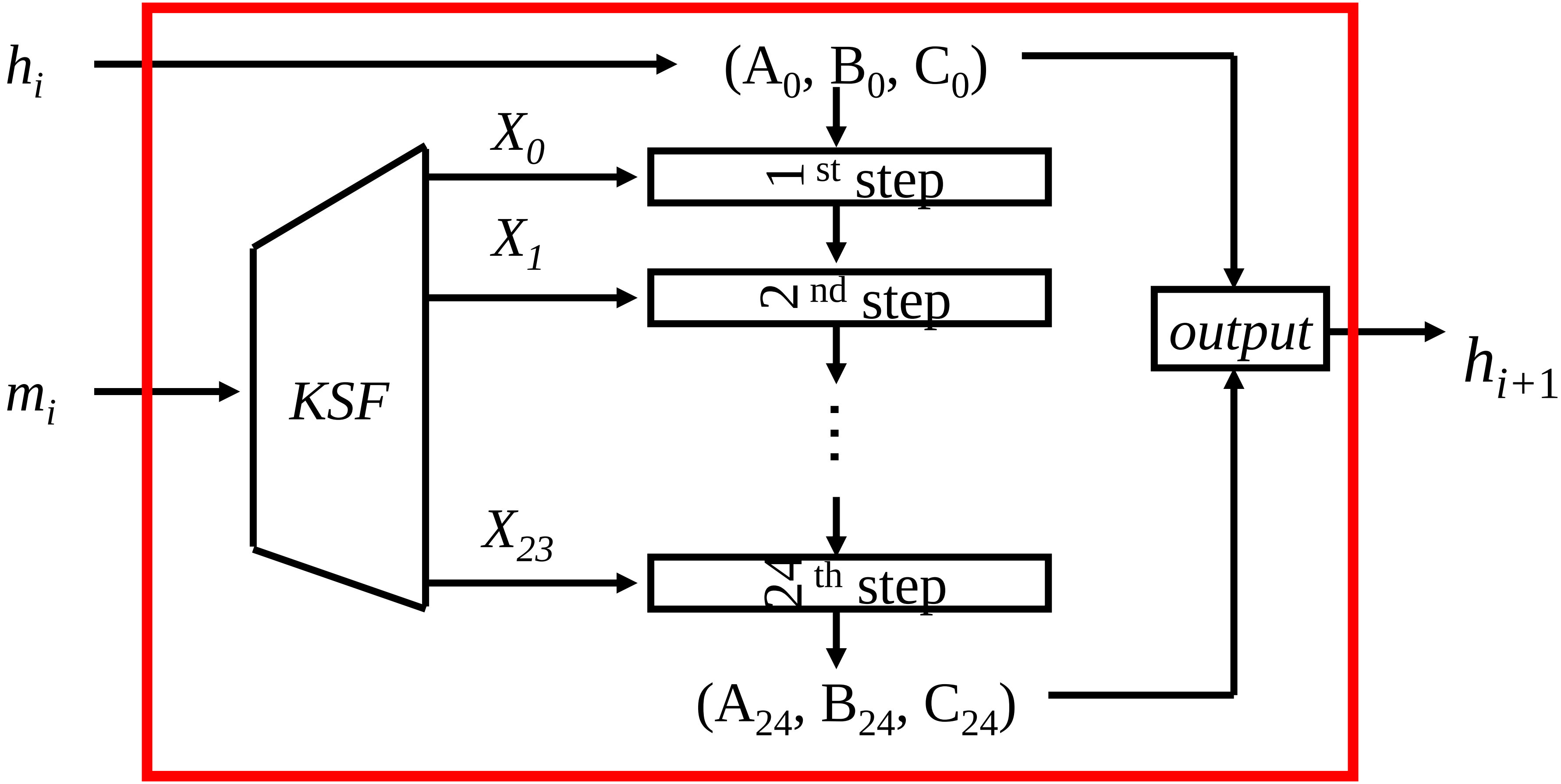
- An iterated hash function designed by Anderson and Biham.
  - Narrow-pipe Merkle-Damgård structure



  - m_i: 512 bits, h_i: 192 bits.
  - *tiger*: compression function of Tiger

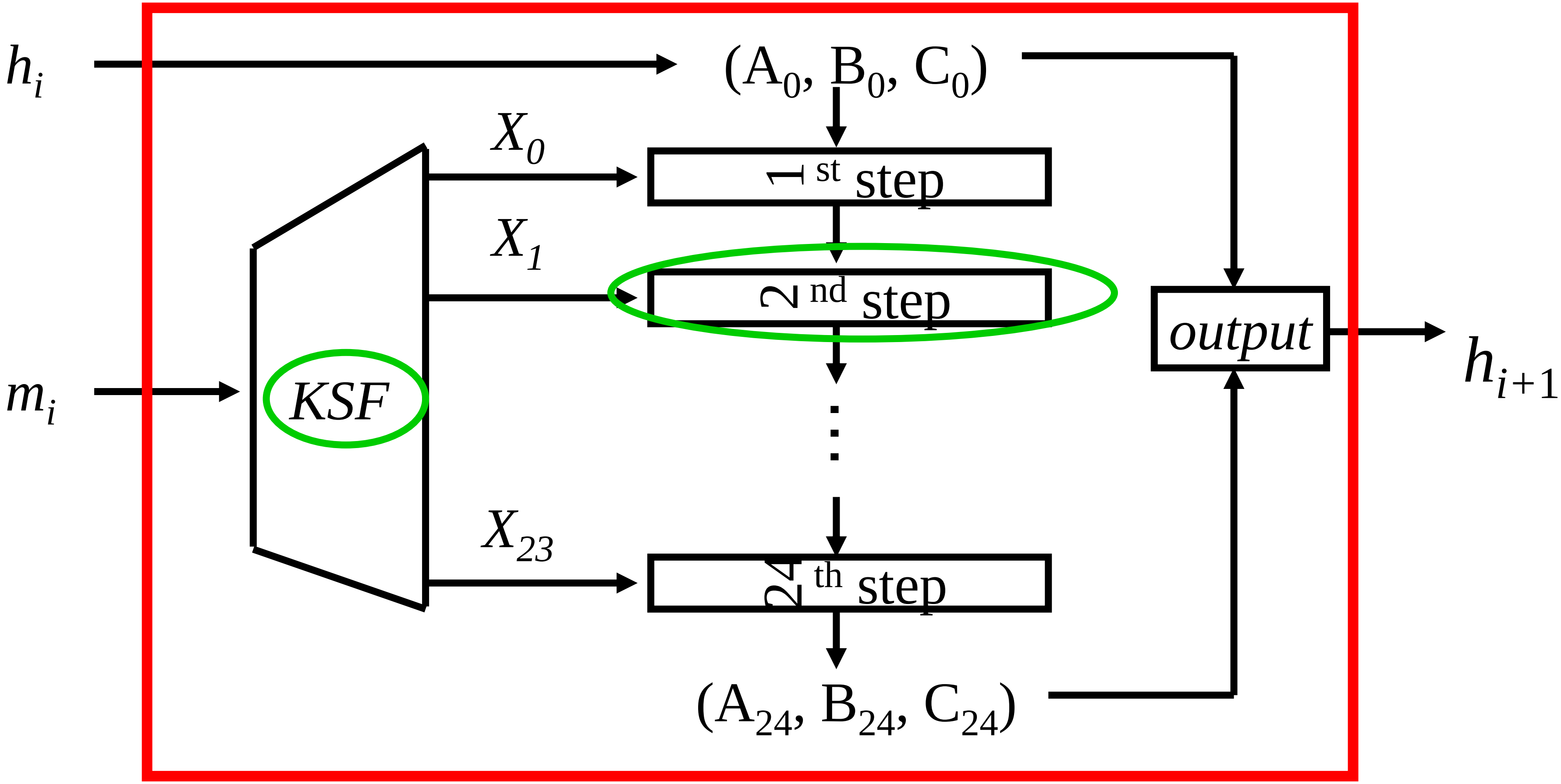
$$\{0,1\}^{192} \times \{0,1\}^{512} \rightarrow \{0,1\}^{192}$$

Structure of *tiger*



- X_j, A_j, B_j, C_j: 64-bit
- *KSF*: key schedule function
- $h_i = A_0 || B_0 || C_0$
- $h_{i+1} = (A_{24} \oplus A_0) || (B_{24} - B_0) || (C_{24} + C_0)$

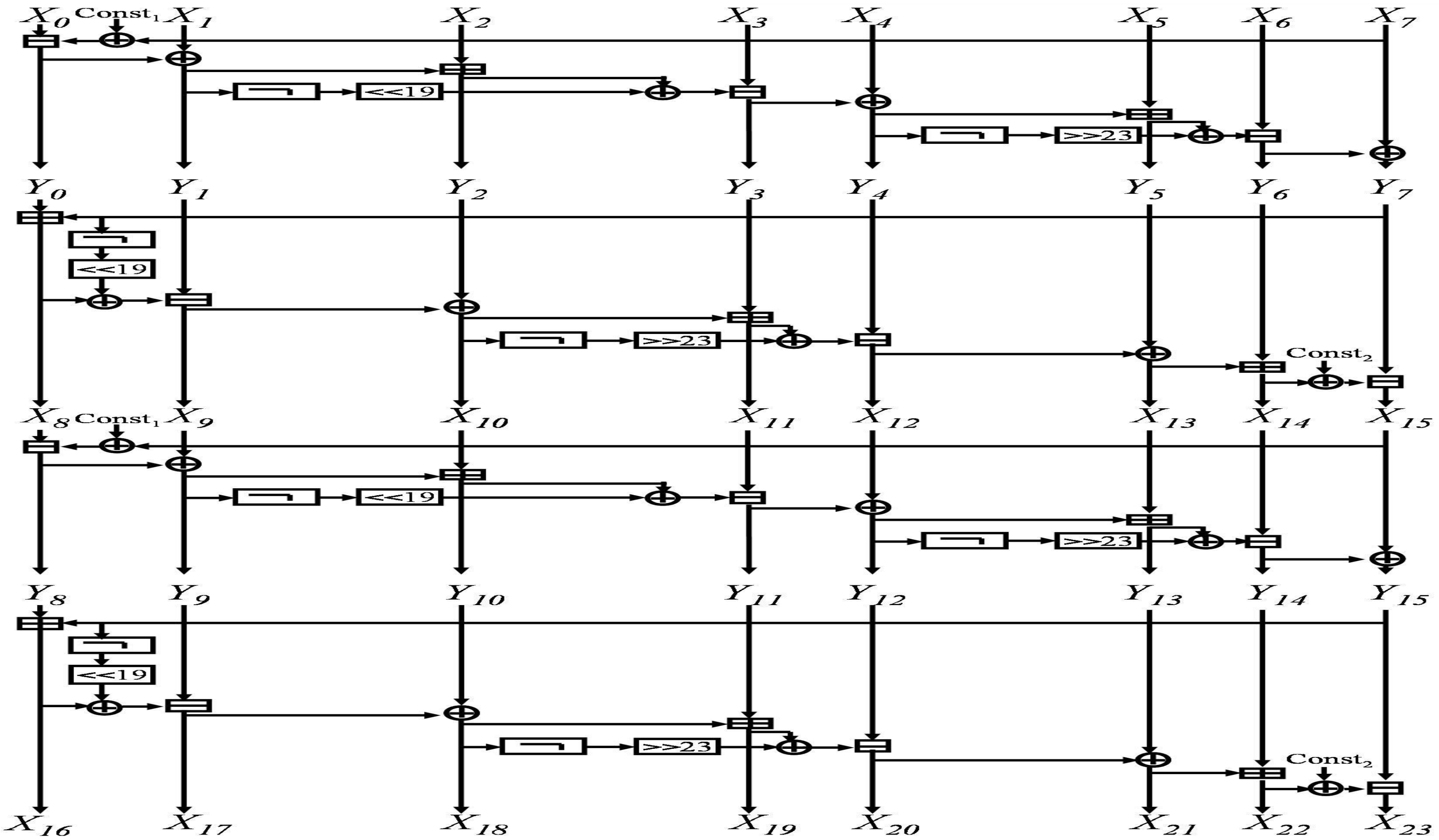
Structure of *tiger*



- X_j, A_j, B_j, C_j: 64-bit
- *KSF*: key schedule function
- $h_i = A_0 || B_0 || C_0$
- $h_{i+1} = (A_{24} \oplus A_0) || (B_{24} - B_0) || (C_{24} + C_0)$

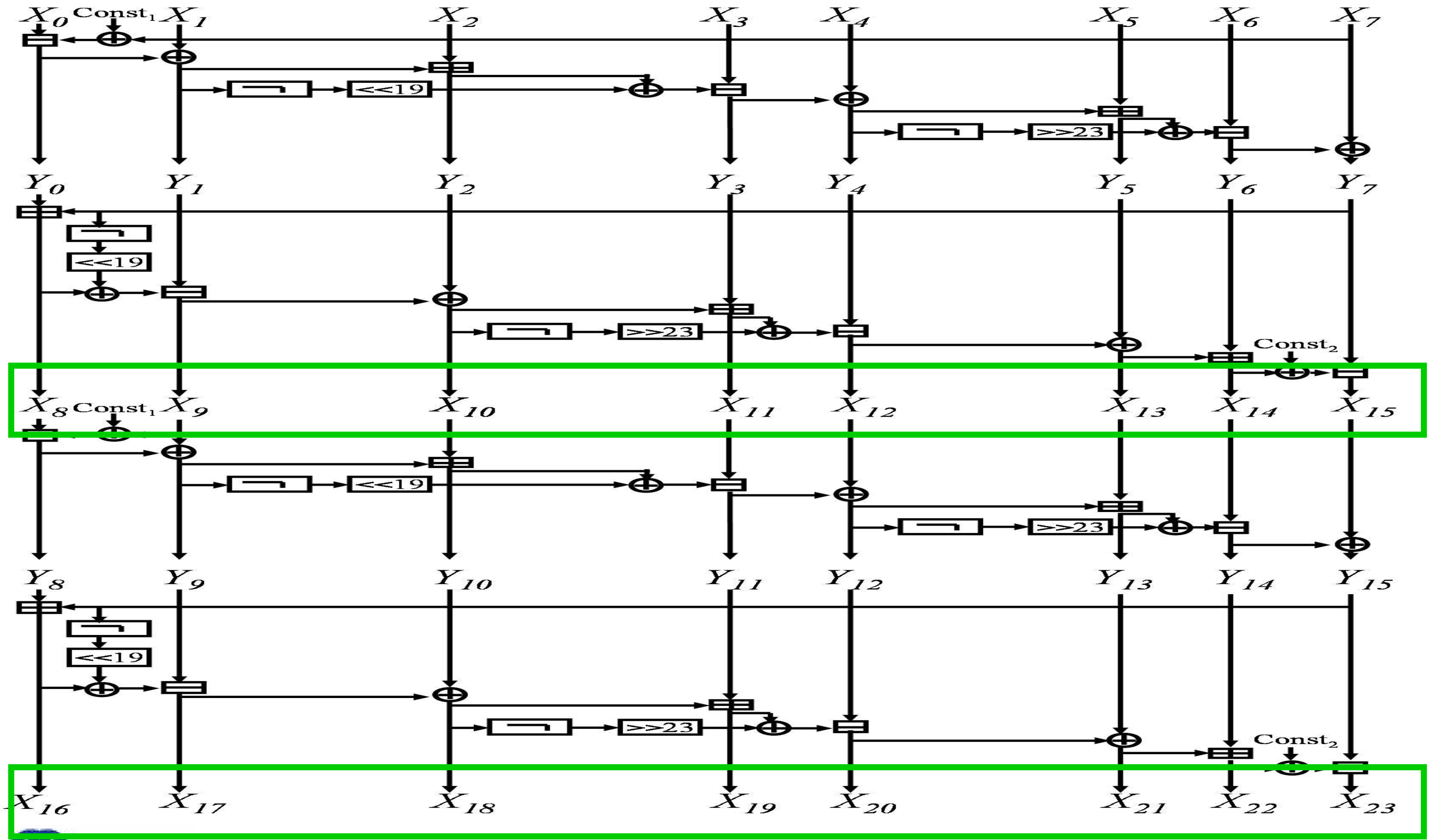
KSF of *tiger*

- $m_i = X_0||X_1||X_2||X_3||X_4||X_5||X_6||X_7$



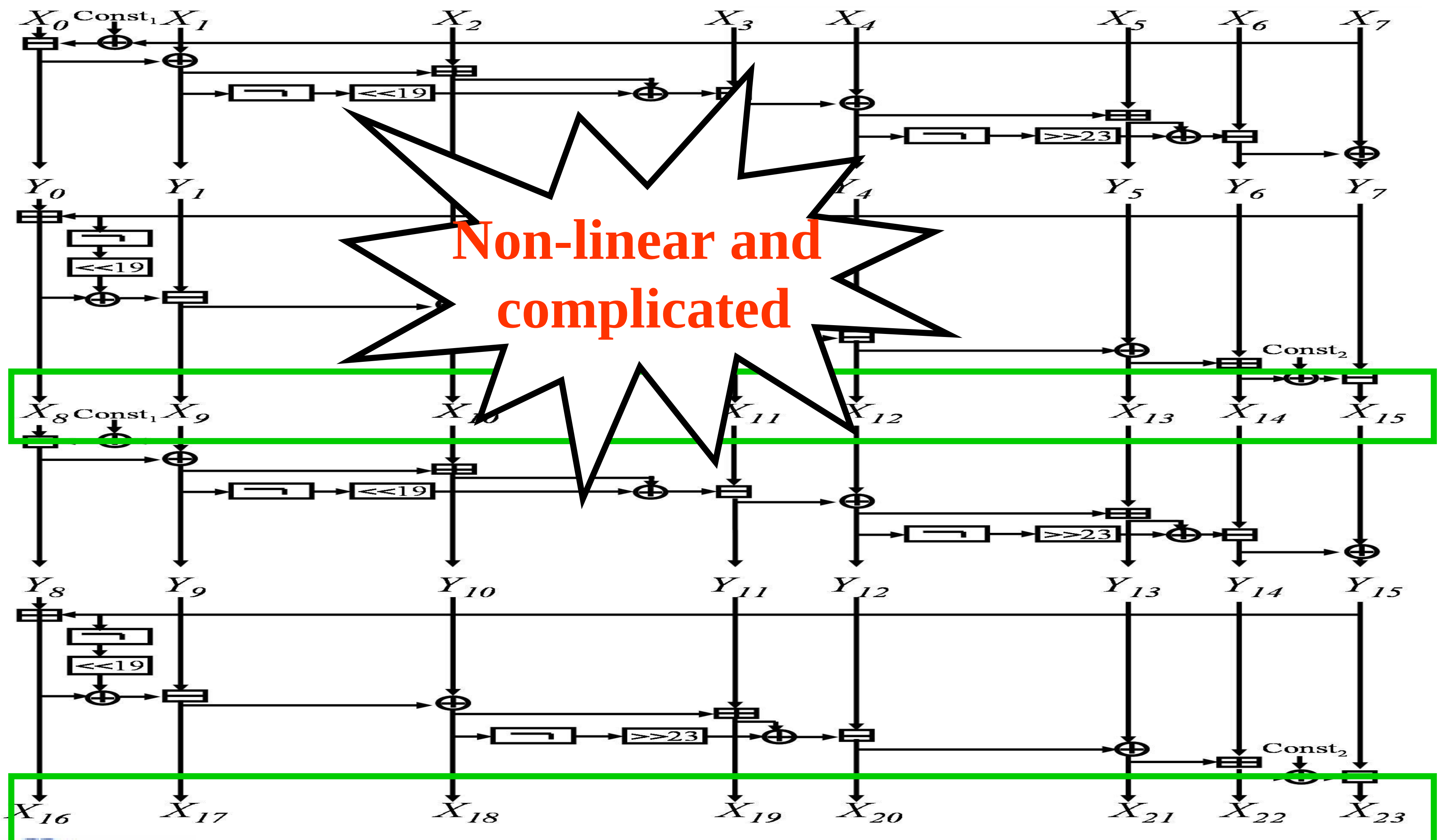
KSF of *tiger*

- $m_i = X_0||X_1||X_2||X_3||X_4||X_5||X_6||X_7$

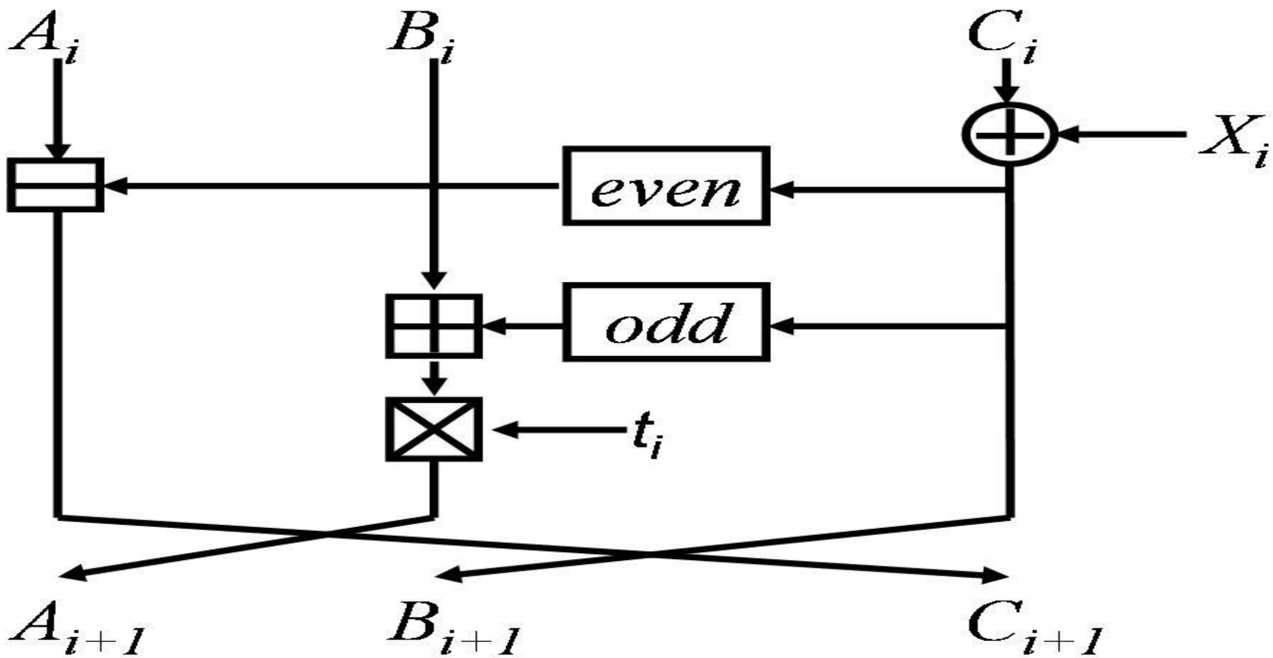


KSF of *tiger*

- $m_i = X_0||X_1||X_2||X_3||X_4||X_5||X_6||X_7$

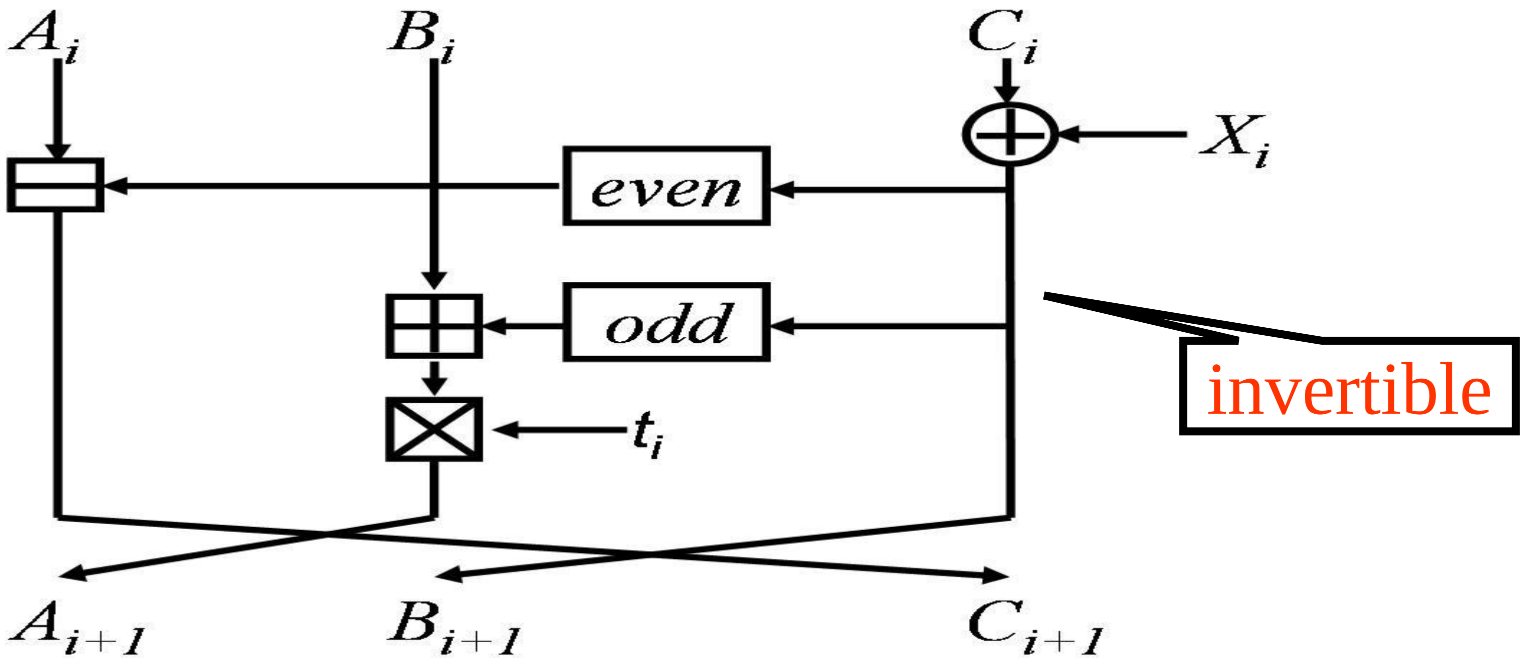


Step function of *tiger*



- $X_i \oplus C_i = c_7||c_6||\cdots||c_0$
- c_j: 8-bit
- t_i: a small constant
- $even = T_0(c_0) \oplus T_1(c_2) \oplus T_2(c_4) \oplus T_3(c_6)$
- $odd = T_3(c_1) \oplus T_2(c_3) \oplus T_1(c_5) \oplus T_0(c_7)$
- T_j: S-boxes mapping 8-bit values to 64-bit values

Step function of *tiger*



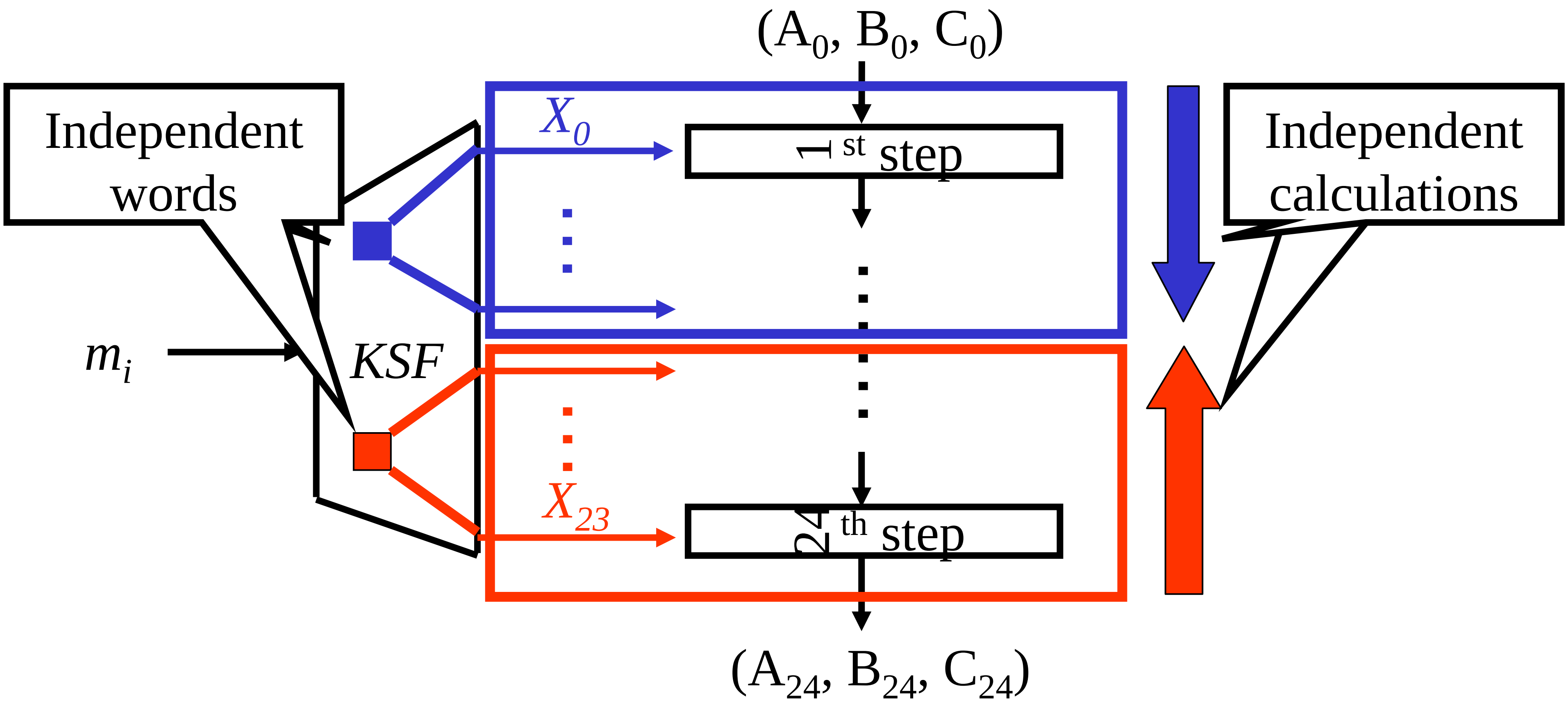
- $X_i \oplus C_i = c_7||c_6||\cdots||c_0$
- c_j: 8-bit
- t_i: a small constant
- $even = T_0(c_0) \oplus T_1(c_2) \oplus T_2(c_4) \oplus T_3(c_6)$
- $odd = T_3(c_1) \oplus T_2(c_3) \oplus T_1(c_5) \oplus T_0(c_7)$
- T_j: S-boxes mapping 8-bit values to 64-bit values

Outline

- Motivation
- Tiger hash function
- Pseudo-preimage attack on 23-step Tiger
- Conclusion

Attack scenario

- Meet-in-the-middle approach: based on weakness of *KSF*.



- Finding pseudo-preimages is enough for finding preimages.

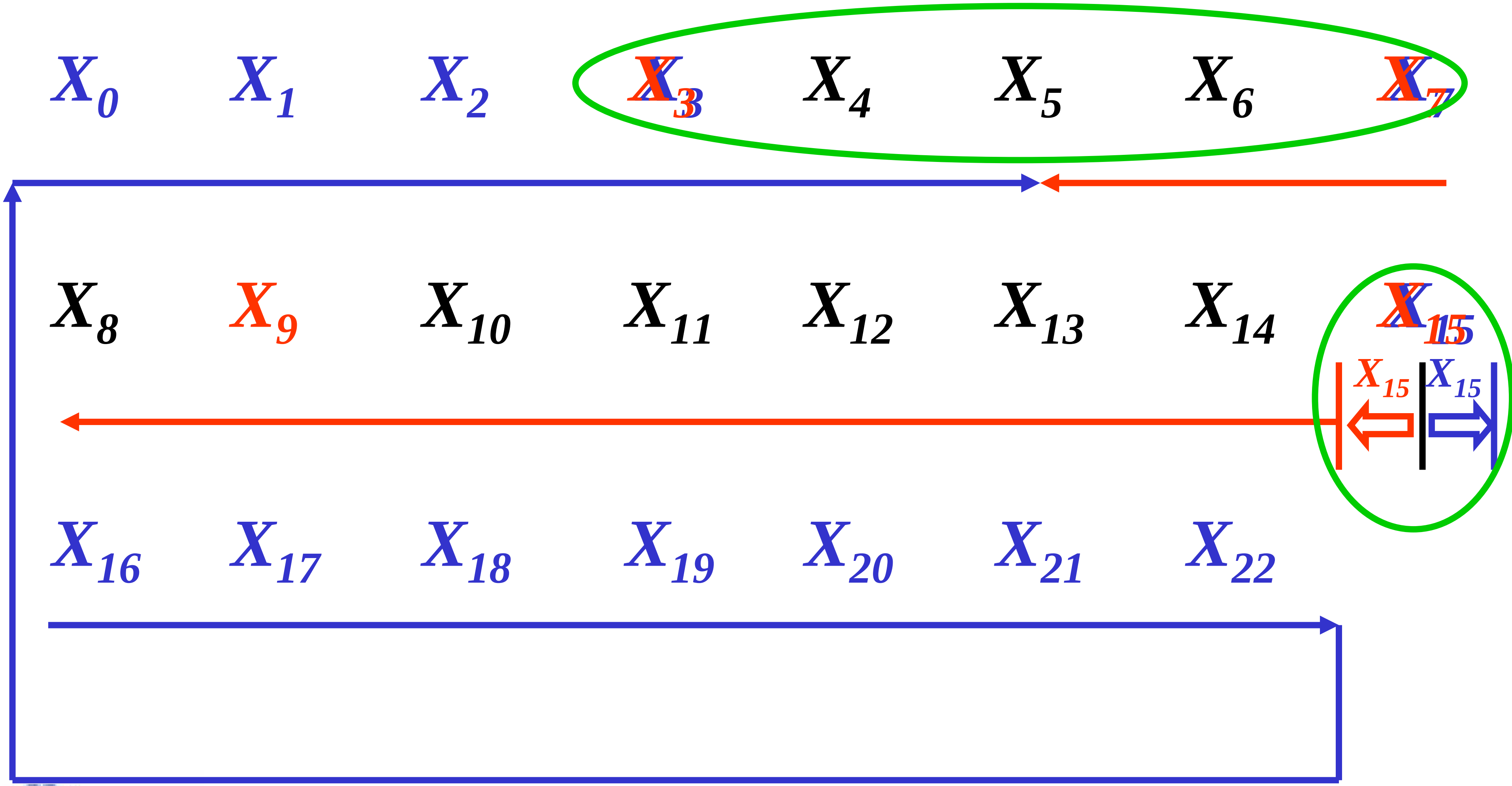
Our independent words: (X_{15}, X_{23})

◆ X_{15} changes its 11 LSBs*, and X_{23} changes its 19 MSBs.

X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7
X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
X_{16}	X_{17}	X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	X_{23}

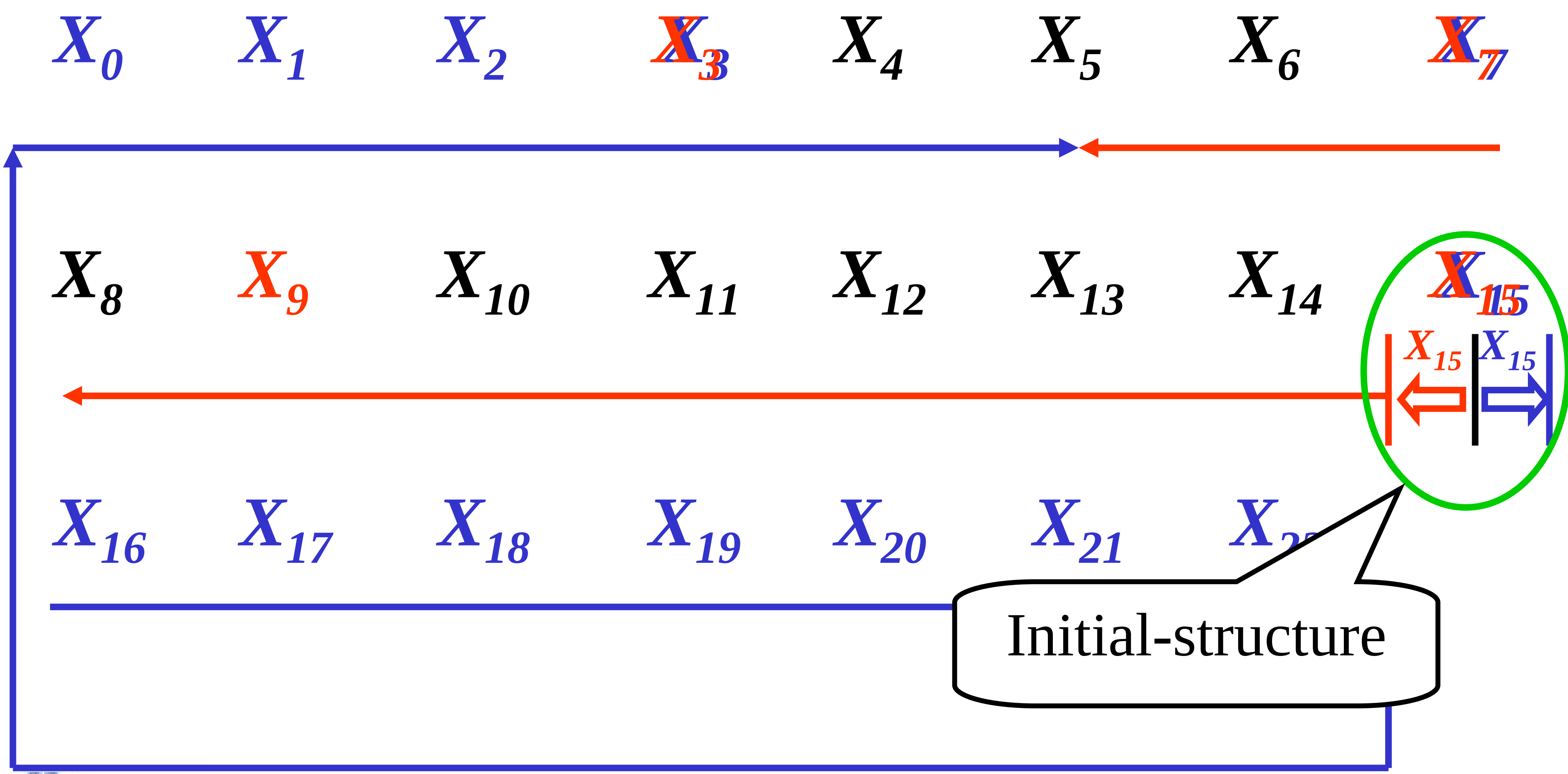
Overview of our attack

◆ We use message words to represent the corresponding step function.



Overview of our attack

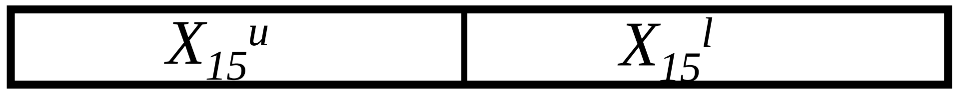
◆ We use message words to represent the corresponding step function.

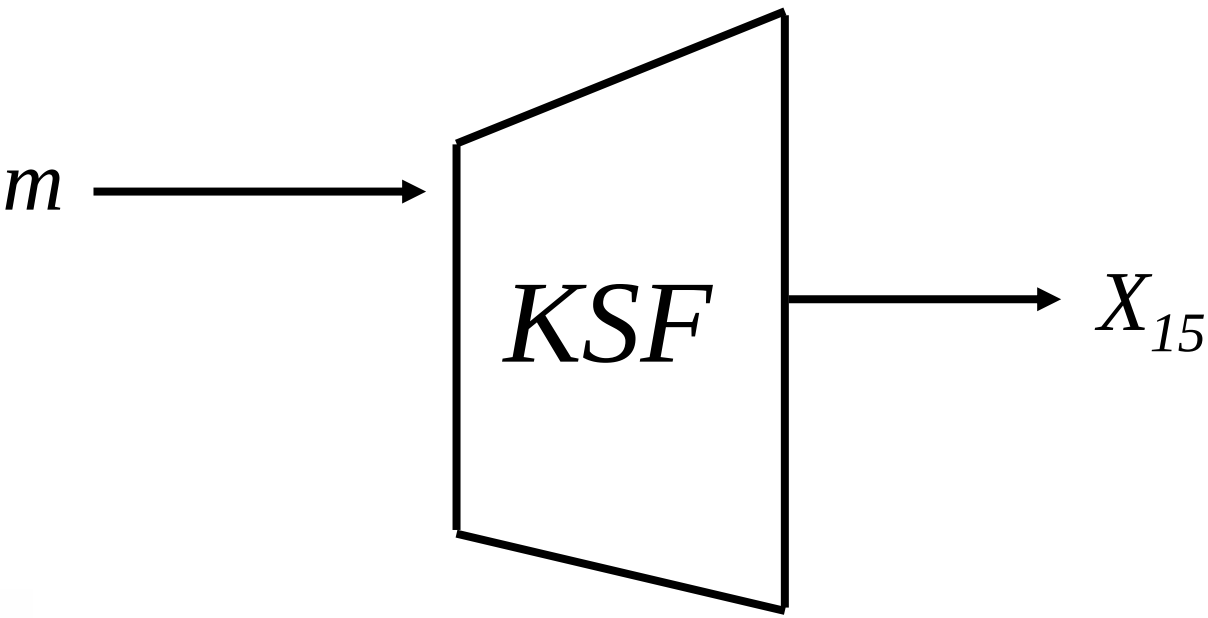


Split X_{15} into independent X_{15} and X_{15}

- Split X_{15} into upper and lower halves

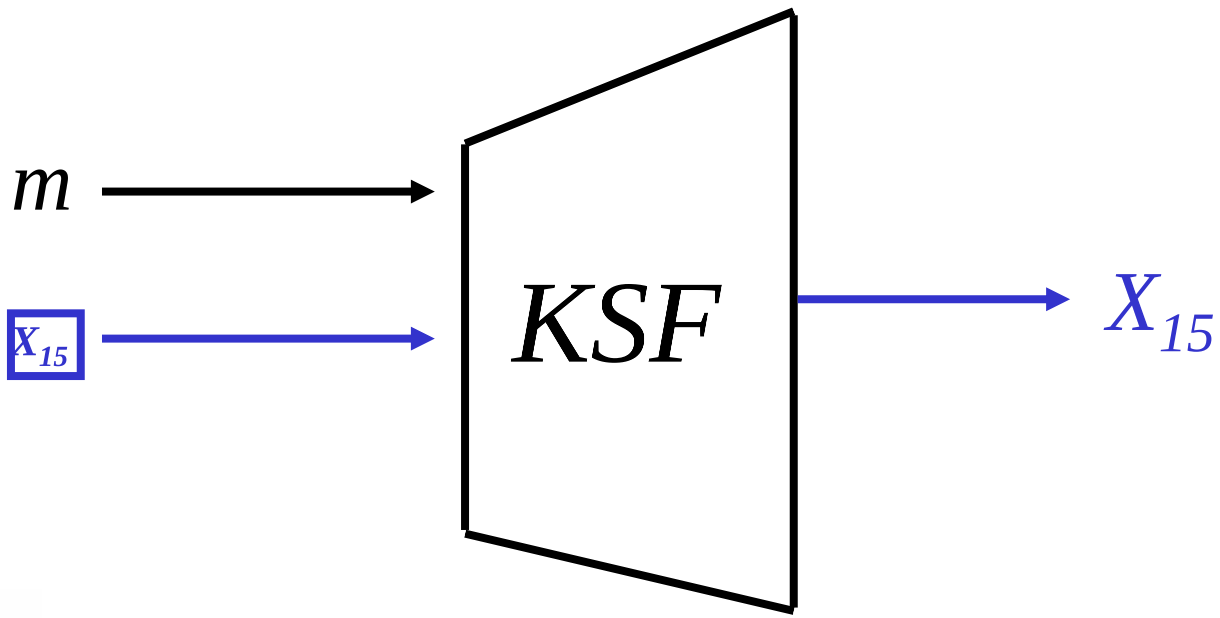
Original X_{15}





Split X_{15} into independent X_{15} and X_{15}

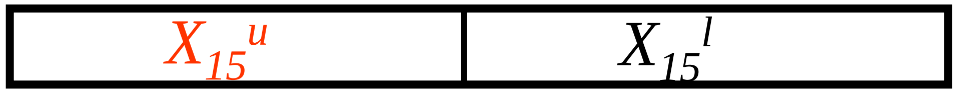
- Split X_{15} into upper and lower halves

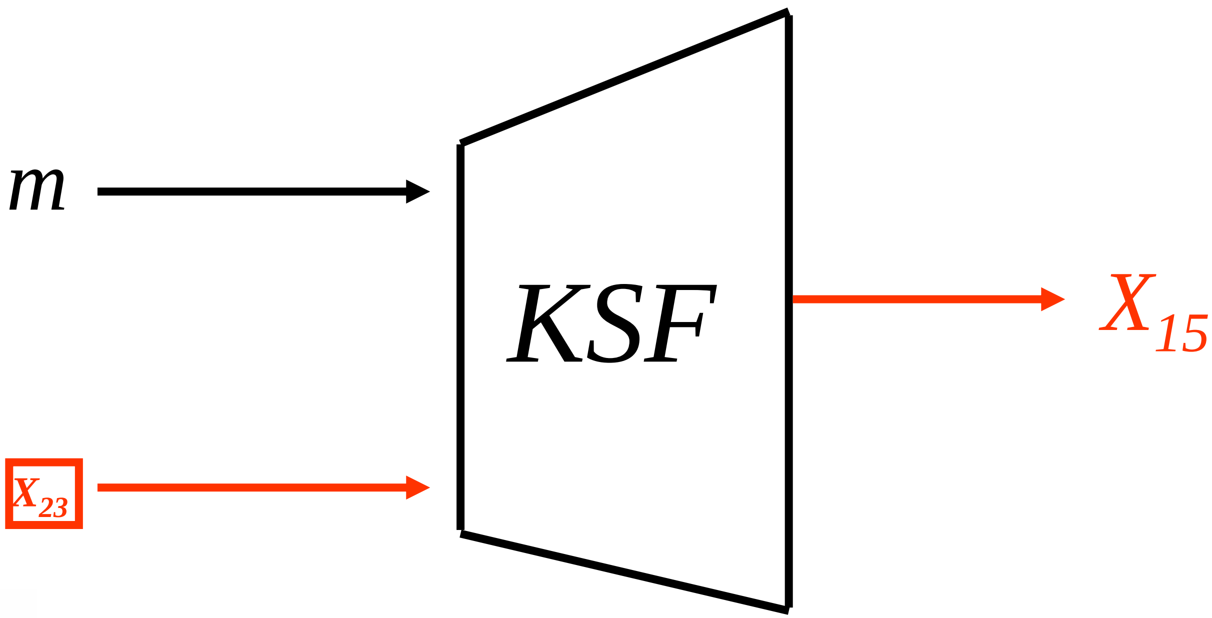


Split X_{15} into independent X_{15} and X_{15}

- Split X_{15} into upper and lower halves

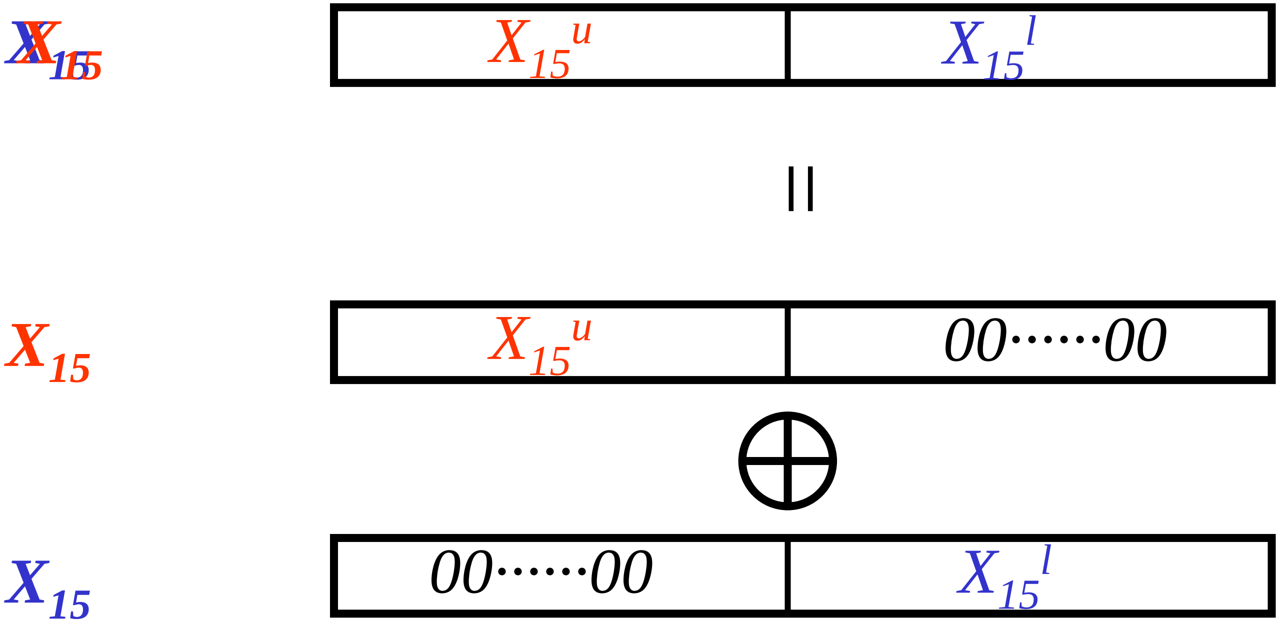
X_{15}



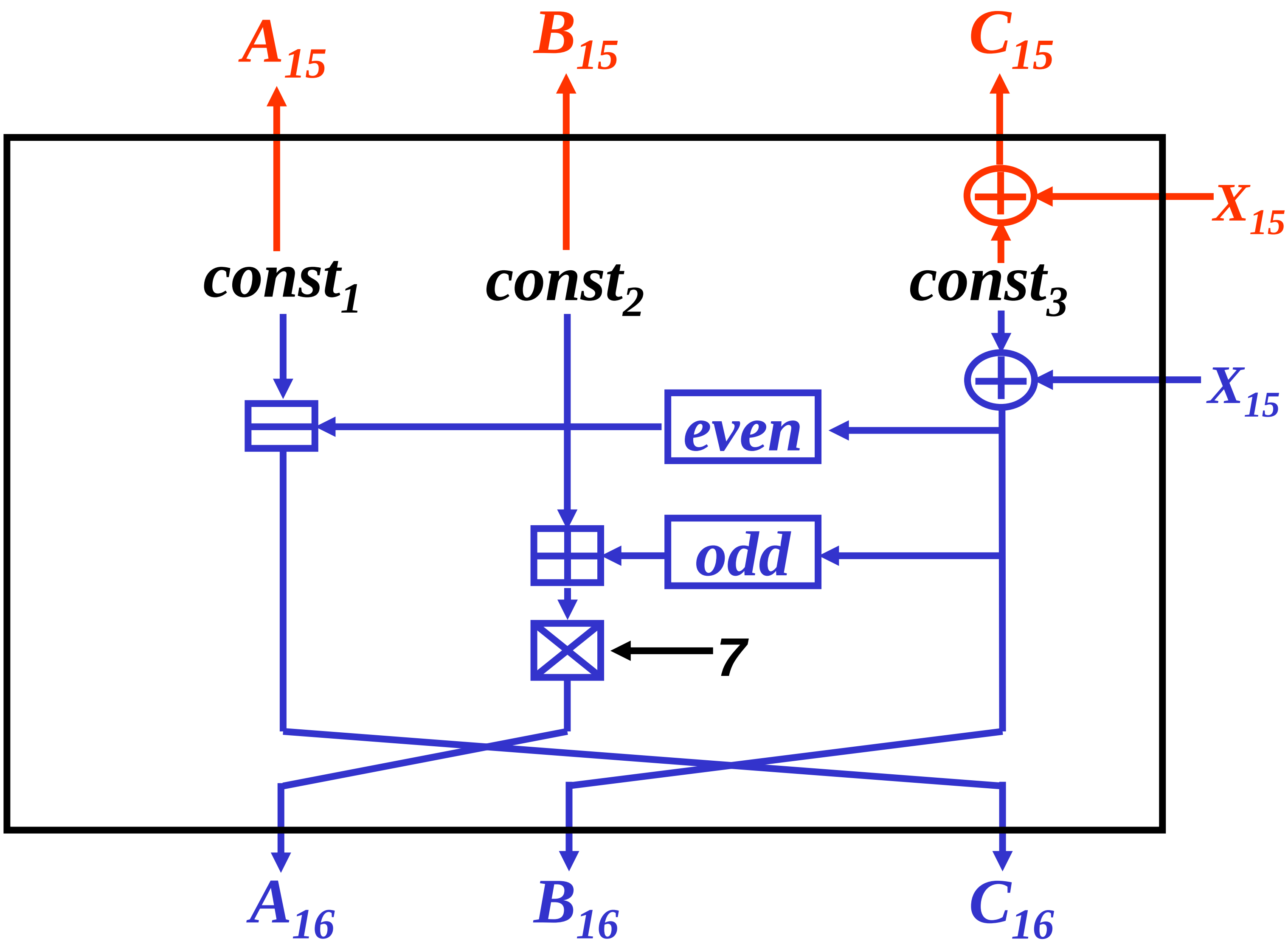


Split X_{15} into independent X_{15} and X_{15}

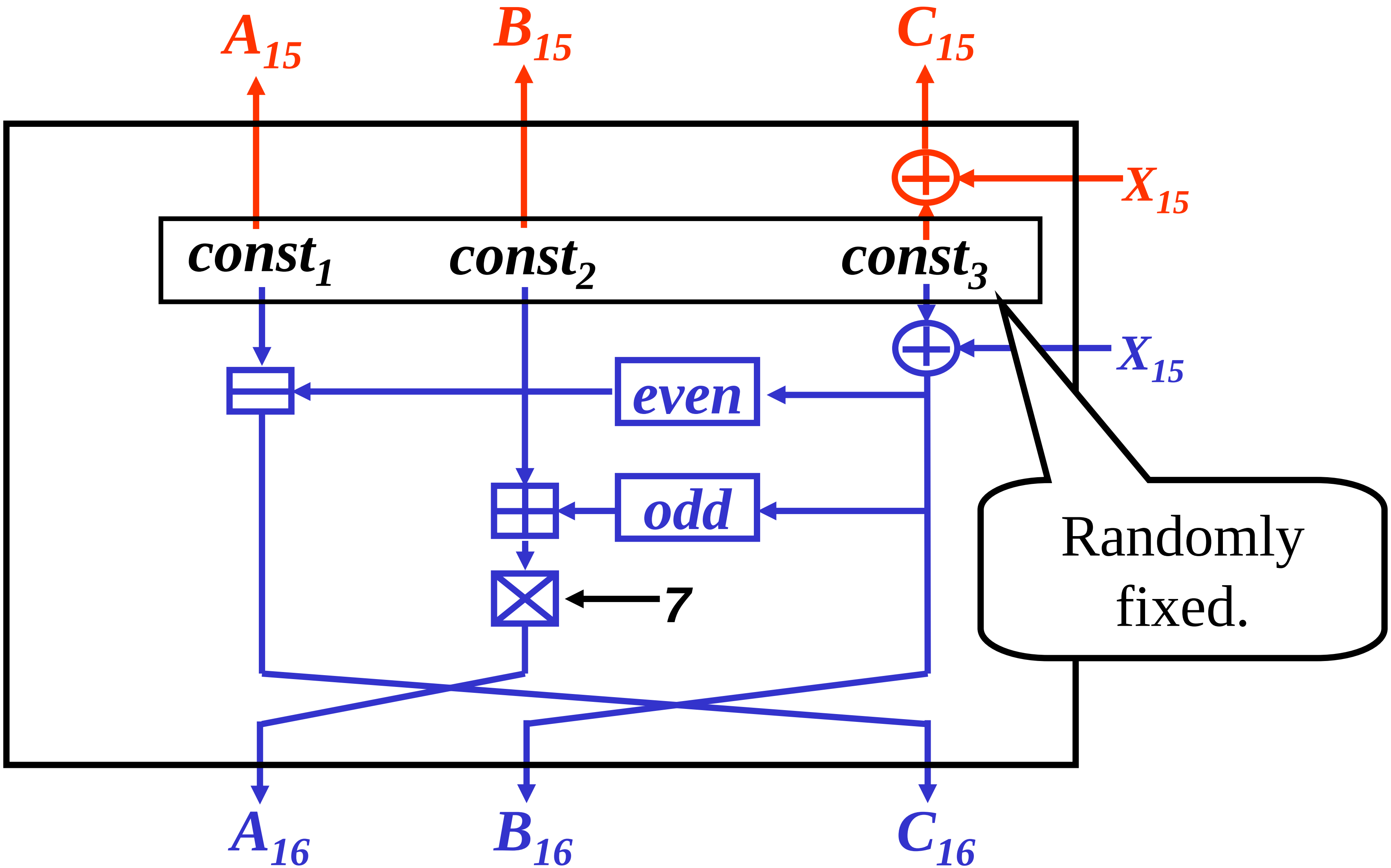
- Split X_{15} into upper and lower halves



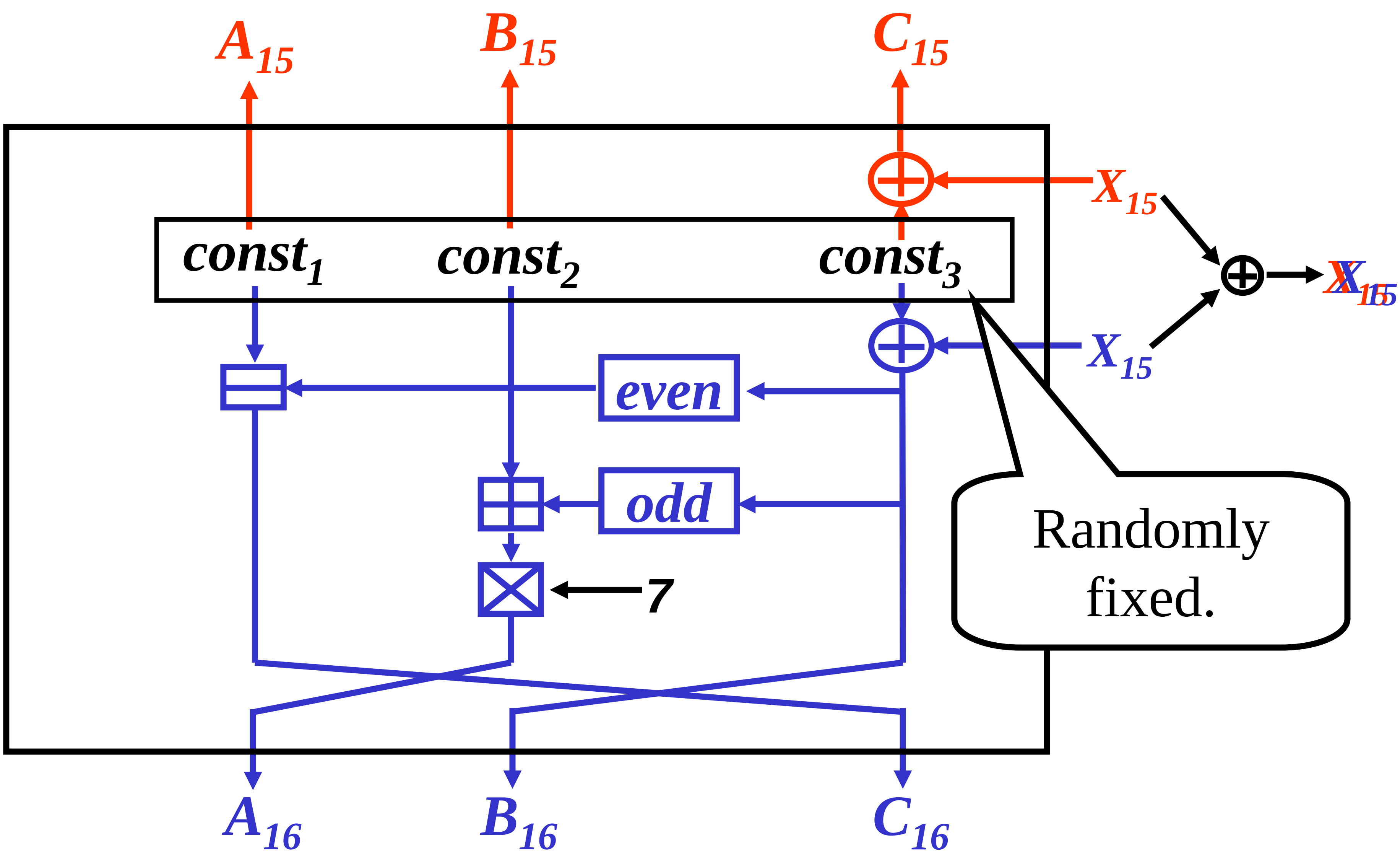
Initial structure at step 16



Initial structure at step 16

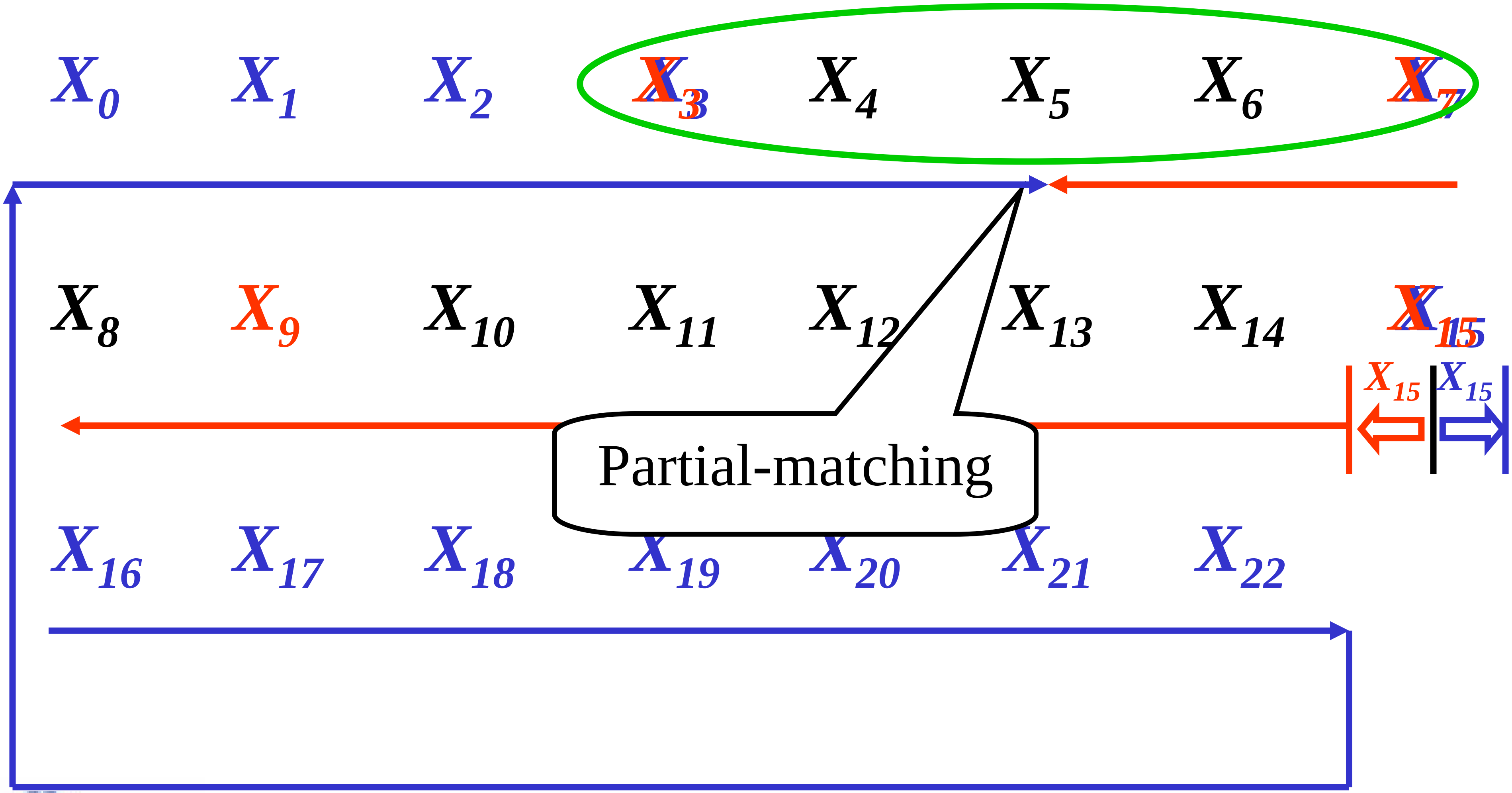


Initial structure at step 16



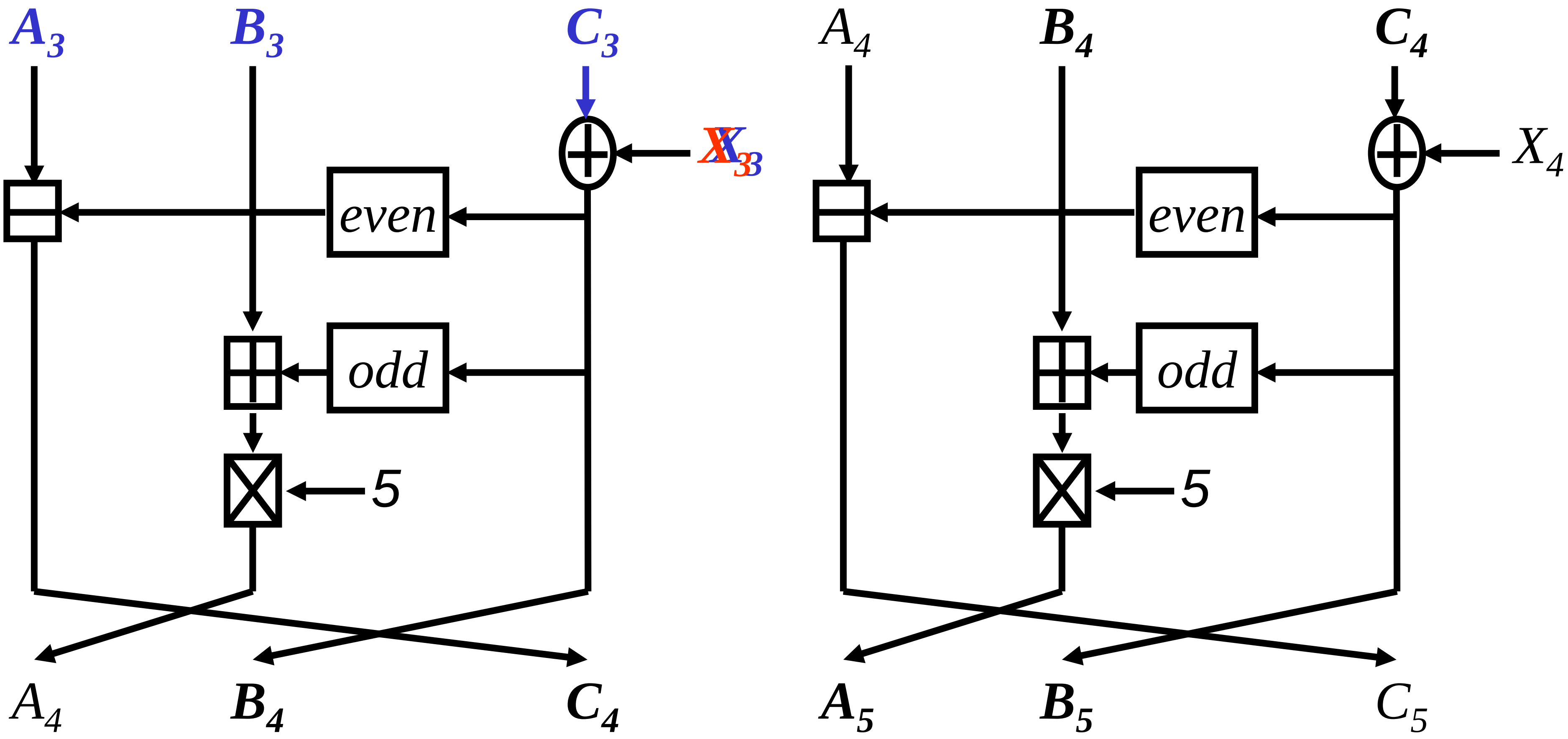
Overview of our attack

◆ We use message words to represent the corresponding step function.



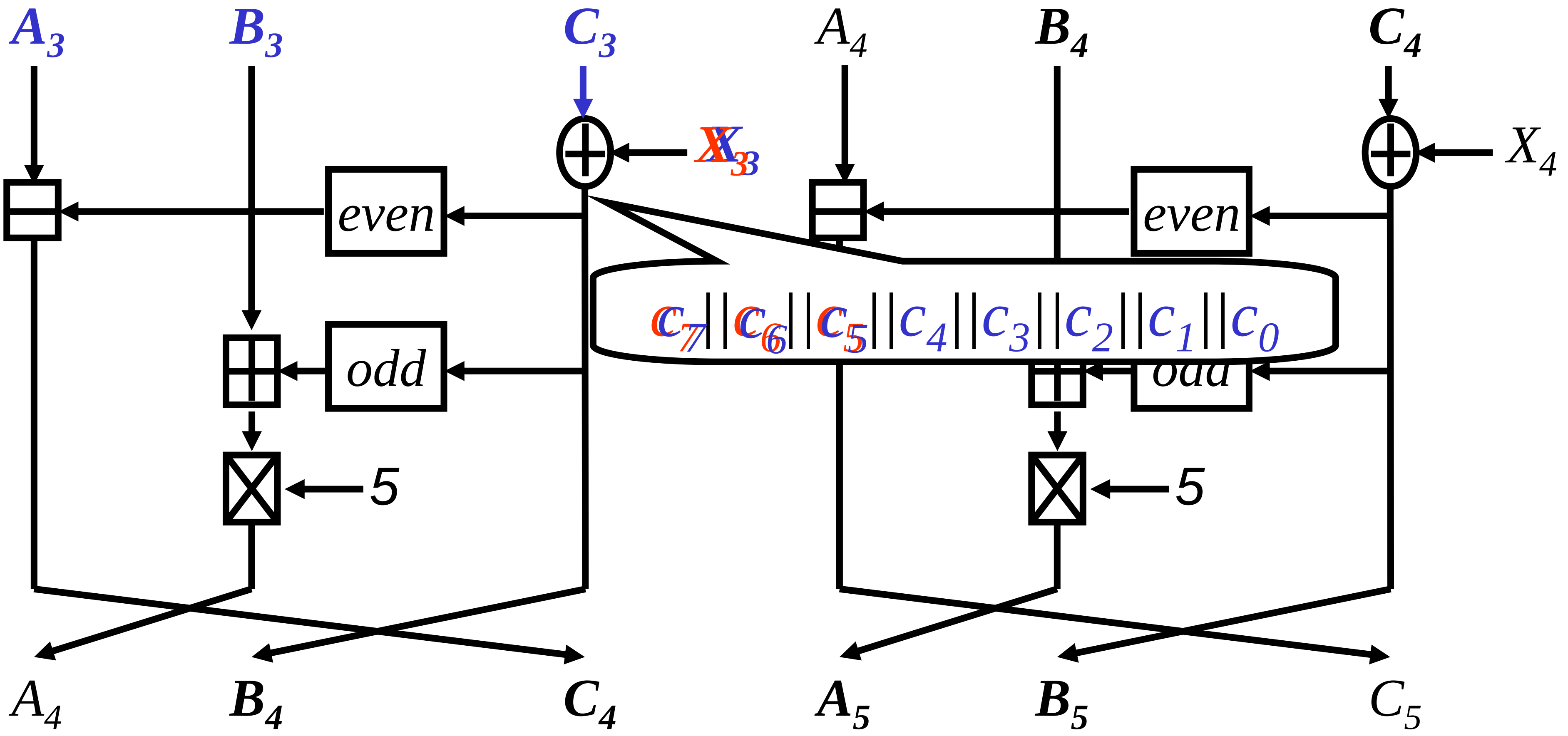
Calculation from step 4 to 5

- Note that red word only change 19 MSBs of X_3.



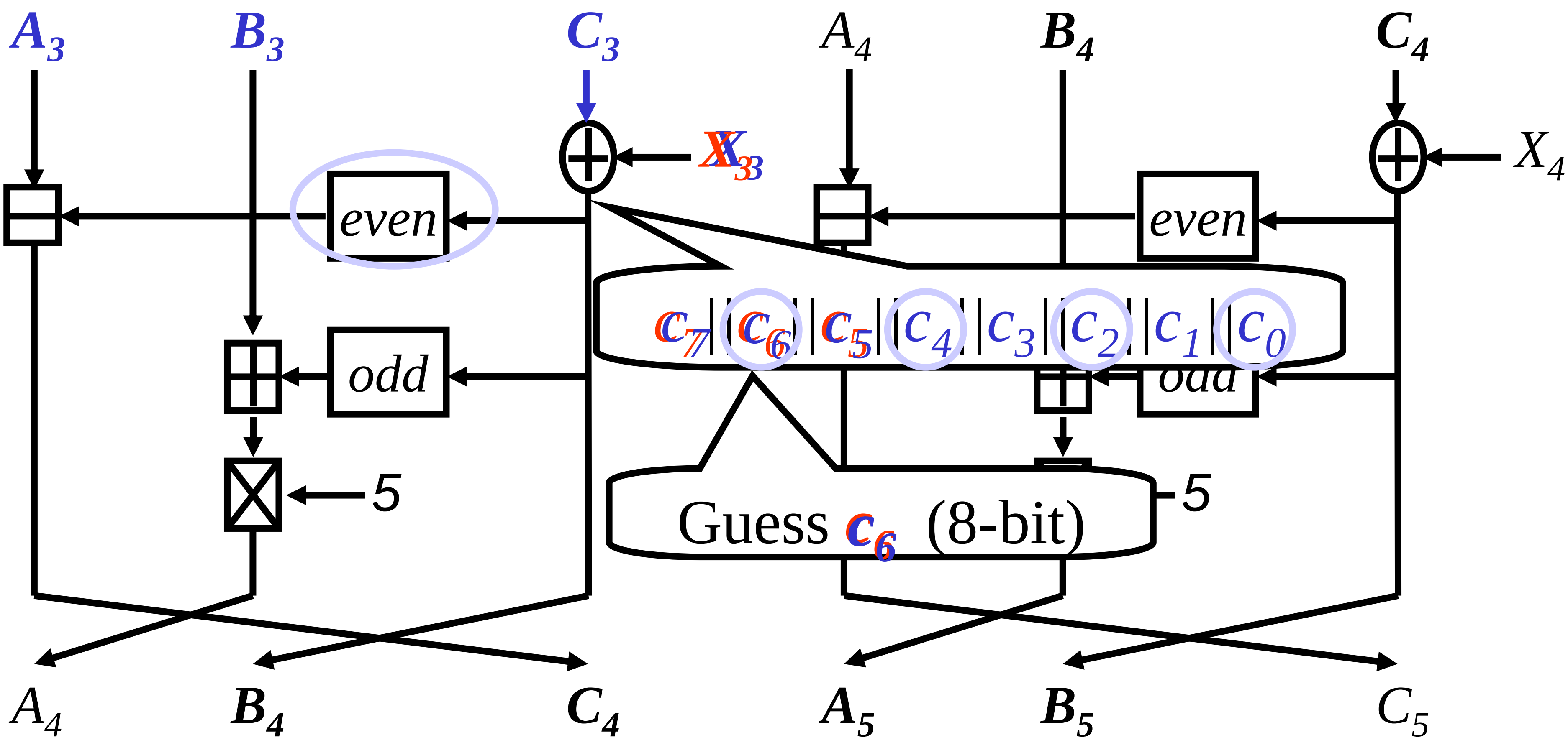
Calculation from step 4 to 5

- Note that red word only change 19 MSBs of X_3.



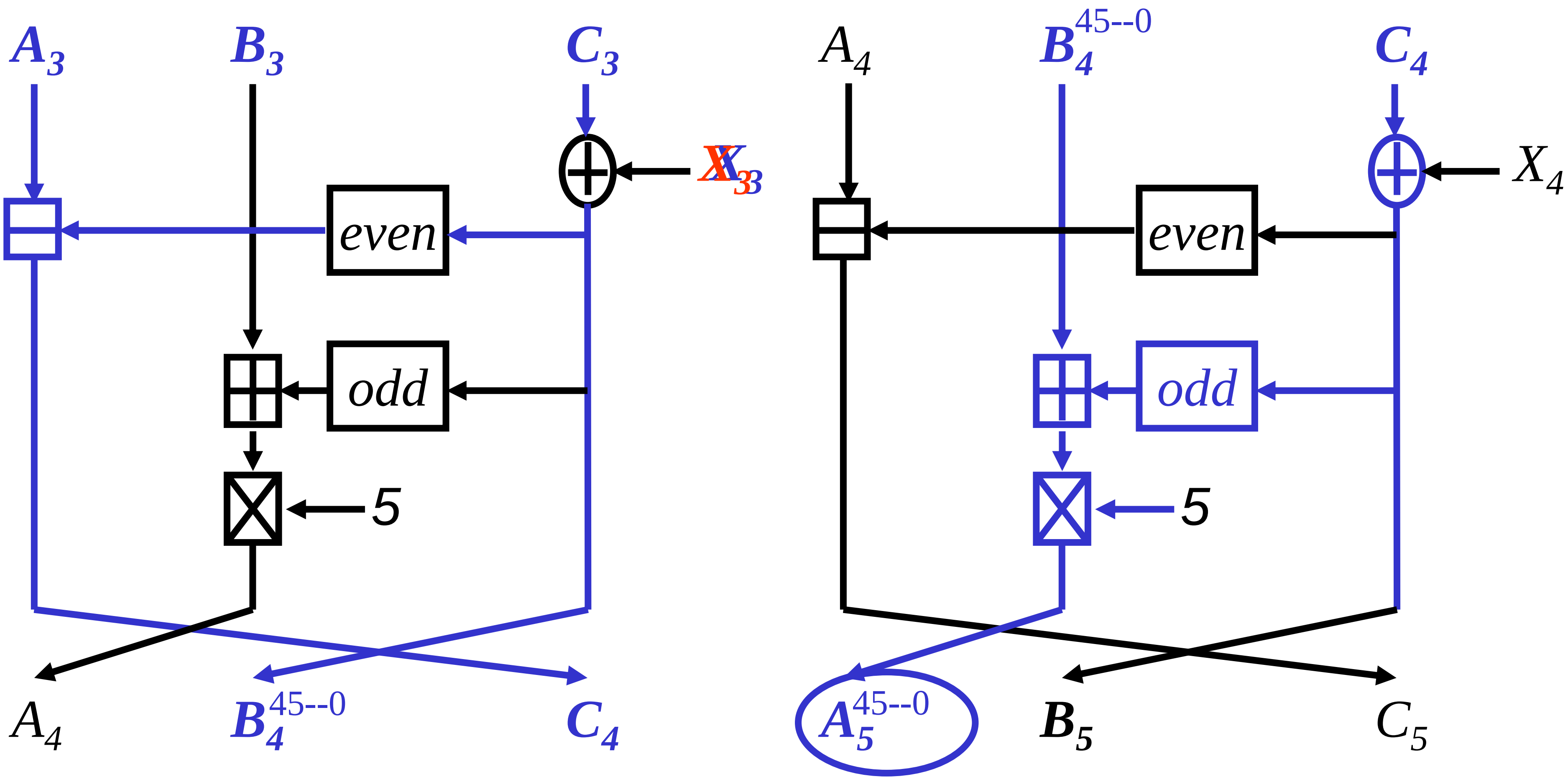
Calculation from step 4 to 5

- Note that red word only change 19 MSBs of X_3.

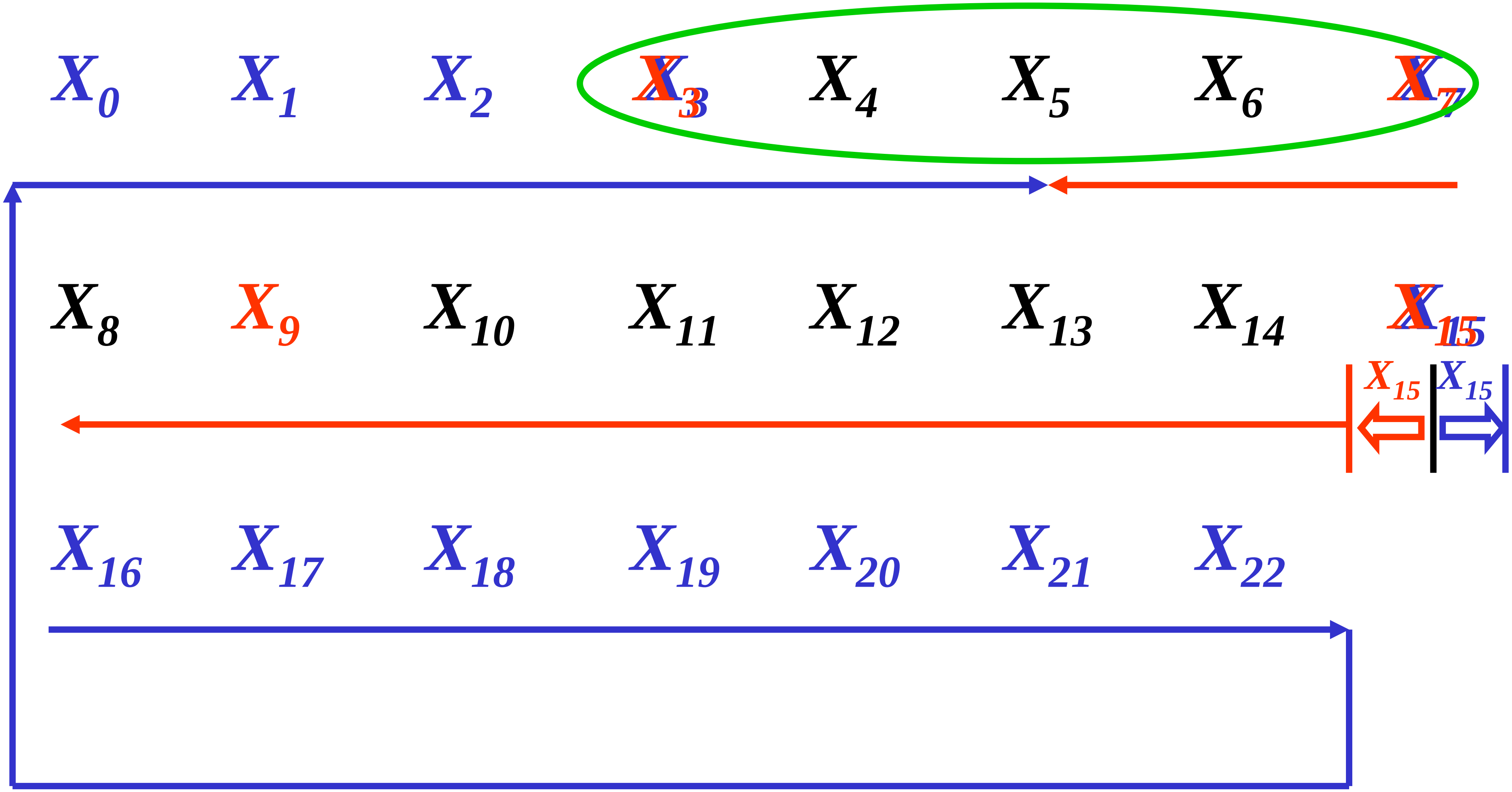


Calculation from step 4 to 5

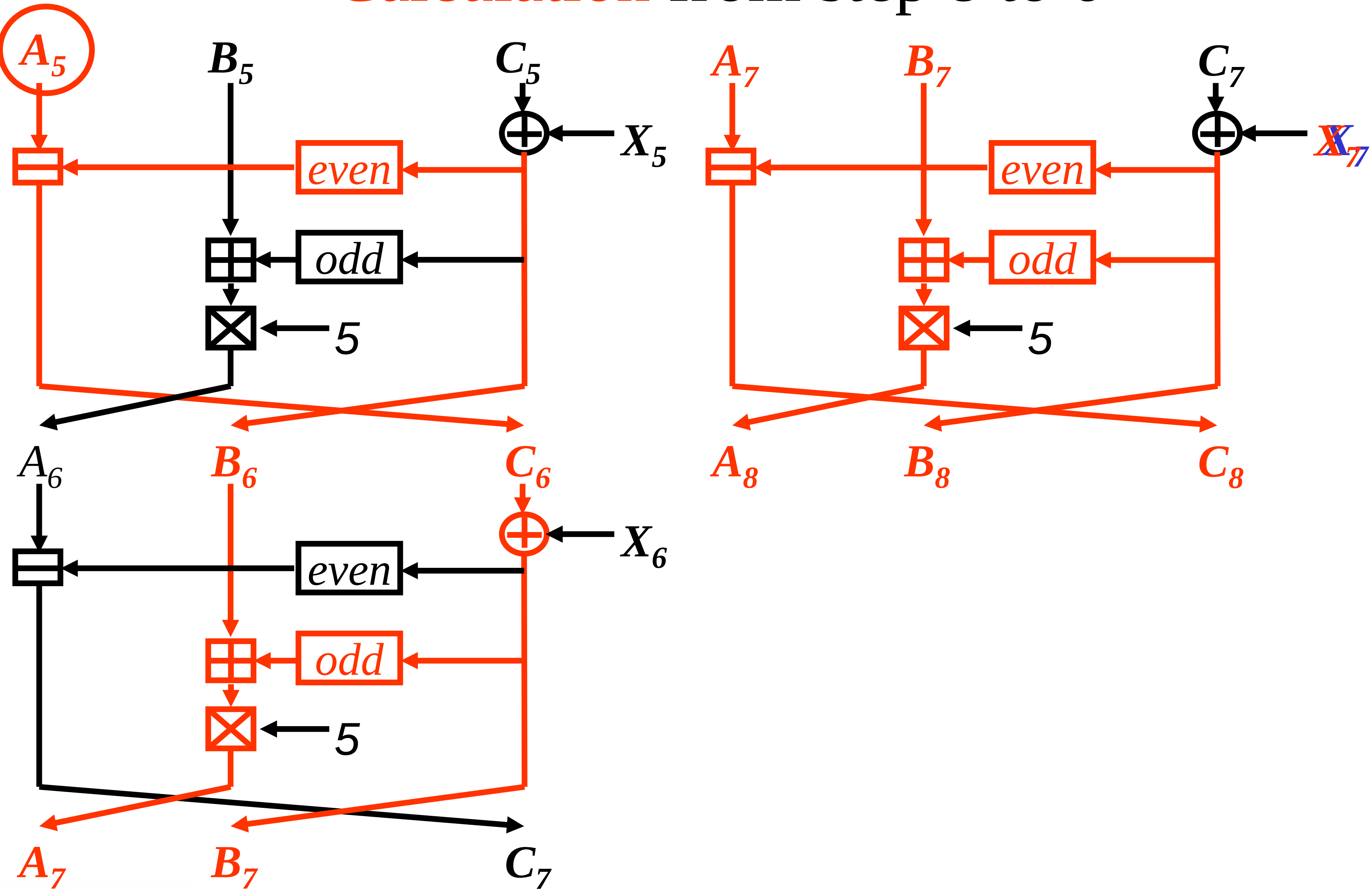
- Note that red word only change 19 MSBs of X_3.



Overview of our attack

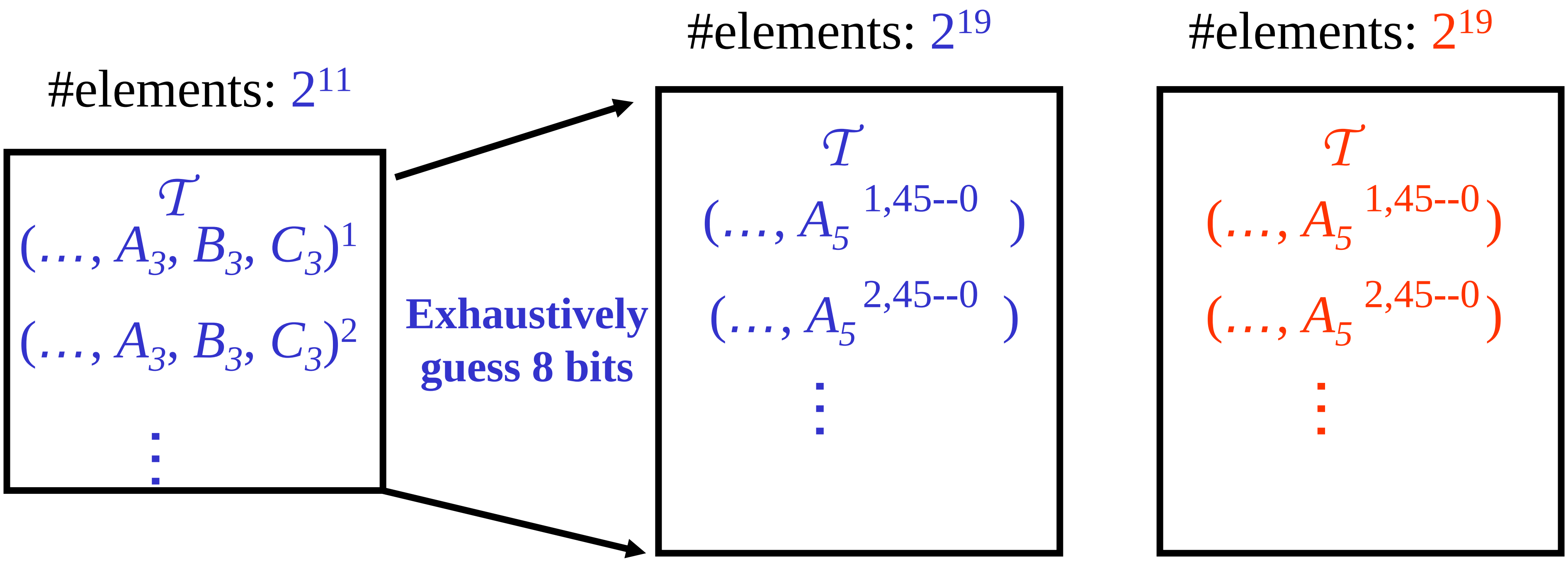


Calculation from step 8 to 6



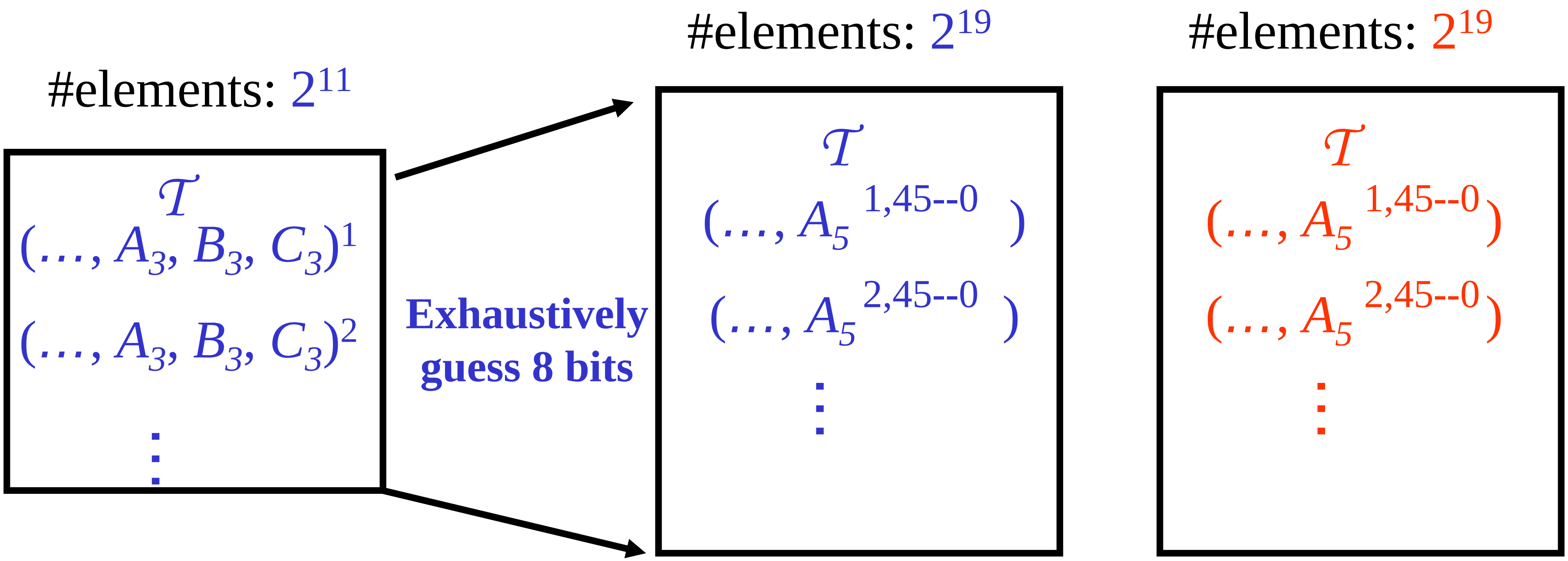
Evaluating the complexity

◆ Recall that X_{15} changes its 11 LSBs and X_{23} changes its 19 MSBs.



Evaluating the complexity

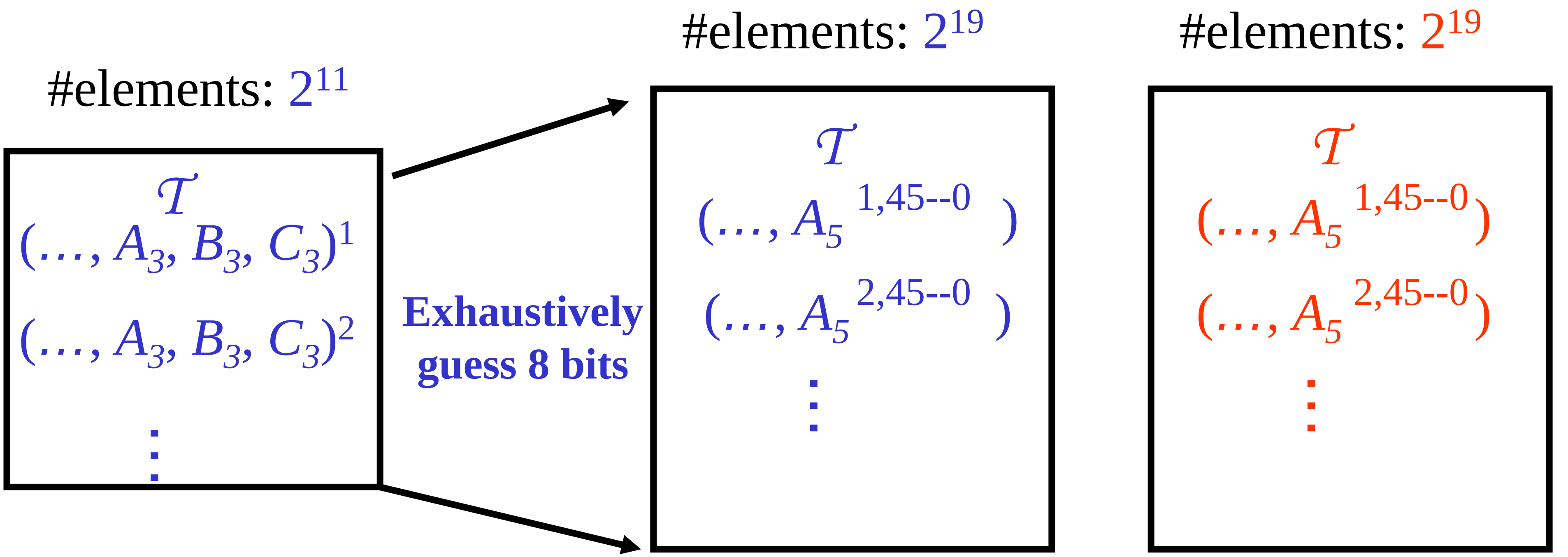
◆ Recall that X_{15} changes its 11 LSBs and X_{23} changes its 19 MSBs.



◆ 2^{19} tiger computations will contribute to 2^{38} pairs.

Evaluating the complexity

- Recall that X_{15} changes its 11 LSBs and X_{23} changes its 19 MSBs.



- 2^{19} tiger computations will contribute to 2^{38} pairs.
- For each pair, the success probability of guess is 2^{-8}.

Evaluating the complexity

- Recall that X_{15} changes its 11 LSBs and X_{23} changes its 19 MSBs.

#elements: 2^{11}

$\mathcal{T}$

$(\ldots, A_3, B_3, C_3)^1$

#elements: 2^{19}

$\mathcal{T}$

$(\ldots, A_5^{\ 1,45\text{--}0})$

#elements: 2^{19}

$\mathcal{T}$

$(\ldots, A_5^{\ 1,45\text{--}0})$

Finally the complexity of finding a pseudo-preimage is 2^{181} ($2^{192-19+8}$).

- 2^{19} tiger computations will contribute to 2^{38} pairs.
- For each pair, the success probability of guess is 2^{-8}.

Note on preimage attacks

- We use the generic conversion from pseudo-preimages to preimages.

  → (2nd) preimages with a complexity of $2^{187.5}$.

- If padding is considered, message freedom in an independent chunk is reduced by 9 bits.

  → preimages with a complexity of 1.4×2^{189}.

Outline

- Motivation
- Tiger hash function
- Pseudo-preimage attack on 23-step Tiger
- Conclusion

Conclusion

◆ We have found a preimage attack on 23-step Tiger based on the recently developed MitM approach.

◆ The complexity of our attack is as follows:

	Time	Memory
Preimages	1.4×2^{189}	2^{22}
2^{nd} Preimages	$2^{187.5}$	2^{22}

Thank you for your attention !!