Key Committing Attacks against AES-based AEAD Schemes

Patrick Derbez[1], Pierre-Alain Fouque[1], Takanori Isobe[2], **Mostafizar Rahman**[2], André Schrottenloher[1]

[1]Univ Rennes, Inria, CNRS, IRISA, Rennes, France
[2]University of Hyogo, Kobe, Japan

FSE 2024
Leuven, Belgium

Introduction

- Traditional focus of AEAD- Confidentiality and Integrity
- Sometimes they are not sufficient
- Attack on facebook message franking
- Additional Requirement: A ciphertext can only be decrypted using the original key (Key Commitment)

Background

- Aegis was considered as a fully committing scheme [1]
- In earlier draft of IETF, it was also considered as key committing [2]
- Attacking the key committing security of Aegis, also acknowledged as an open problem [3]

[1] John Preuß Mattsson, Ben Smeets, and Erik Thormarker. Proposals for Standardization of Encryption Schemes.

[2] Frank Denis and Samuel Lucas. The AEGIS Family of Authenticated Encryption Algorithms.

[3] Stefan Kölbl. Open Questions around Key Committing AEADs. https://frisiacrypt2022.cs.ru.nl/assets/slides/stefan-frisiacrypt2022.pdf.

- Several notions introduced based on what the ciphertext is committed to-
  - FROB Game
  - CMT-1 Game
  - CMT-3 Game
  - CMT-4 Game

Committing Authenticated Encryption

1. $(C, (K_1, N_1, AD_1), (K_2, N_2, AD_2)) \xleftarrow{\$} \mathcal{A}$
2. $P_1 \leftarrow \Sigma_{Dec}(K_1, N_1, AD_1, C)$
3. $P_2 \leftarrow \Sigma_{Dec}(K_2, N_2, AD_2, C)$
4. If $P_1 = \perp$ or $P_2 = \perp$ then
   Return false
5. **If** $K_1 = K_2$ or $N_1 \neq N_2$ **then**
   Return false
6. Return true

Committing Authenticated Encryption

1. $(C, (K_1, N_1, AD_1), (K_2, N_2, AD_2)) \xleftarrow{\$} \mathcal{A}$
2. $P_1 \leftarrow \Sigma_{Dec}(K_1, N_1, AD_1, C)$
3. $P_2 \leftarrow \Sigma_{Dec}(K_2, N_2, AD_2, C)$
4. If $P_1 = \perp$ or $P_2 = \perp$ then
   Return false
5. **If** $K_1 = K_2$ or $N_1 \neq N_2$ **then**
   Return false
6. Return true

Committing Authenticated Encryption

1. $(C, (K_1, N_1, AD_1), (K_2, N_2, AD_2)) \xleftarrow{\$} \mathcal{A}$
2. $P_1 \leftarrow \Sigma_{Dec}(K_1, N_1, AD_1, C)$
3. $P_2 \leftarrow \Sigma_{Dec}(K_2, N_2, AD_2, C)$
4. If $P_1 = \perp$ or $P_2 = \perp$ then
   Return false
5. **If** $K_1 = K_2$ or $N_1 \neq N_2$ **then**
   Return false
6. Return true

Committing Authenticated Encryption

1. $(C, (K_1, N_1, AD_1), (K_2, N_2, AD_2)) \xleftarrow{\$} \mathcal{A}$
2. $P_1 \leftarrow \Sigma_{Dec}(K_1, N_1, AD_1, C)$
3. $P_2 \leftarrow \Sigma_{Dec}(K_2, N_2, AD_2, C)$
4. If $P_1 = \perp$ or $P_2 = \perp$ then
   Return false
5. **If** $K_1 = K_2$ or $N_1 \neq N_2$ **then**
   Return false
6. Return true

Committing Authenticated Encryption

1. $(C, (K_1, N_1, AD_1), (K_2, N_2, AD_2)) \xleftarrow{\$} \mathcal{A}$
2. $P_1 \leftarrow \Sigma_{Dec}(K_1, N_1, AD_1, C)$
3. $P_2 \leftarrow \Sigma_{Dec}(K_2, N_2, AD_2, C)$
4. If $P_1 = \perp$ or $P_2 = \perp$ then
   Return false
5. **If** $K_1 = K_2$ or $N_1 \neq N_2$ **then**
   Return false
6. Return true

Adversary is required to produce same ciphertext with different key and same nonce

Committing Authenticated Encryption

1. $(C, (K_1, N_1, AD_1), (K_2, N_2, AD_2)) \xleftarrow{\$} \mathcal{A}$
2. $P_1 \leftarrow \Sigma_{Dec}(K_1, N_1, AD_1, C)$
3. $P_2 \leftarrow \Sigma_{Dec}(K_2, N_2, AD_2, C)$
4. If $P_1 = \perp$ or $P_2 = \perp$ then
   Return false
5. **If** $K_1 = K_2$ **then**
   Return false
6. Return true

Adversary is required to produce same ciphertext with different key

Committing Authenticated Encryption

1. $(C, (K_1, N_1, AD_1), (K_2, N_2, AD_2)) \xleftarrow{\$} \mathcal{A}$
2. $P_1 \leftarrow \Sigma_{Dec}(K_1, N_1, AD_1, C)$
3. $P_2 \leftarrow \Sigma_{Dec}(K_2, N_2, AD_2, C)$
4. If $P_1 = \perp$ or $P_2 = \perp$ then
   Return false
5. **If** $(K_1, N_1, AD_1) = (K_2, N_2, AD_2)$ **then**
   Return false
6. Return true

Committing Authenticated Encryption

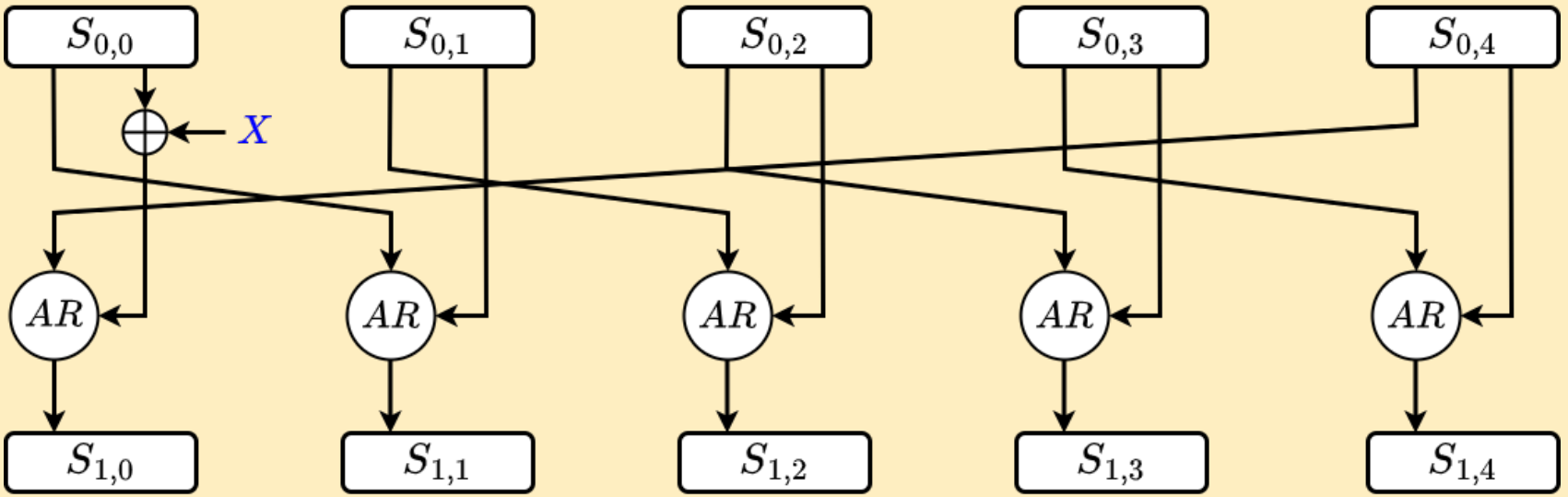
1. $(C, (K_1, N_1, AD_1), (K_2, N_2, AD_2)) \xleftarrow{\$} \mathcal{A}$
2. $P_1 \leftarrow \Sigma_{Dec}(K_1, N_1, AD_1, C)$
3. $P_2 \leftarrow \Sigma_{Dec}(K_2, N_2, AD_2, C)$
4. If $P_1 = \perp$ or $P_2 = \perp$ then
   Return false
5. **If** $(K_1, N_1, AD_1, P_1) = (K_2, N_2, AD_2, P_2)$ **then**
   Return false
6. Return true

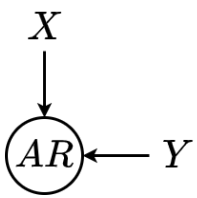
Description on Aegis

- Proposed in SAC 2013
- Three variants: Aegis-128, Aegis-256 and Aegis128L
- Aegis-128: Winner in CAESAR competition (high-performance applications)
- Four phases: *Initialization*, *Associated Data processing*, *Plaintext processing* and *Finalization*

State Update Function of Aegis

Aegis-128

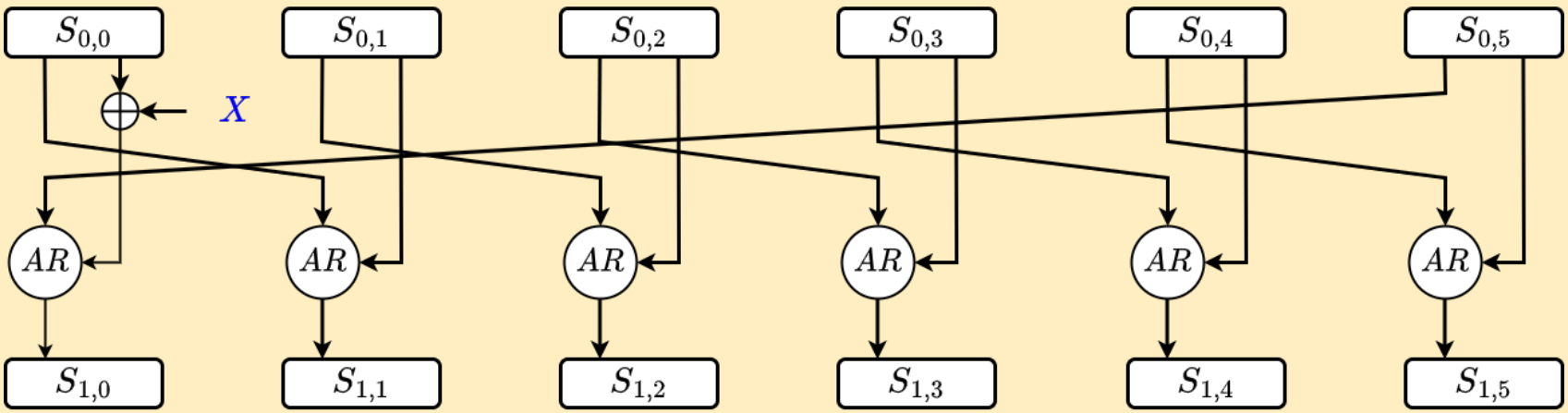


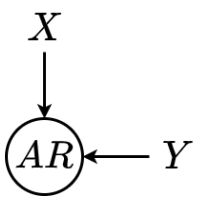


: AR: one-round AES with state X and key Y

State Update Function of Aegis

Aegis-256

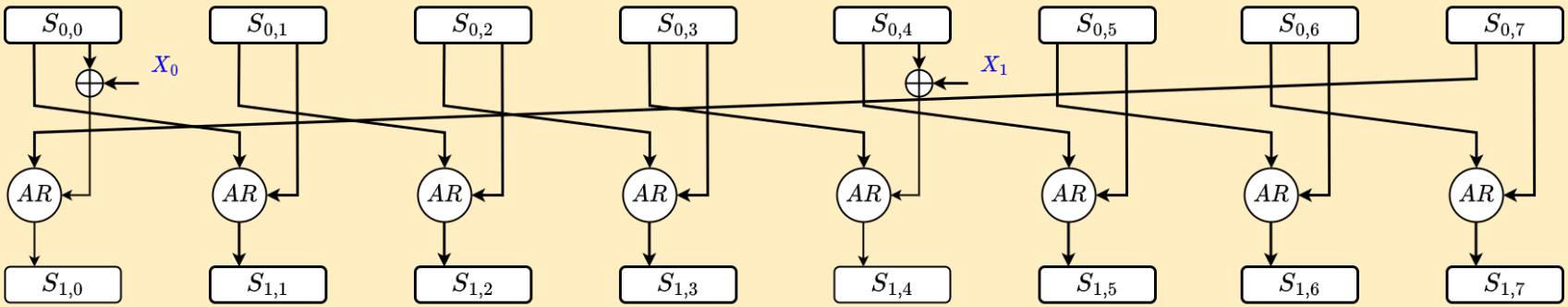


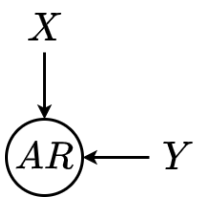


: AR: one-round AES with state X and key Y

State Update Function of Aegis

Aegis-128L



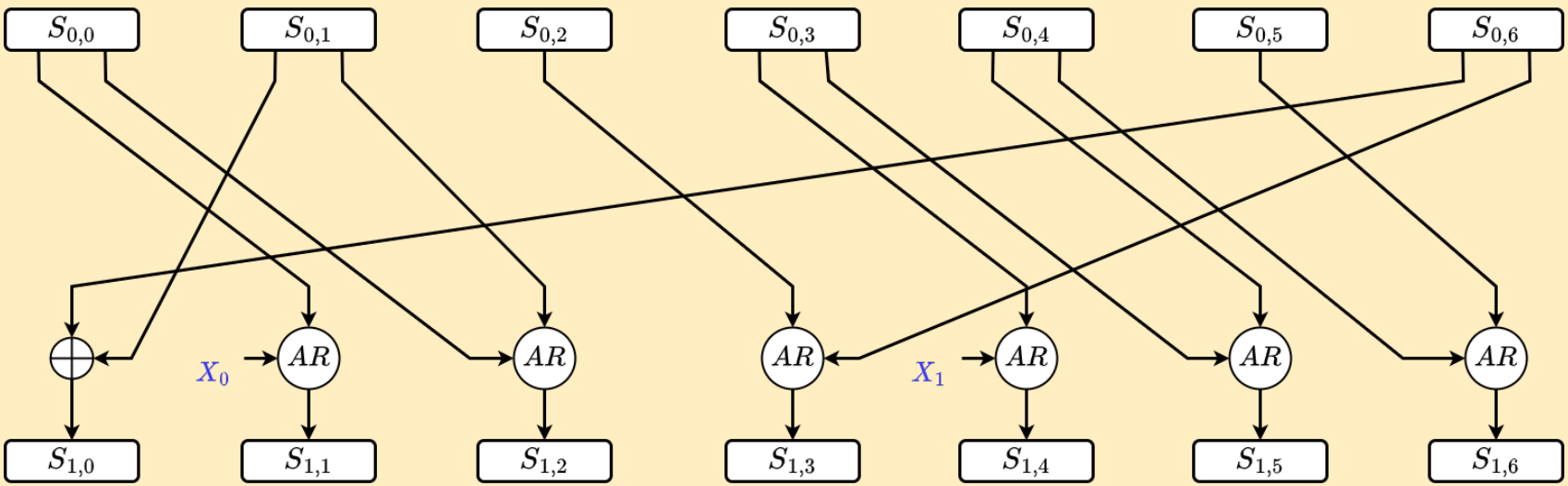


: AR: one-round AES with state X and key Y

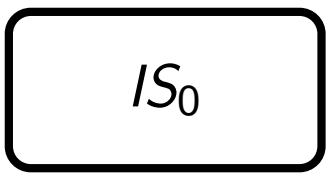
Description on Rocca-S

- Proposed in ESORICS 2023.
- Employs a 256-bit key and 128-bit nonce
- Comprises of *Initialization*, *Associated Data processing*, *Plaintext processing* and *Finalization* phases.

State Update Function

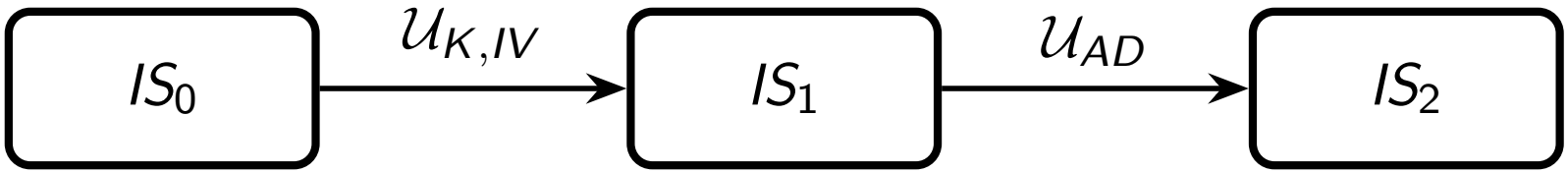


State Transitions of Aegis/Rocca-S

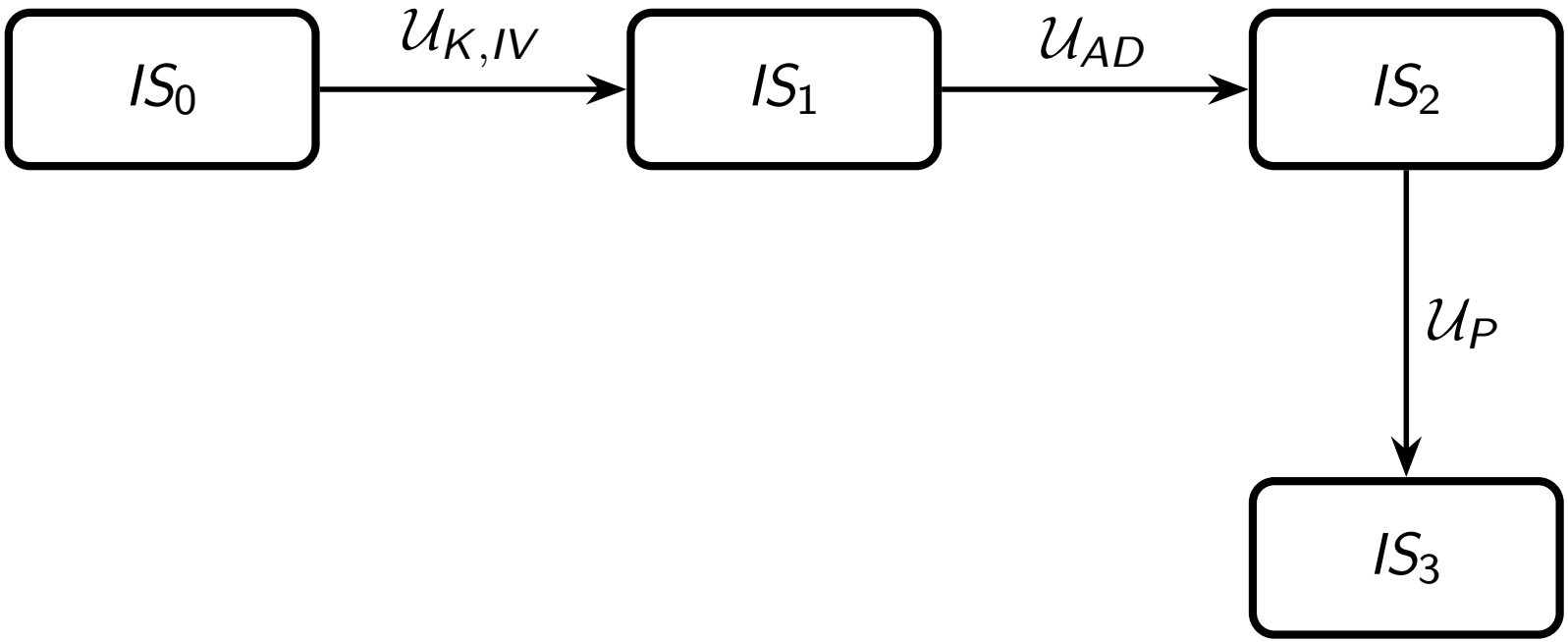


State Transitions of Aegis/Rocca-S

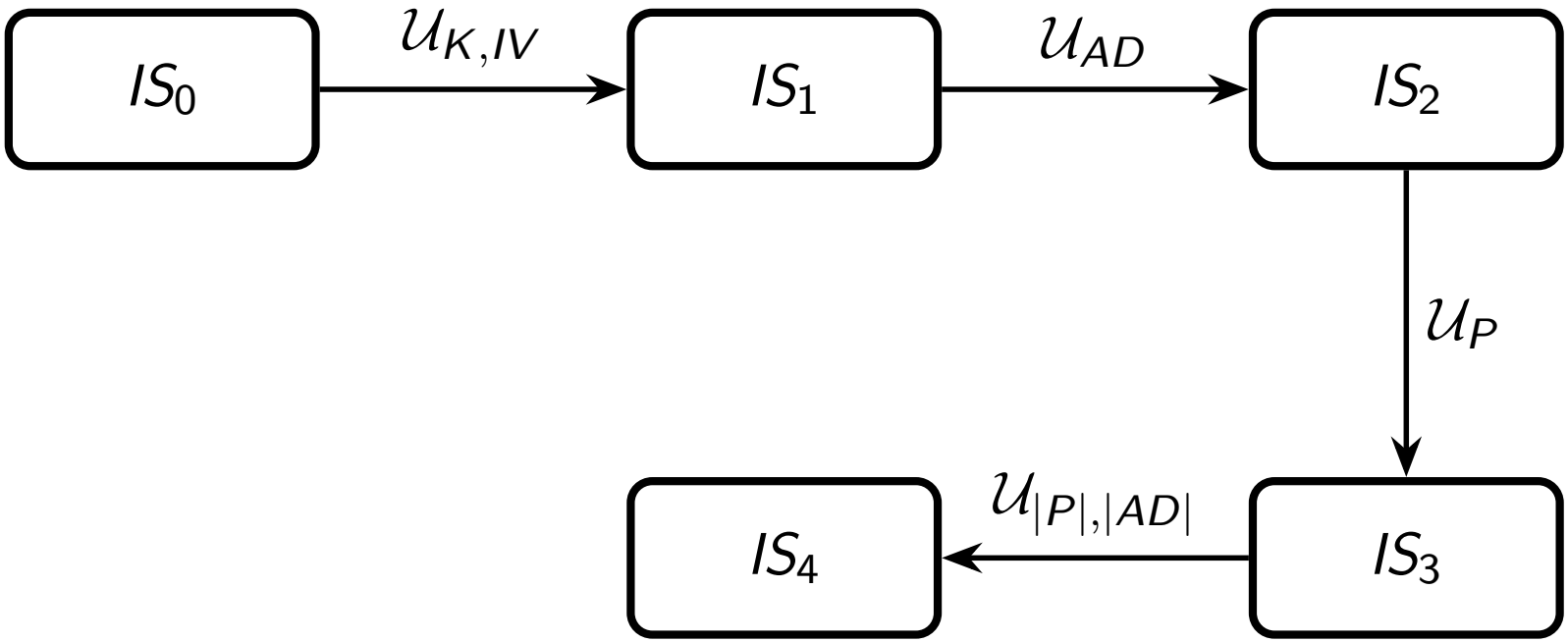
State Transitions of Aegis/Rocca-S



State Transitions of Aegis/Rocca-S

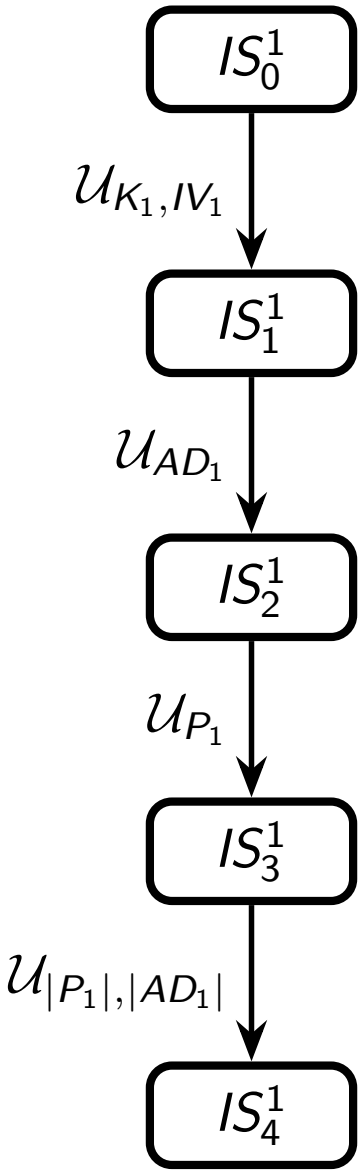


State Transitions of Aegis/Rocca-S



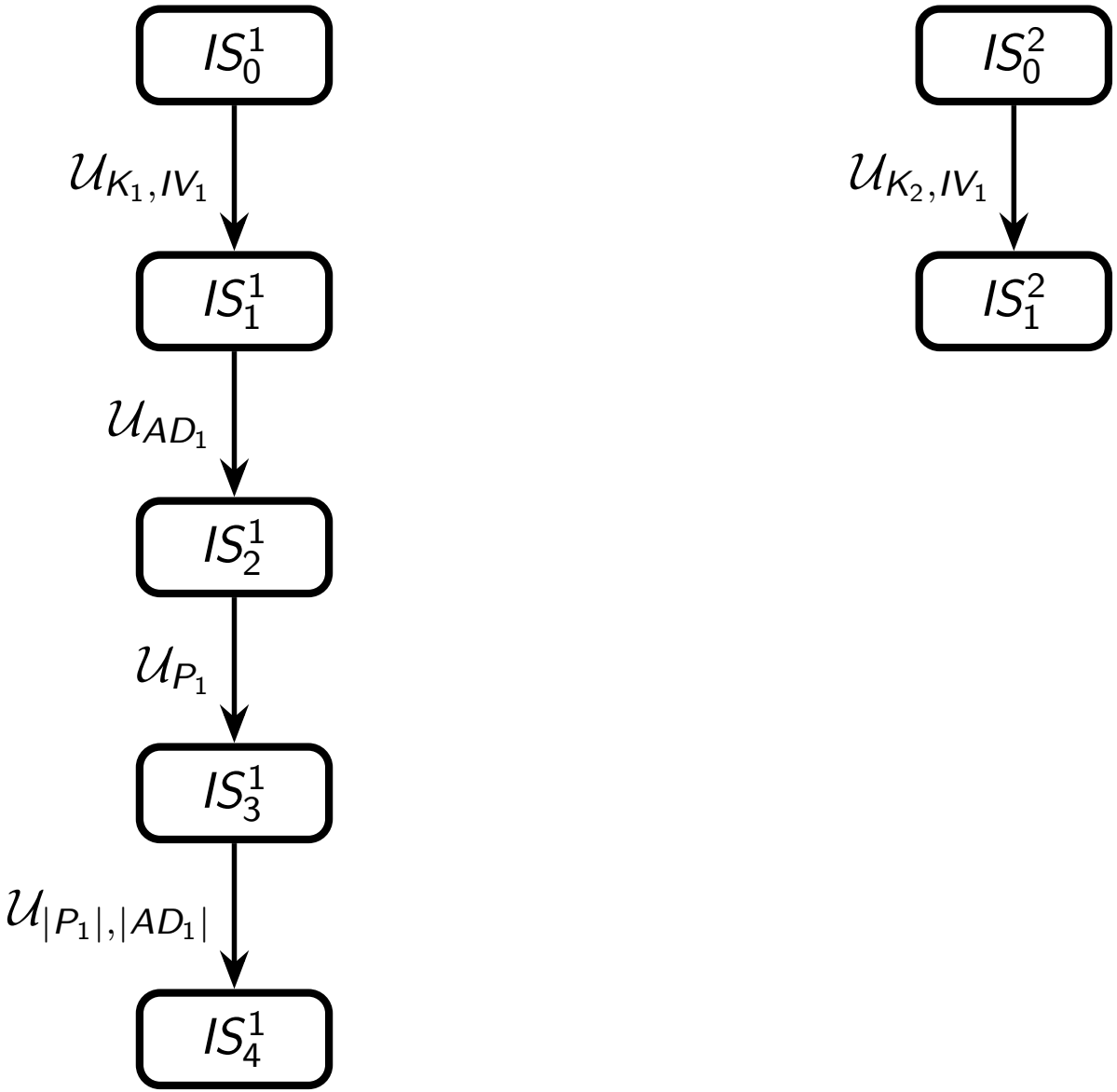
Attack Overview

Intermediate States using (K_1, IV_1, AD_1, P_1)

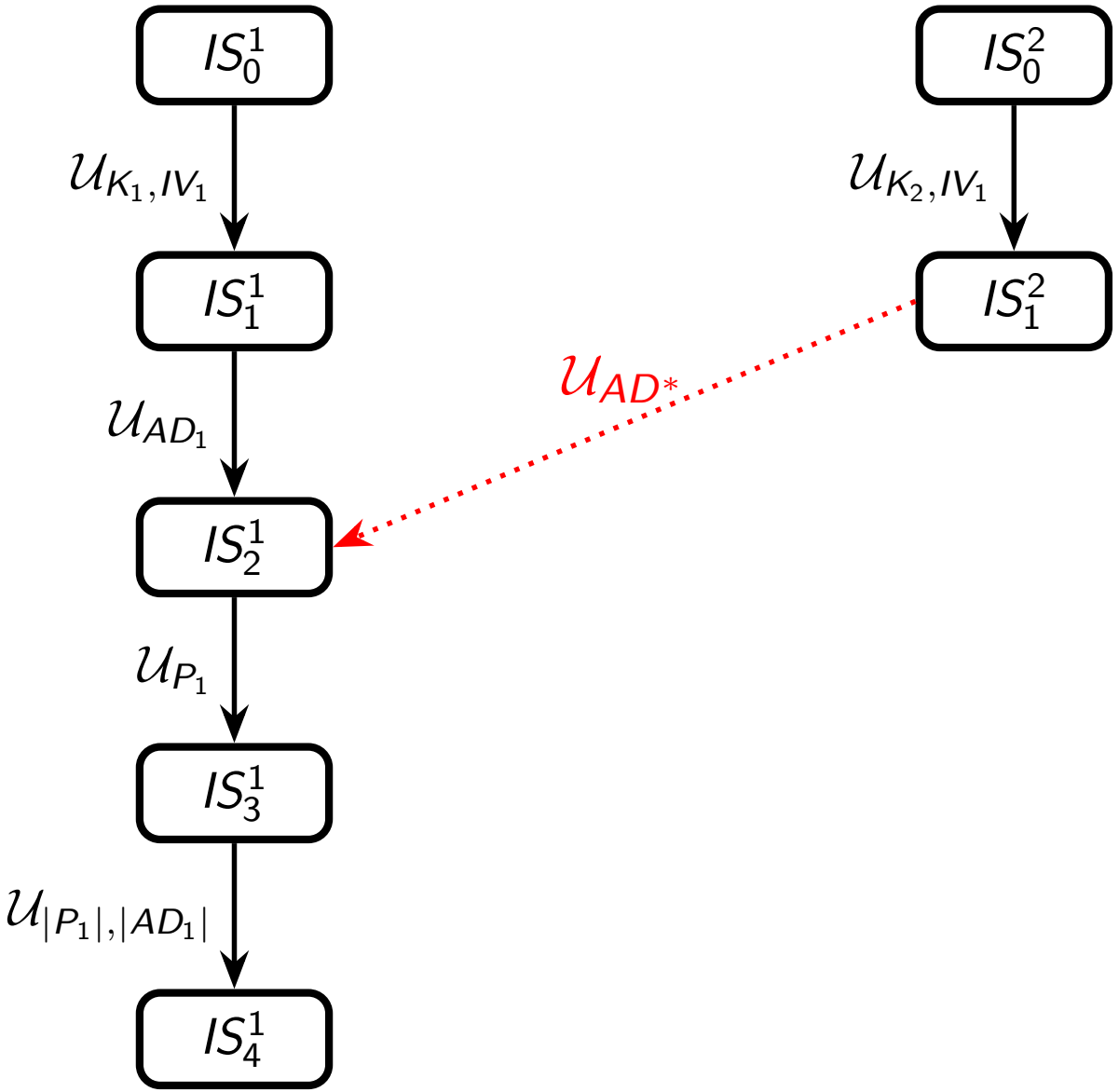


Attack Overview

Intermediate States using (K_2, IV_1)

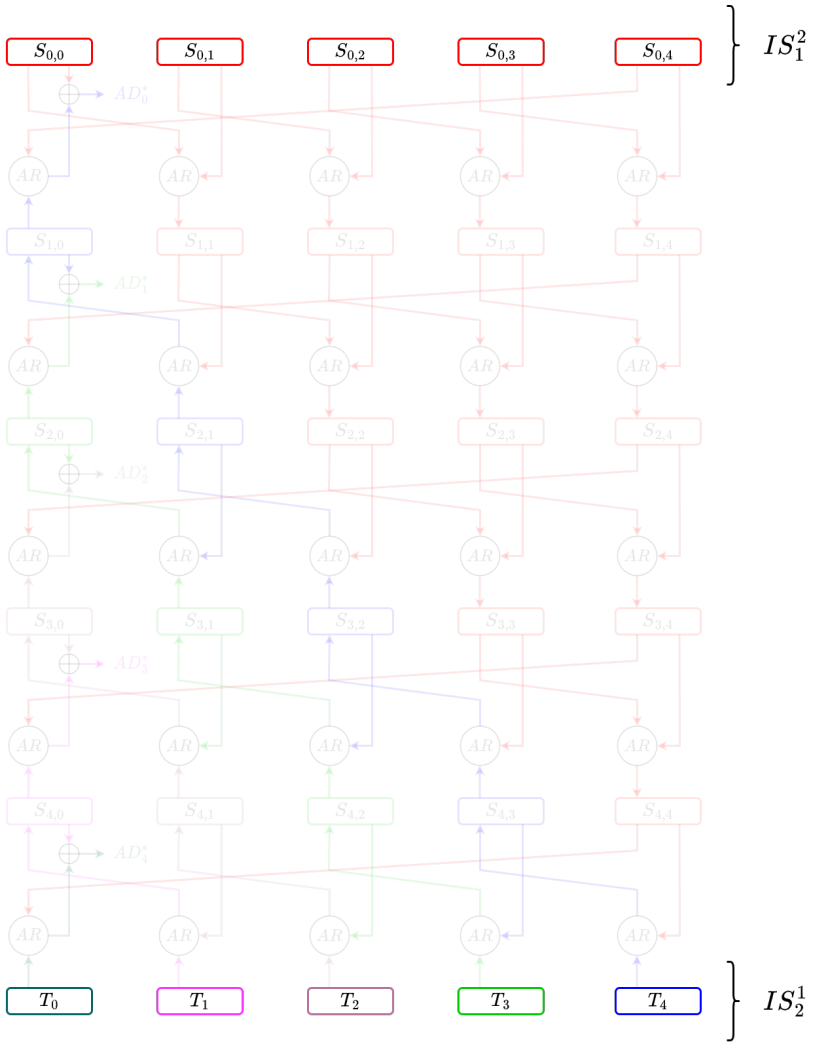


Attack Overview



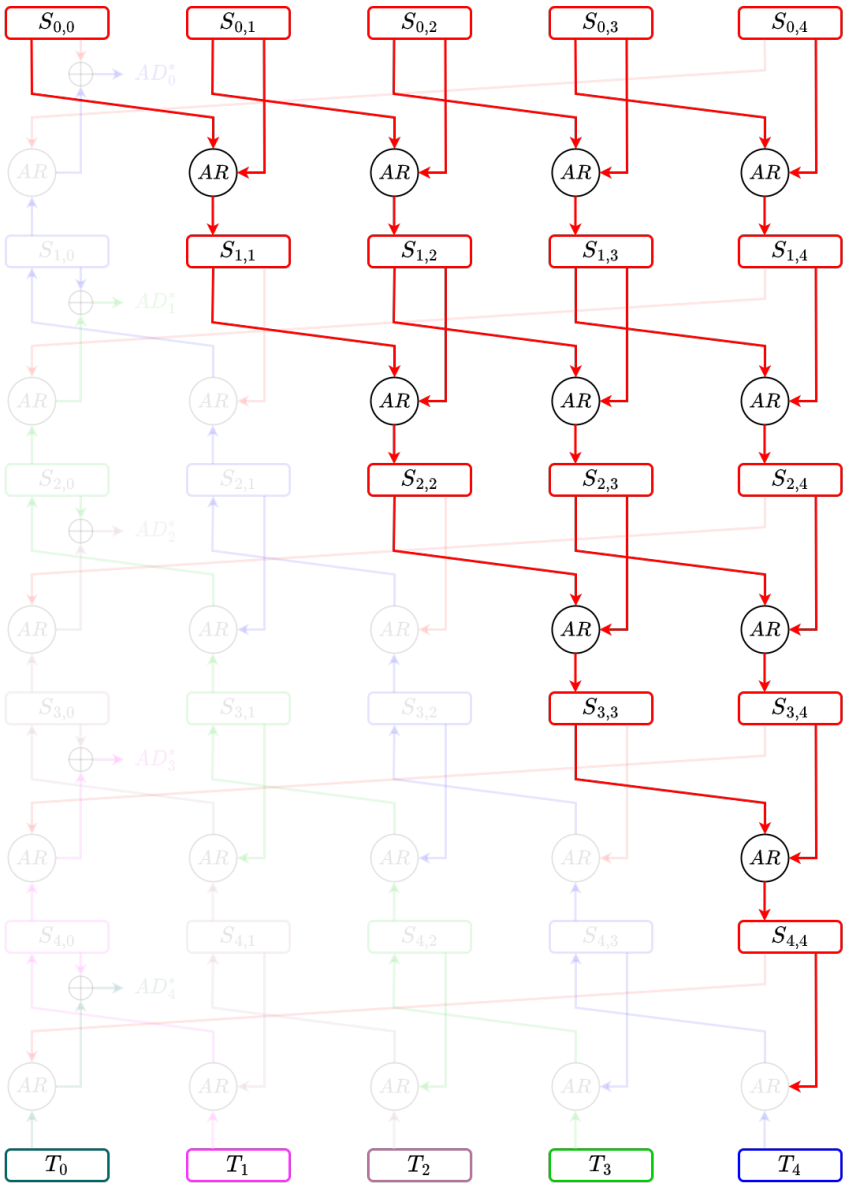
Attack on Aegis-128

Initial States IS_1^2 and IS_2^1



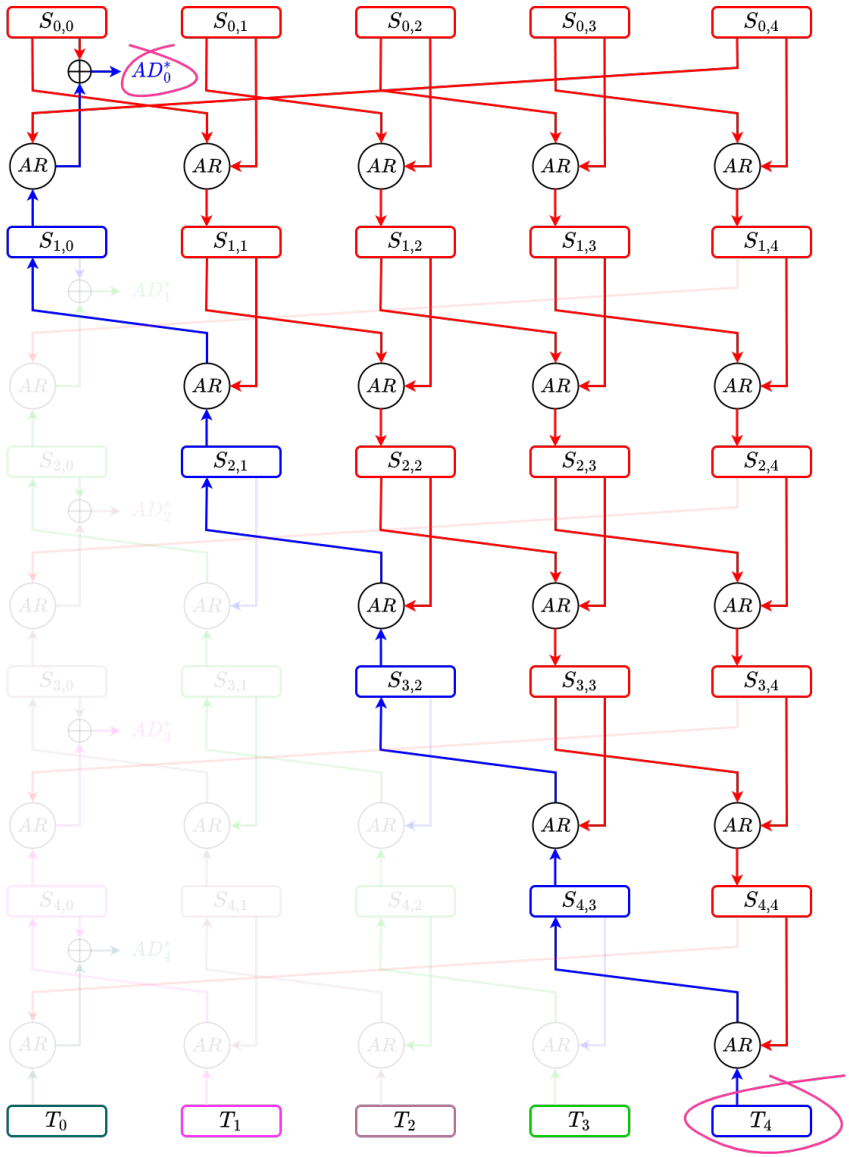
Attack on Aegis-128

Determining the Known Internal States



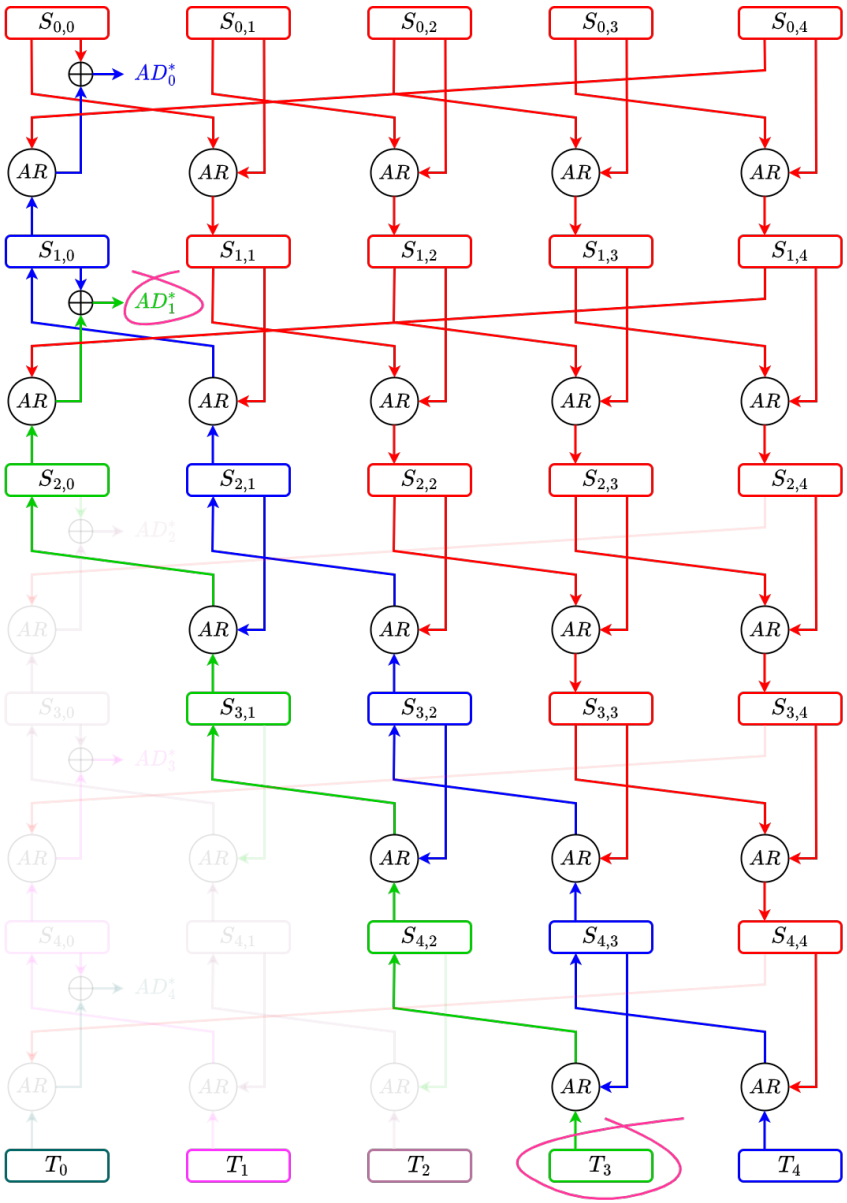
Attack on Aegis-128

Retrieving AD_0^*



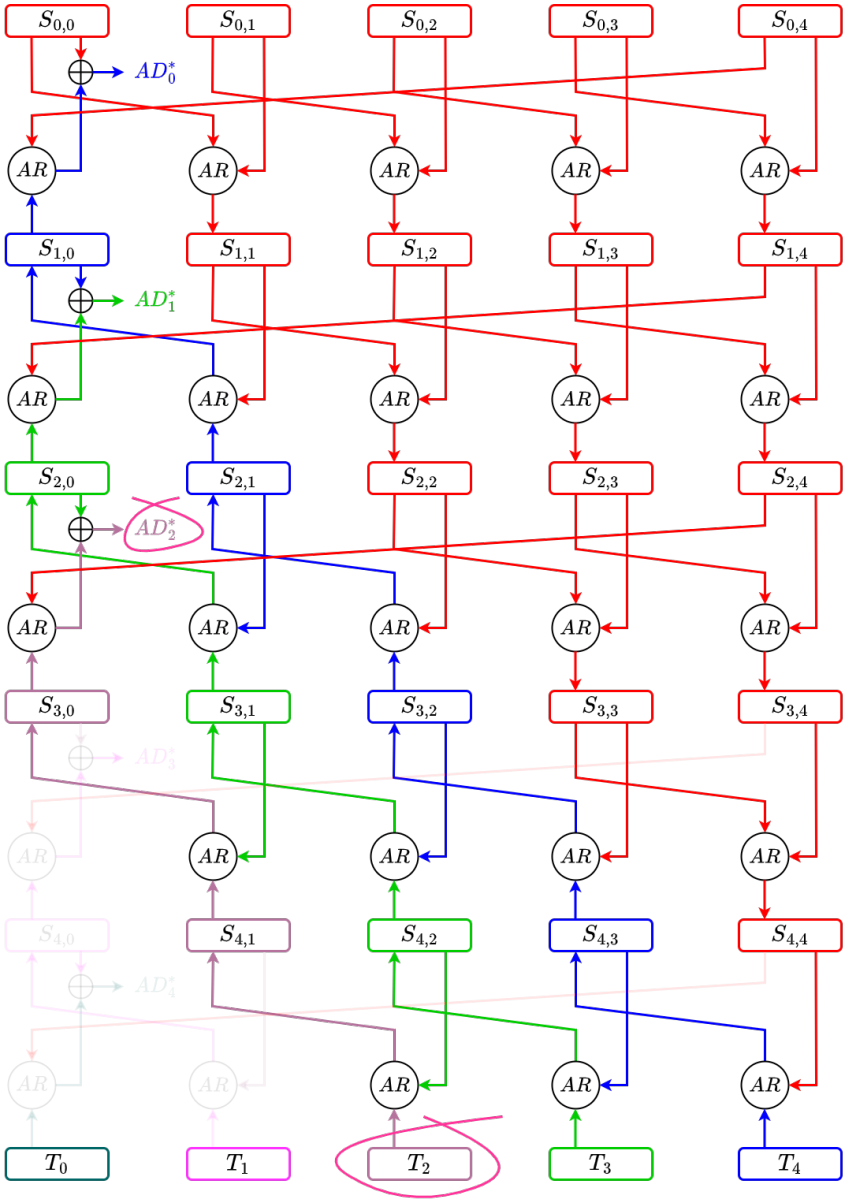
Attack on Aegis-128

Retrieving AD_1^*



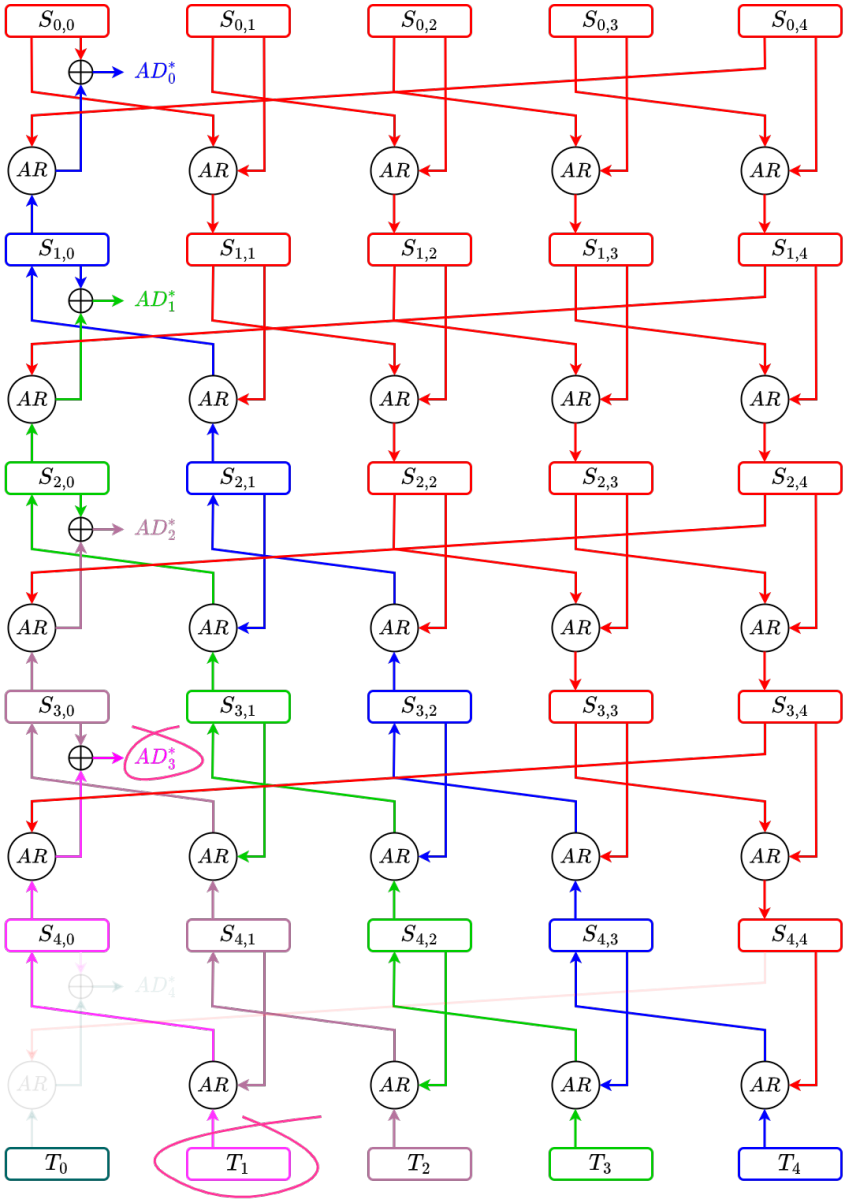
Attack on Aegis-128

Retrieving AD_2^*



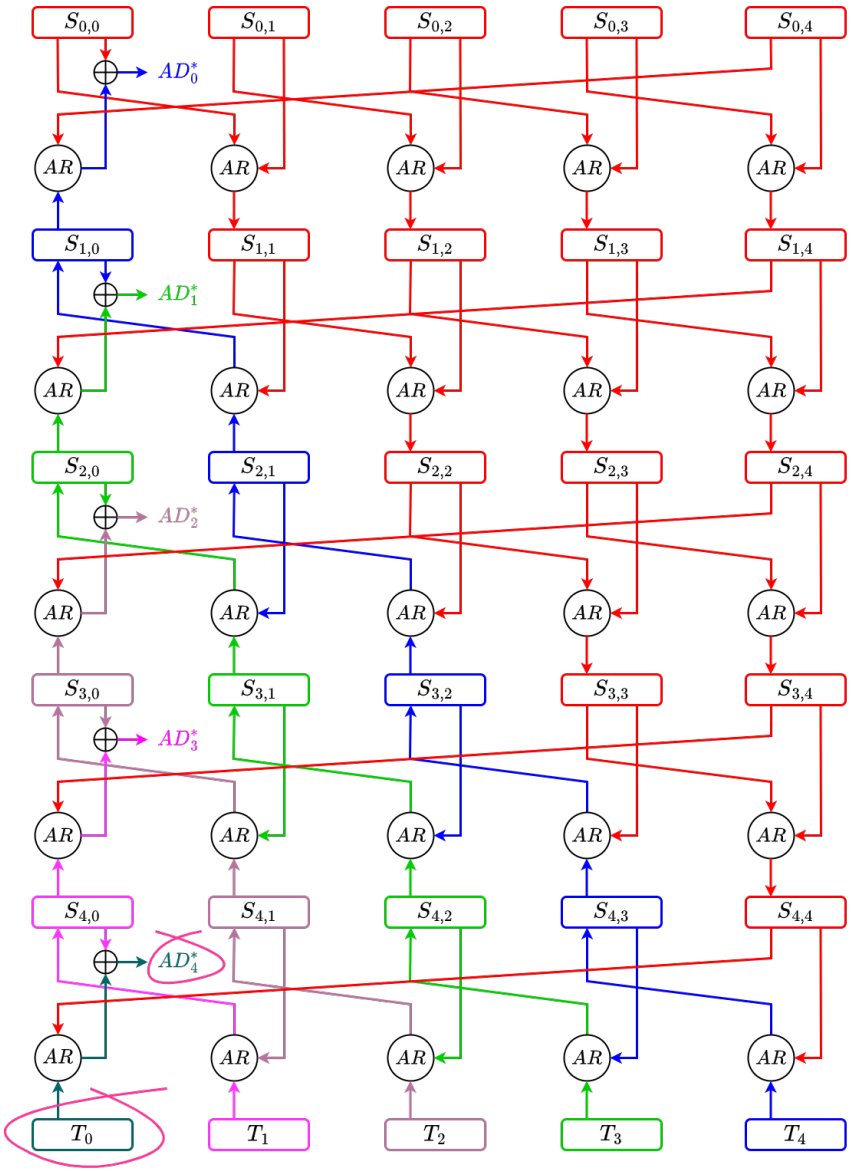
Attack on Aegis-128

Retrieving AD_3^*



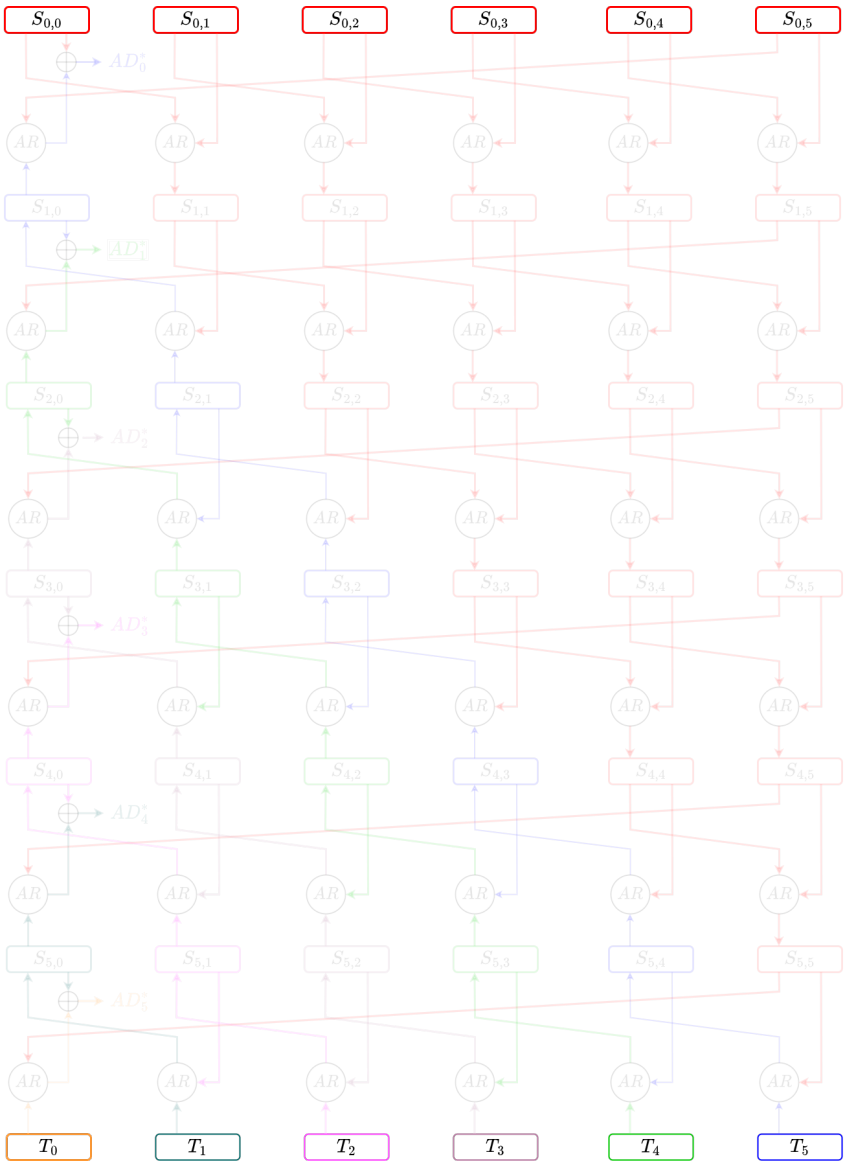
Attack on Aegis-128

Retrieving AD_4^*



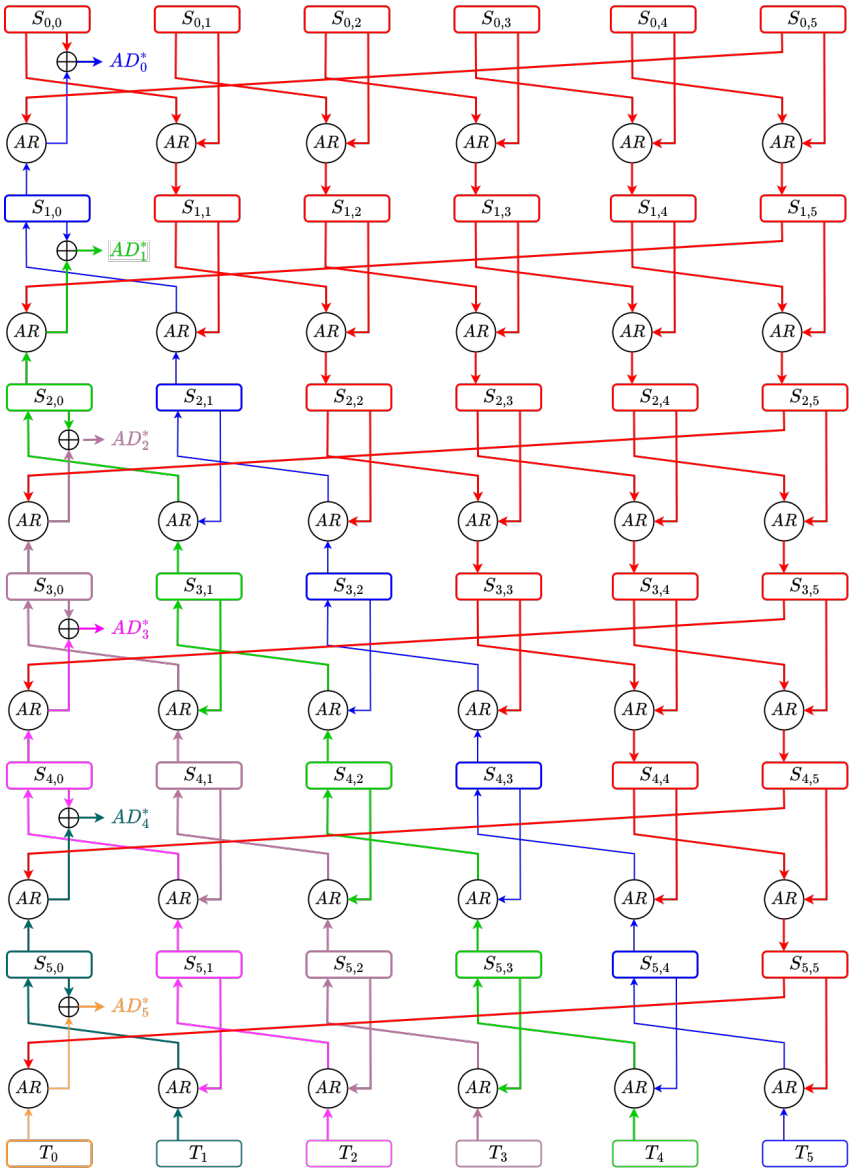
Attack on Aegis-256

Initial States IS_1^2 and IS_2^1



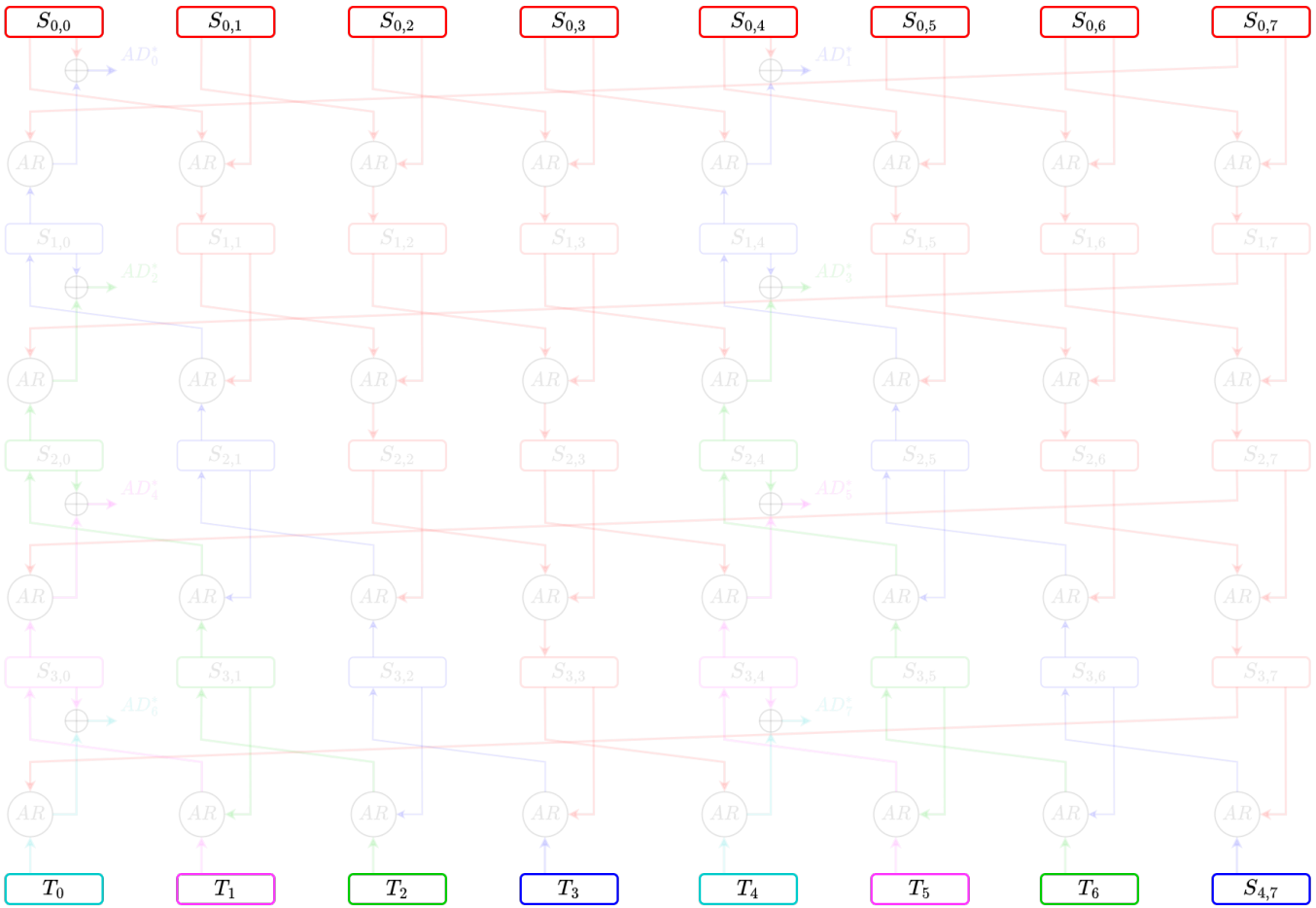
Attack on Aegis-256

Retrieving AD^*



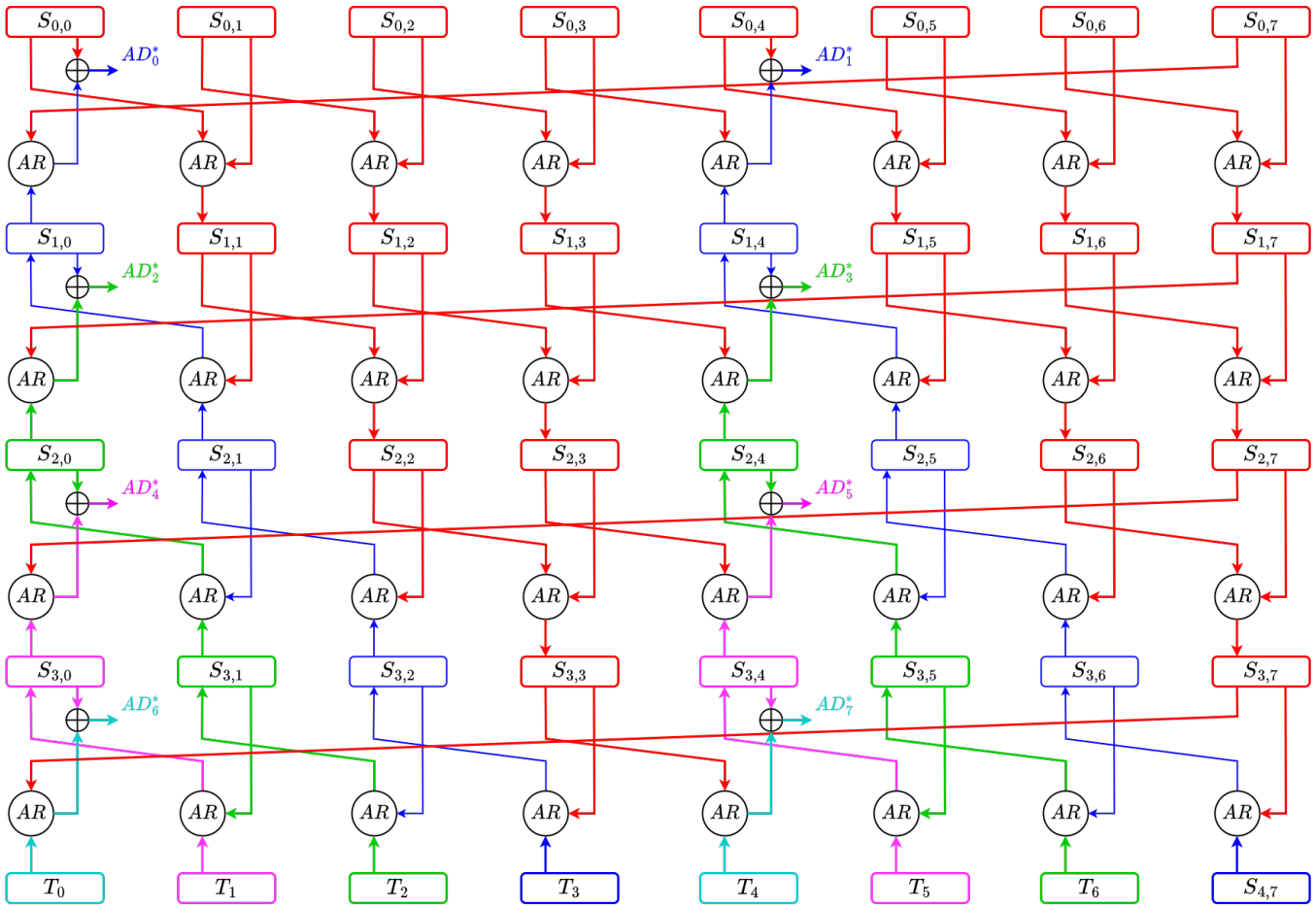
Attack on Aegis-128L

Initial States IS_1^2 and IS_2^1



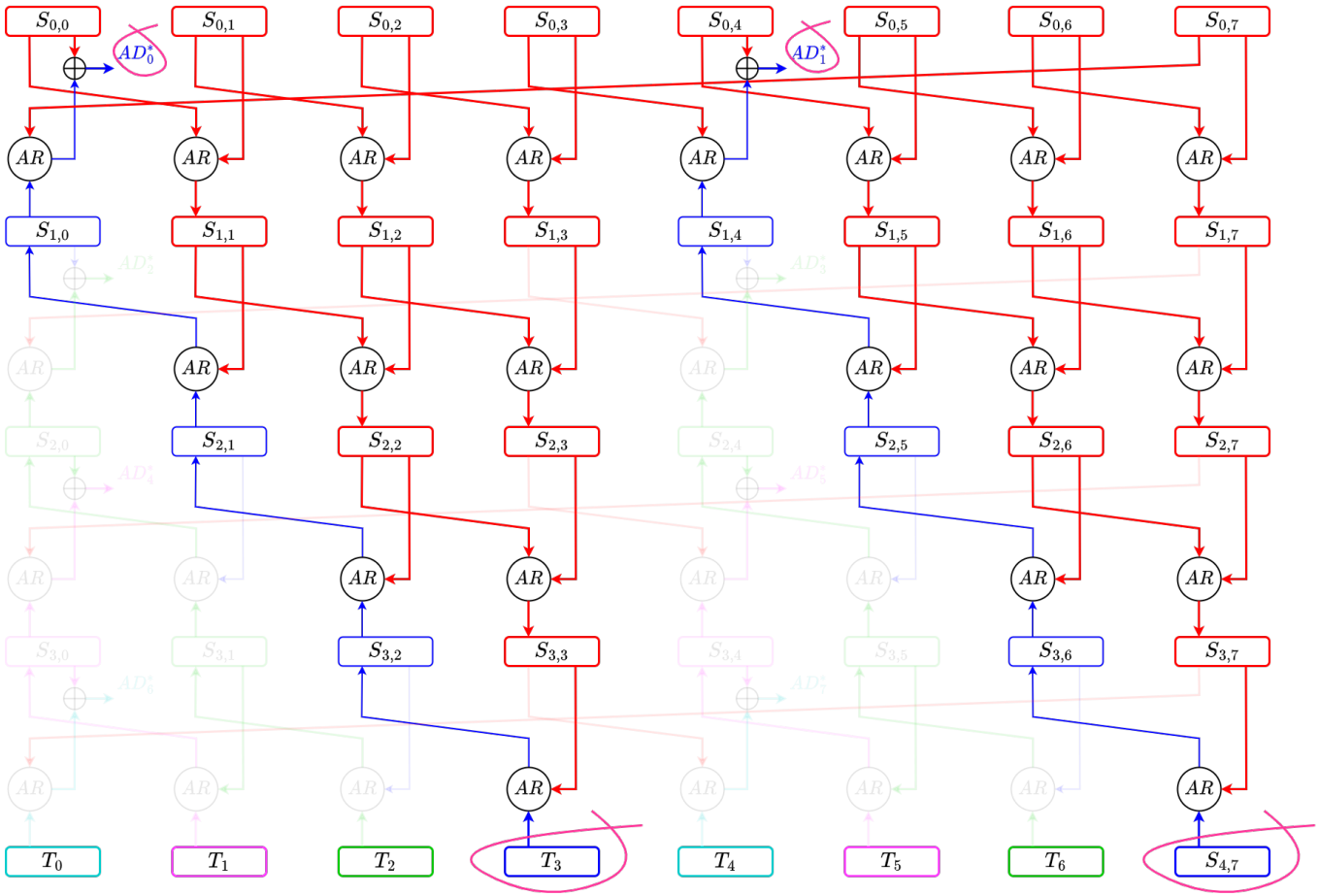
Attack on Aegis-128L

Retrieving AD^*



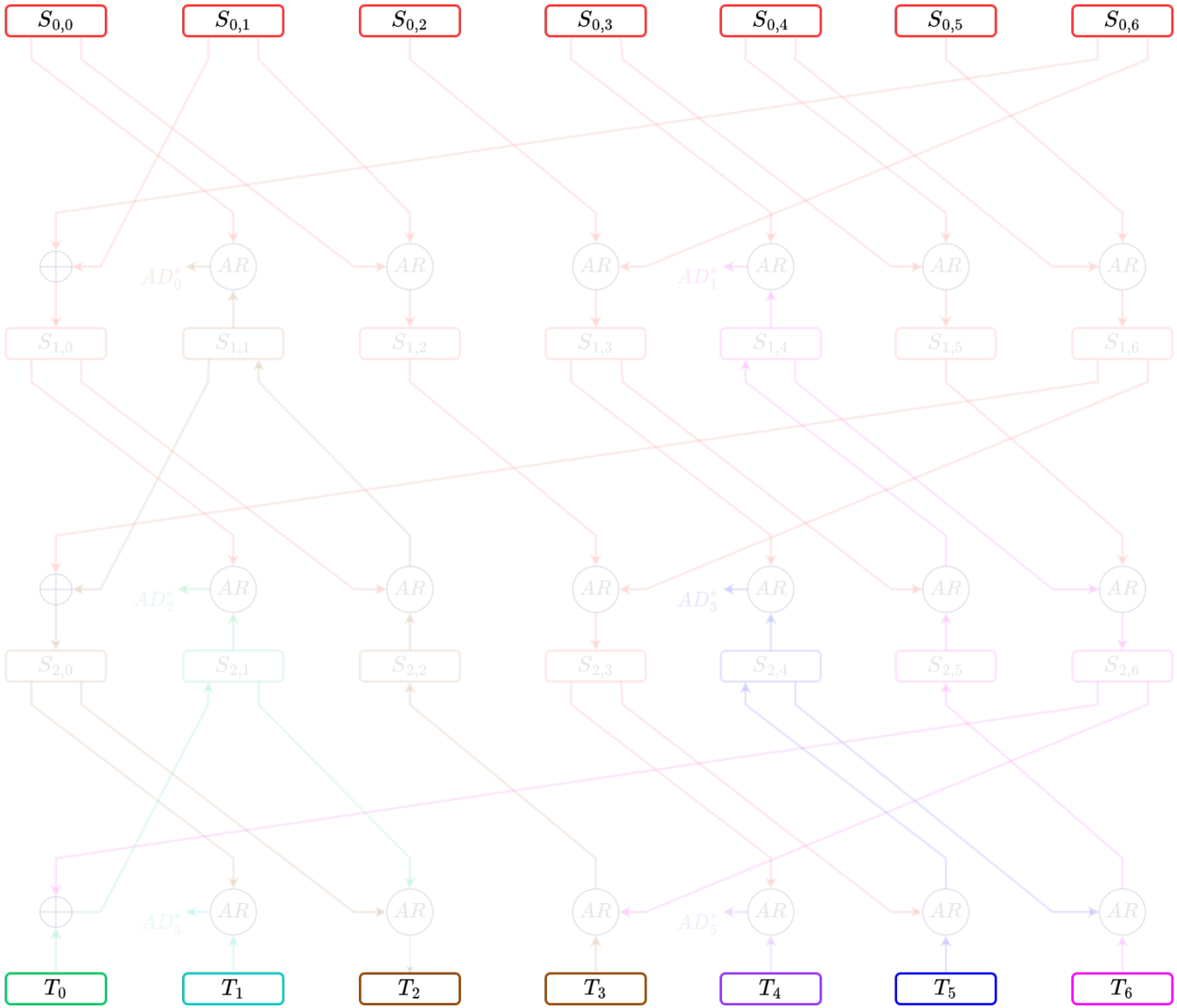
Attack on Aegis-128L

Retrieving two blocks



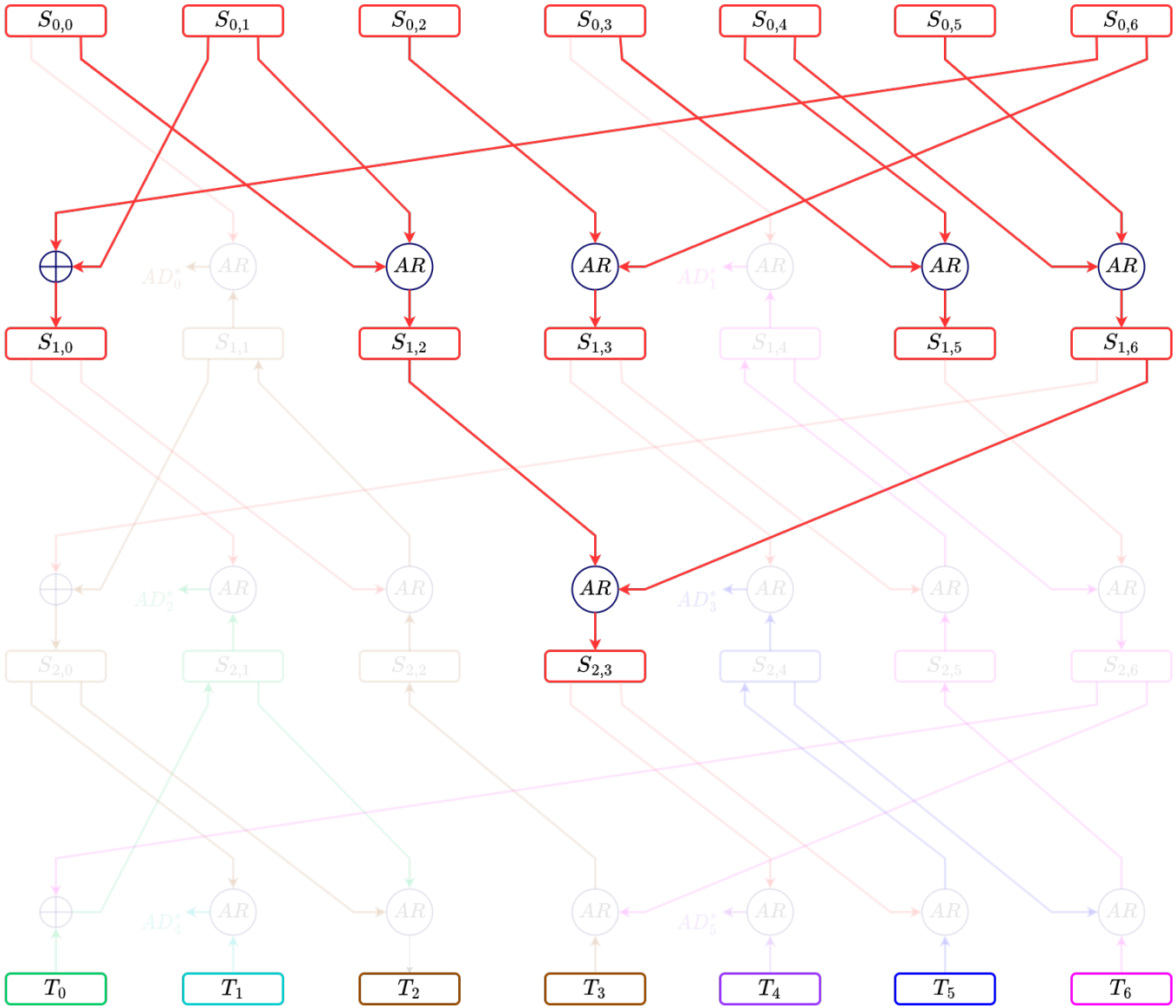
Attack on Rocca-S

Initial States IS_1^2 and IS_2^1



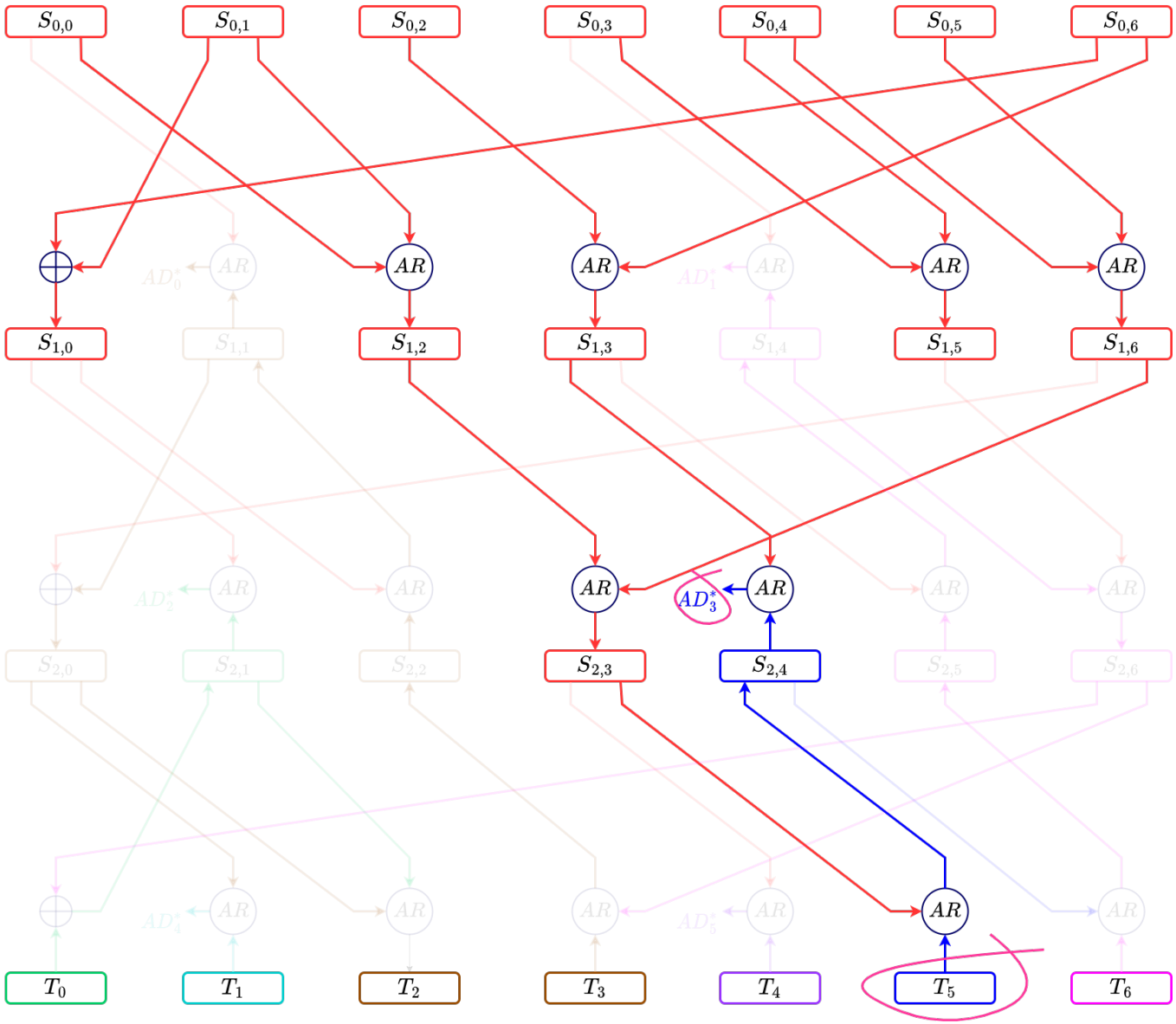
Attack on Rocca-S

Determining the known internal states



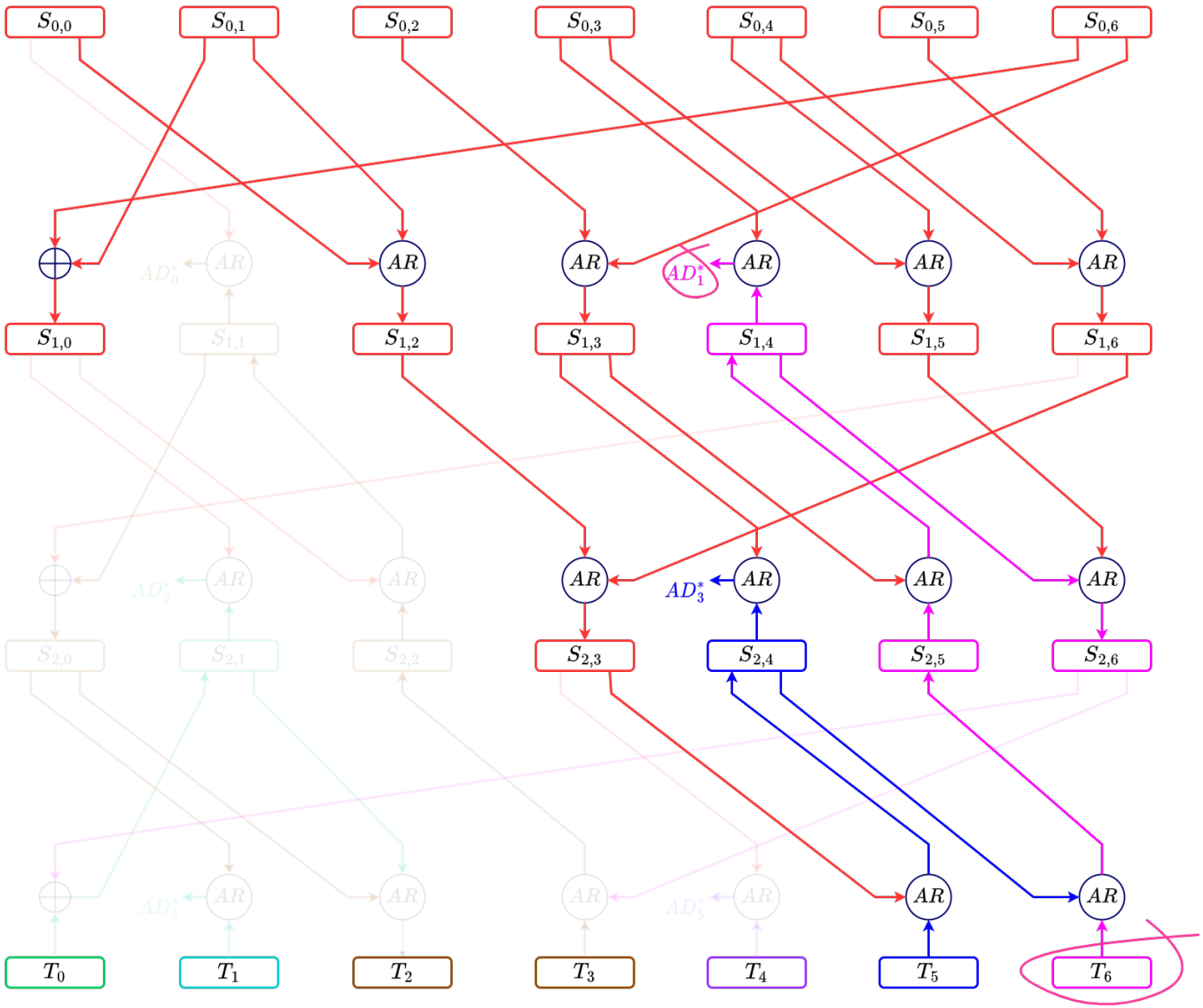
Attack on Rocca-S

Retrieving AD_3^*



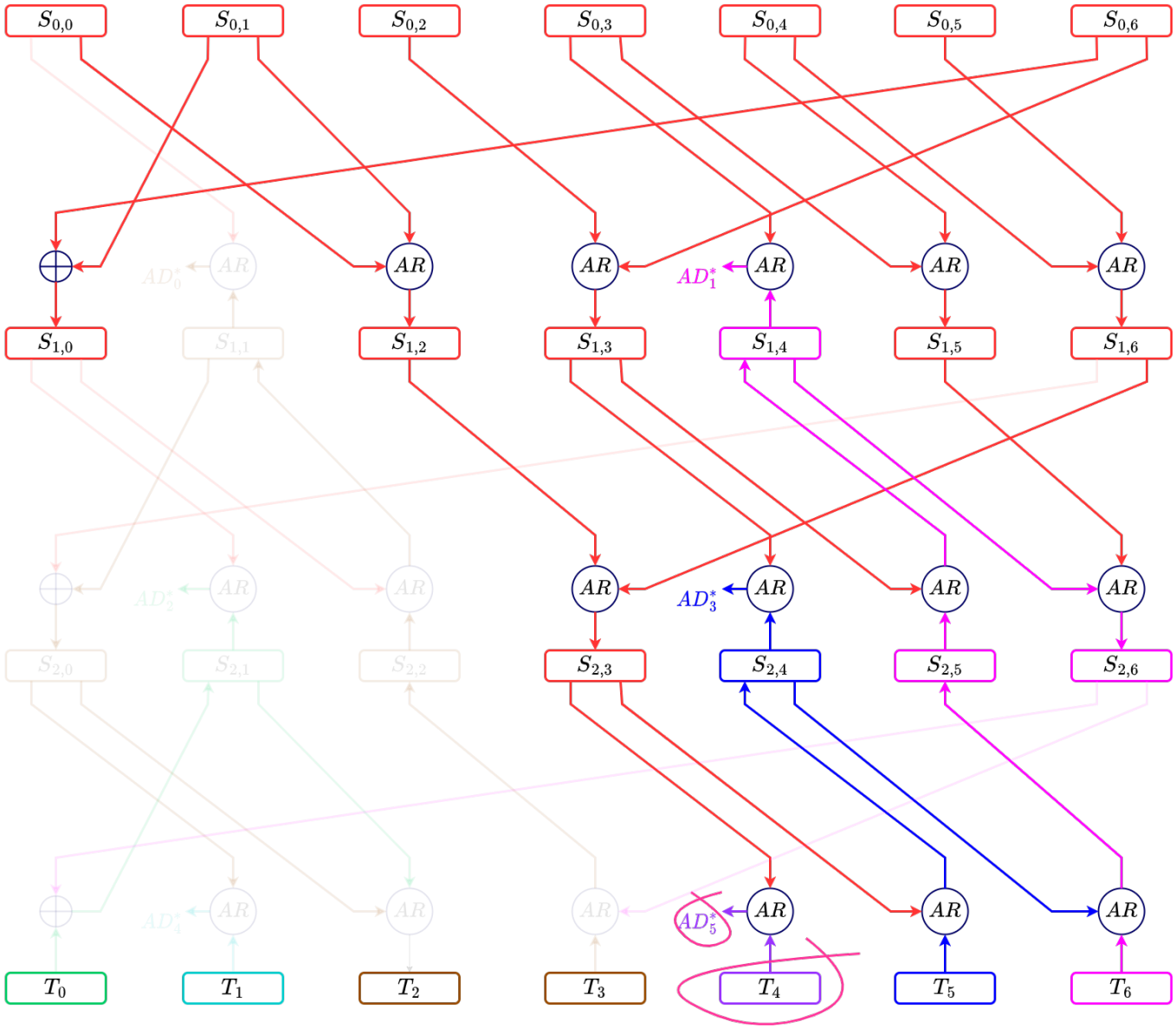
Attack on Rocca-S

Retrieving AD_1^*



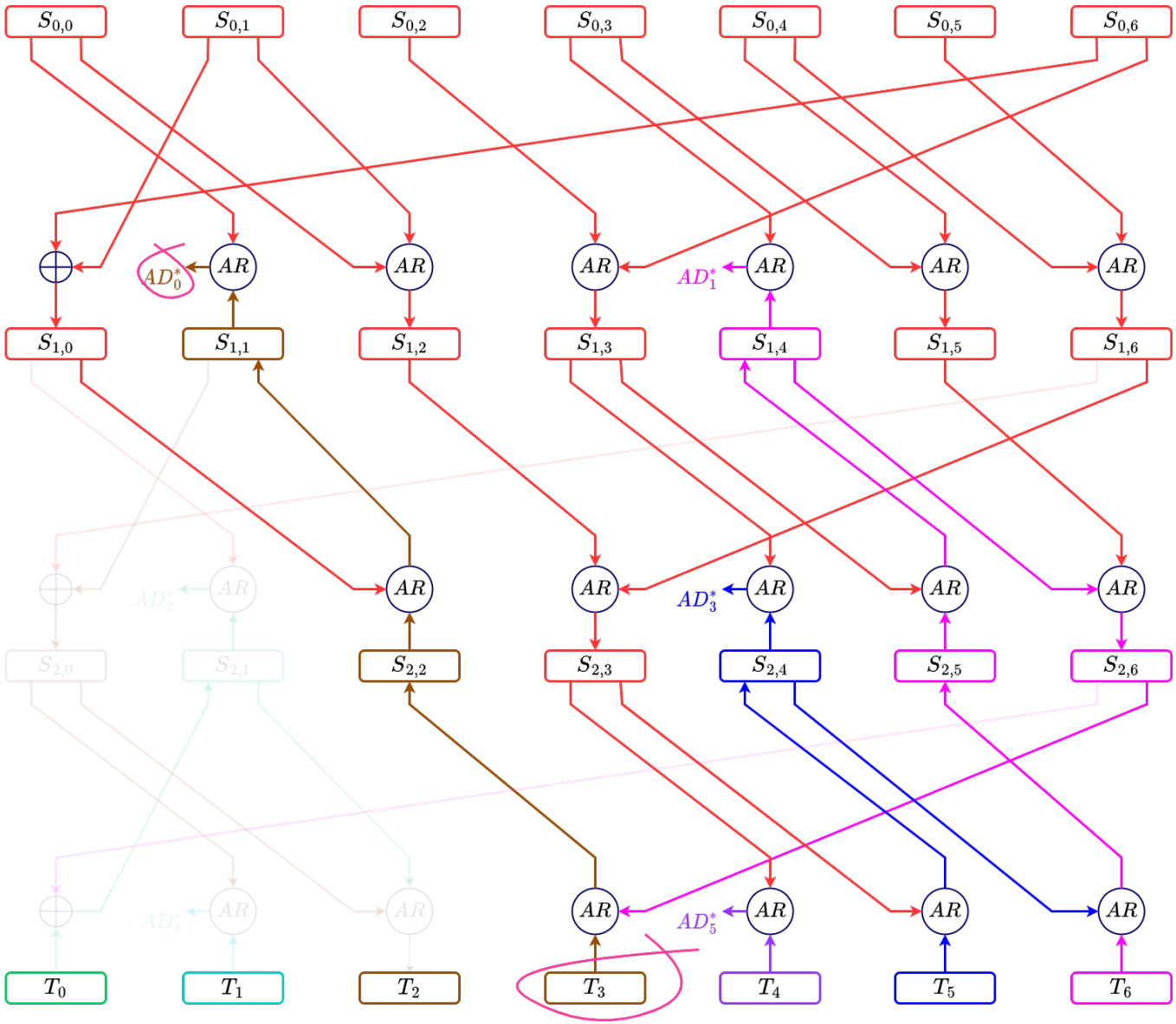
Attack on Rocca-S

Retrieving AD_5^*



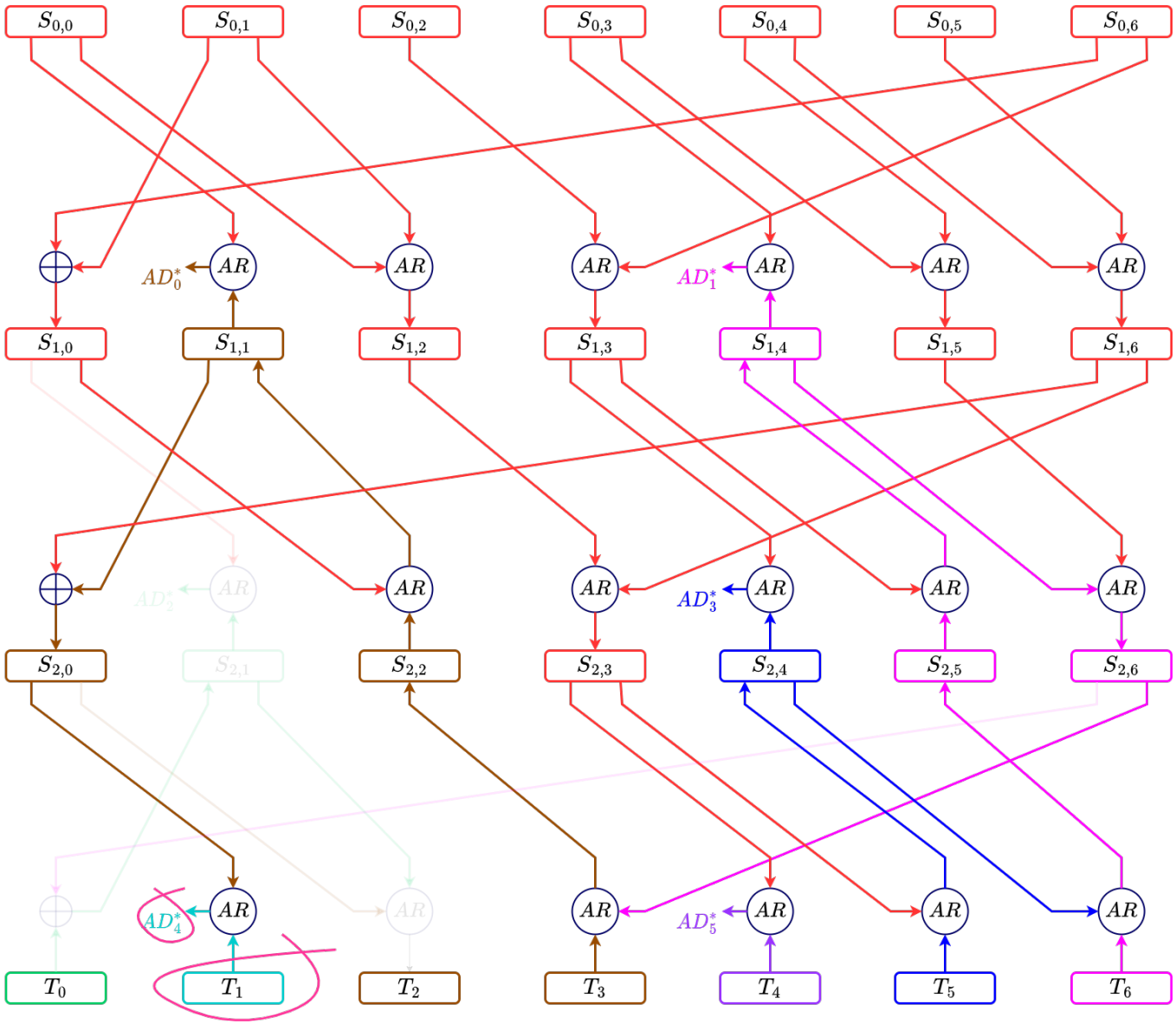
Attack on Rocca-S

Retrieving AD_0^*



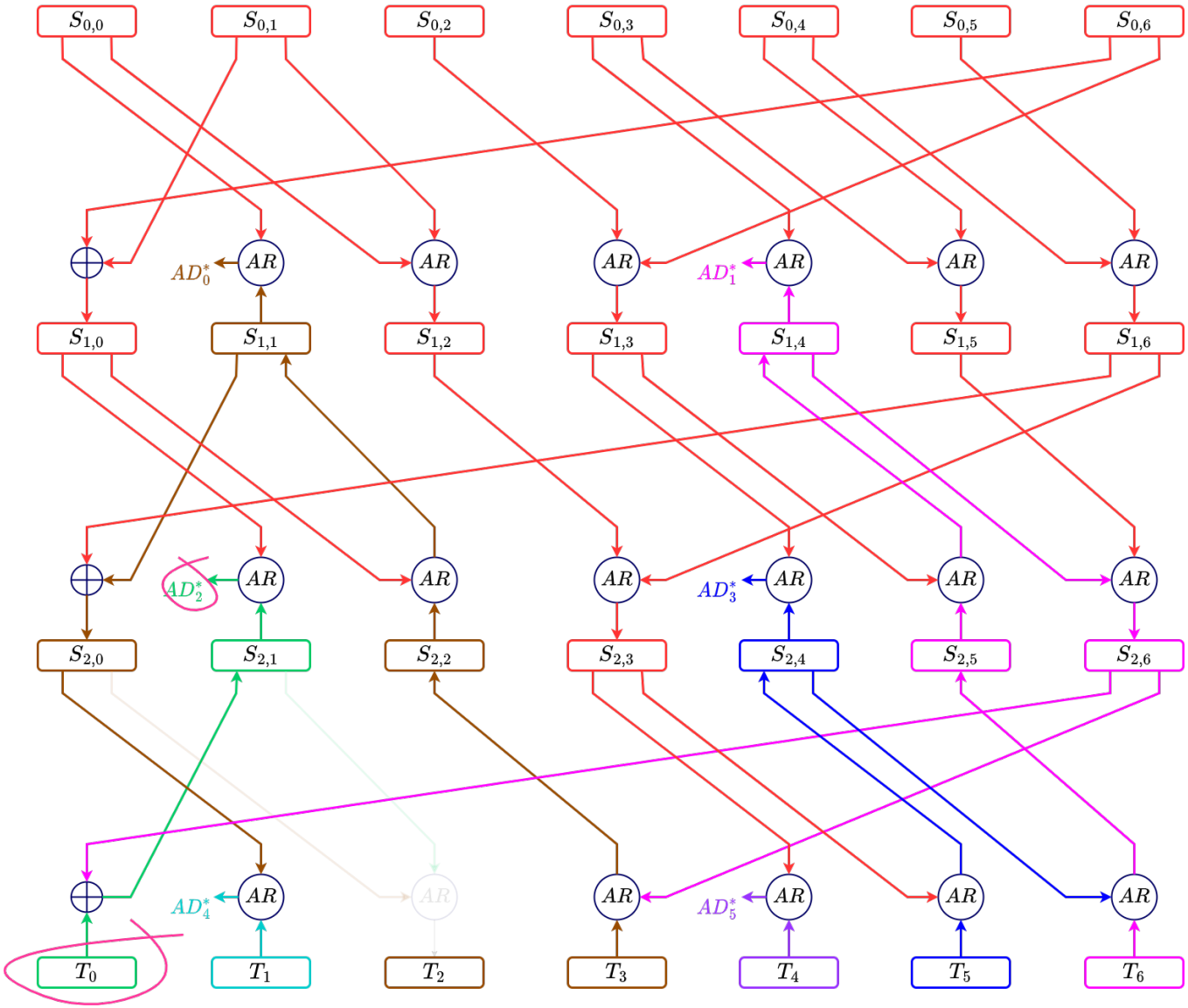
Attack on Rocca-S

Retrieving AD_4^*



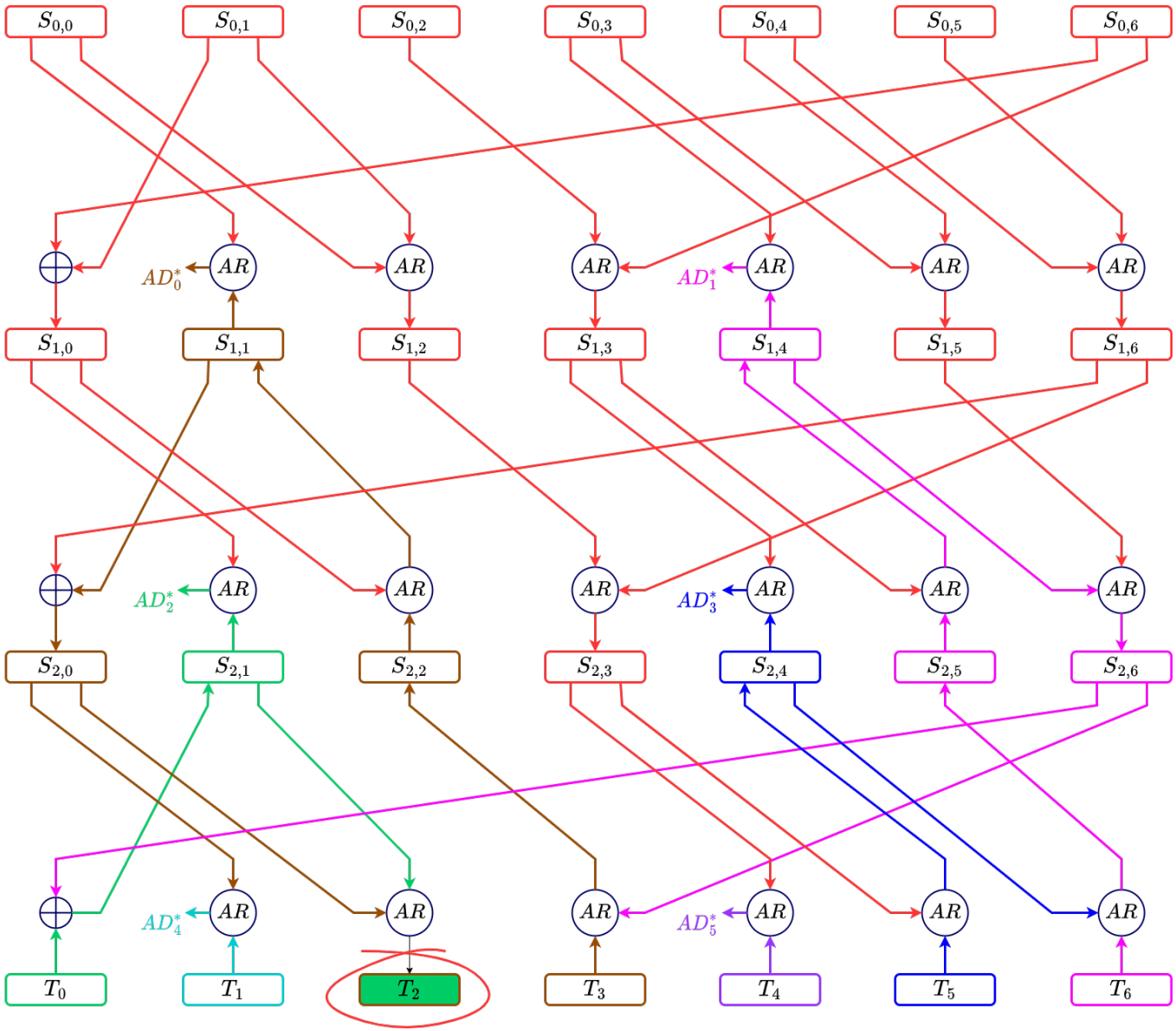
Attack on Rocca-S

Retrieving AD_2^*



Attack on Rocca-S

128-bit Collision



Proposed Attacks

AEAD Scheme	Tag Size (bits)	Attack Complexity
AEGIS-128	128	1
AEGIS-256		
AEGIS-128L	128/256	
Rocca-S	256	2^{64}

Conclusion

- Mounted key committing attacks on Aegis and Rocca-S
- The attacks on Aegis are experimentally verified
- The strategy does not work on Rocca and Tiaoxin-346

Thank You