Collisions for RC4-Hash

Sebastiaan Indesteege Bart Preneel
sebastiaan.indesteege@esat.kuleuven.be

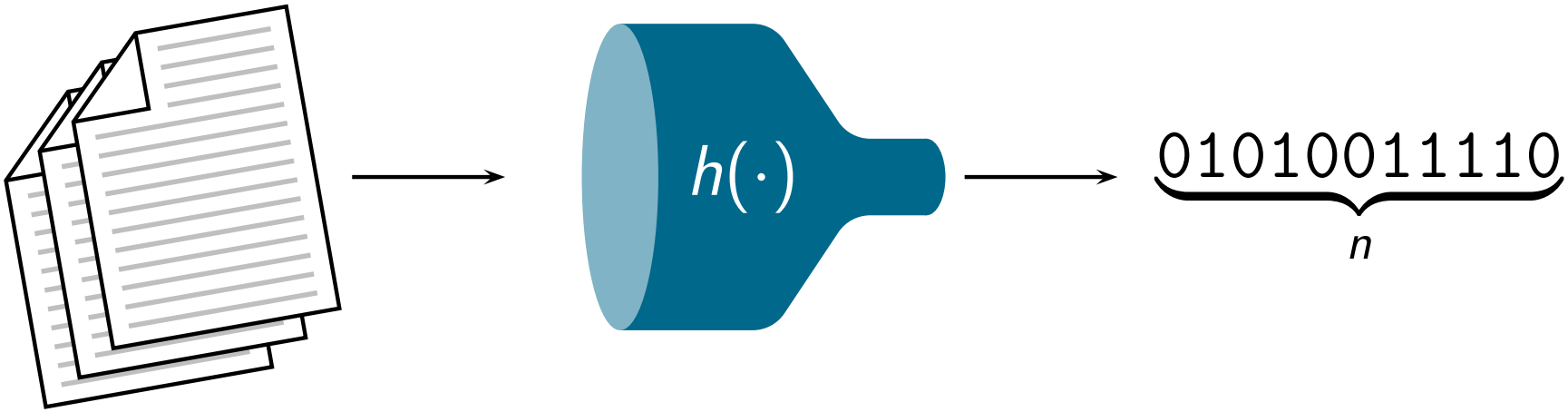
COSIC, ESAT/SCD, K.U. Leuven, Belgium

ISC 2008 — 18 September 2008

Outline

1. **Introduction**
   - Cryptographic Hash Functions
   - RC4-Hash
2. **Fixed Points for RC4-Hash**
   - Type I
   - Type II
3. **Collisions for RC4-Hash**
4. **Conclusion**

Cryptographic Hash Functions



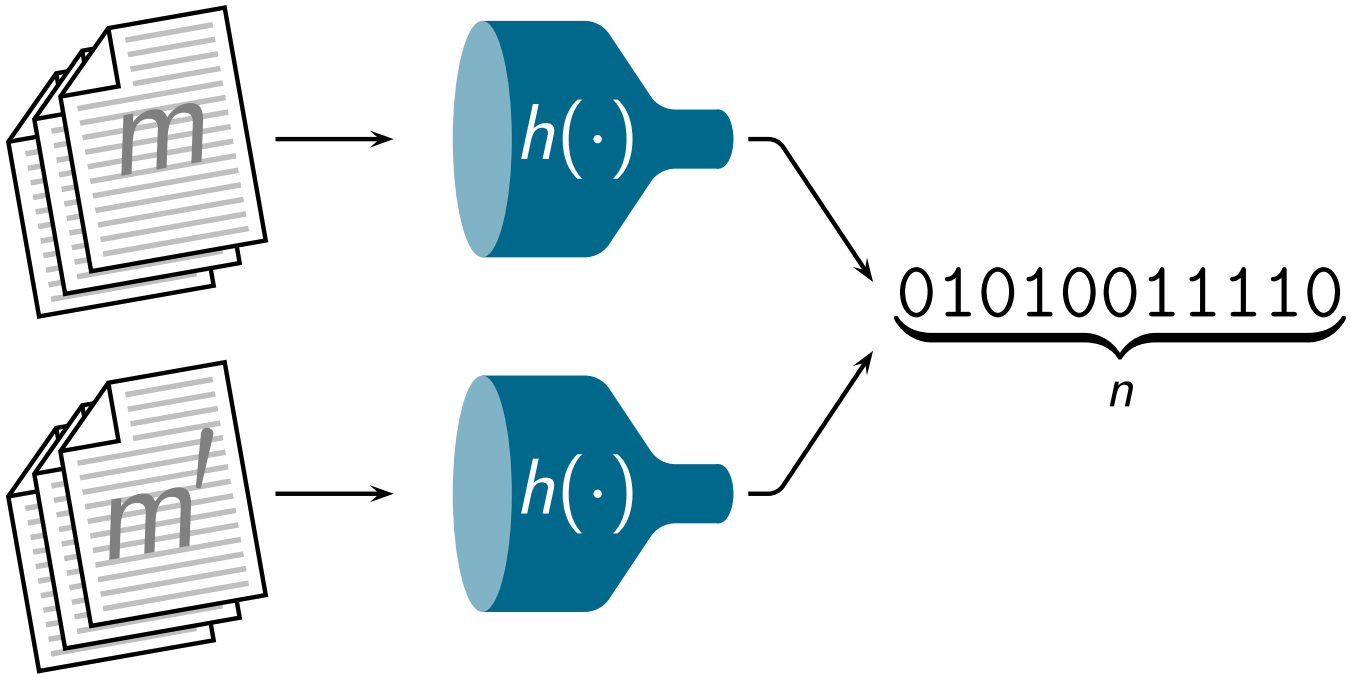
$$h : \{0,1\}^* \mapsto \{0,1\}^n$$

Desired properties

- **Collision resistance**
- (Second) preimage resistance
- ...

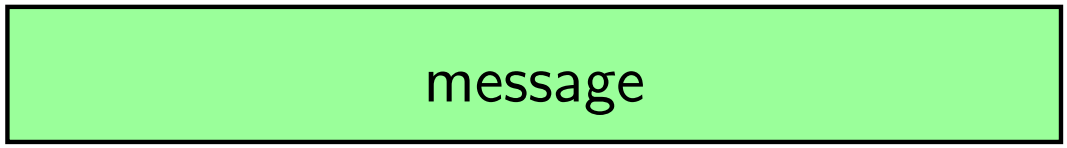
Cryptographic Hash Functions

Collision Resistance



- "Hard" to find $m \neq m'$ s.t. $h(m) = h(m')$.
- Birthday paradox $\mathcal{O}(2^{n/2})$
- This talk: **RC4-Hash is not collision resistant**

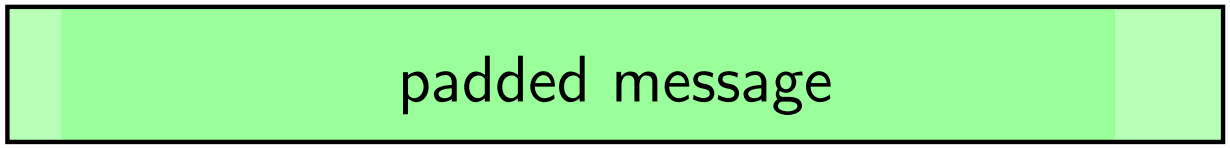
RC4-Hash



RC4-Hash

- D. Chang, K. C. Gupta, and M. Nandi (2006)

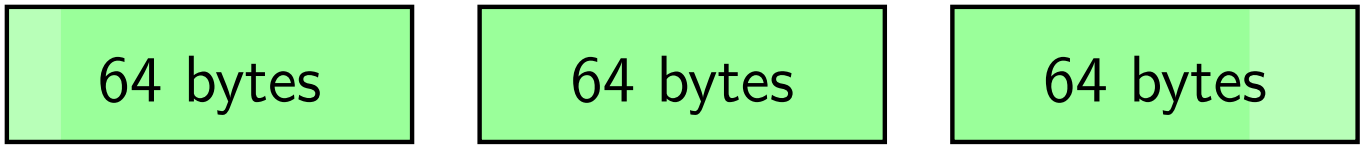
RC4-Hash



RC4-Hash

- D. Chang, K. C. Gupta, and M. Nandi (2006)

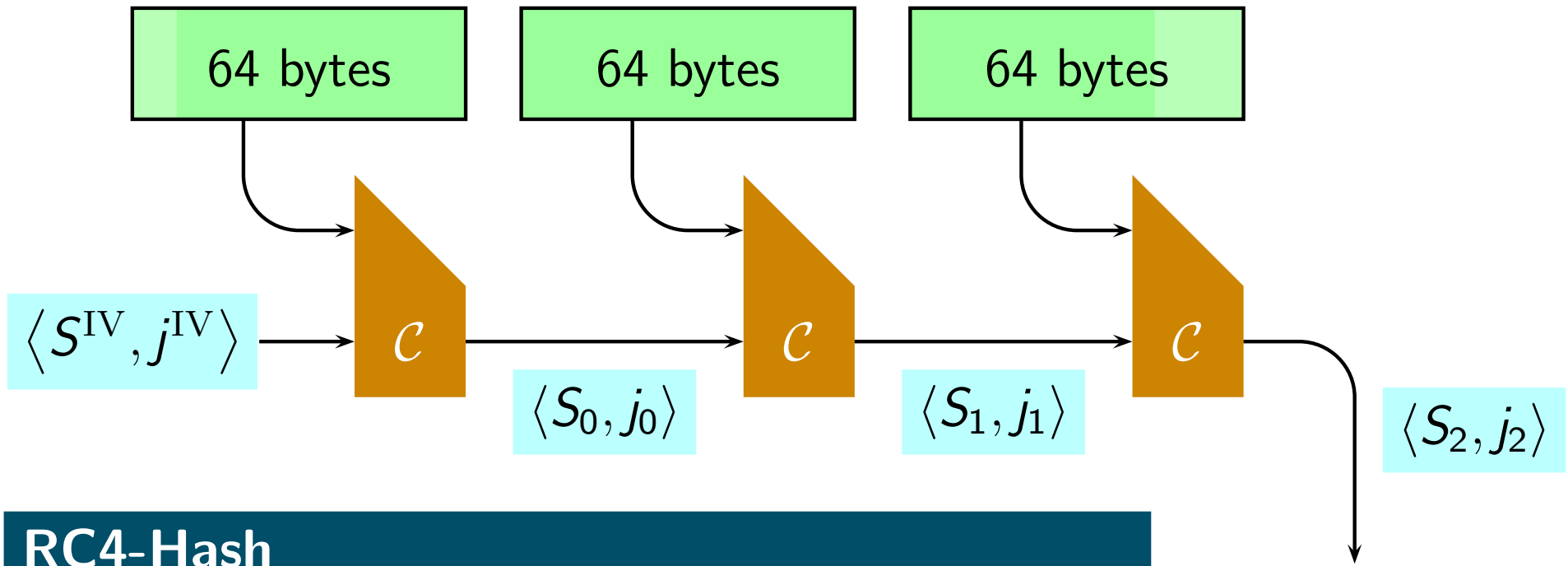
RC4-Hash



RC4-Hash

- D. Chang, K. C. Gupta, and M. Nandi (2006)
- Iterated hash function

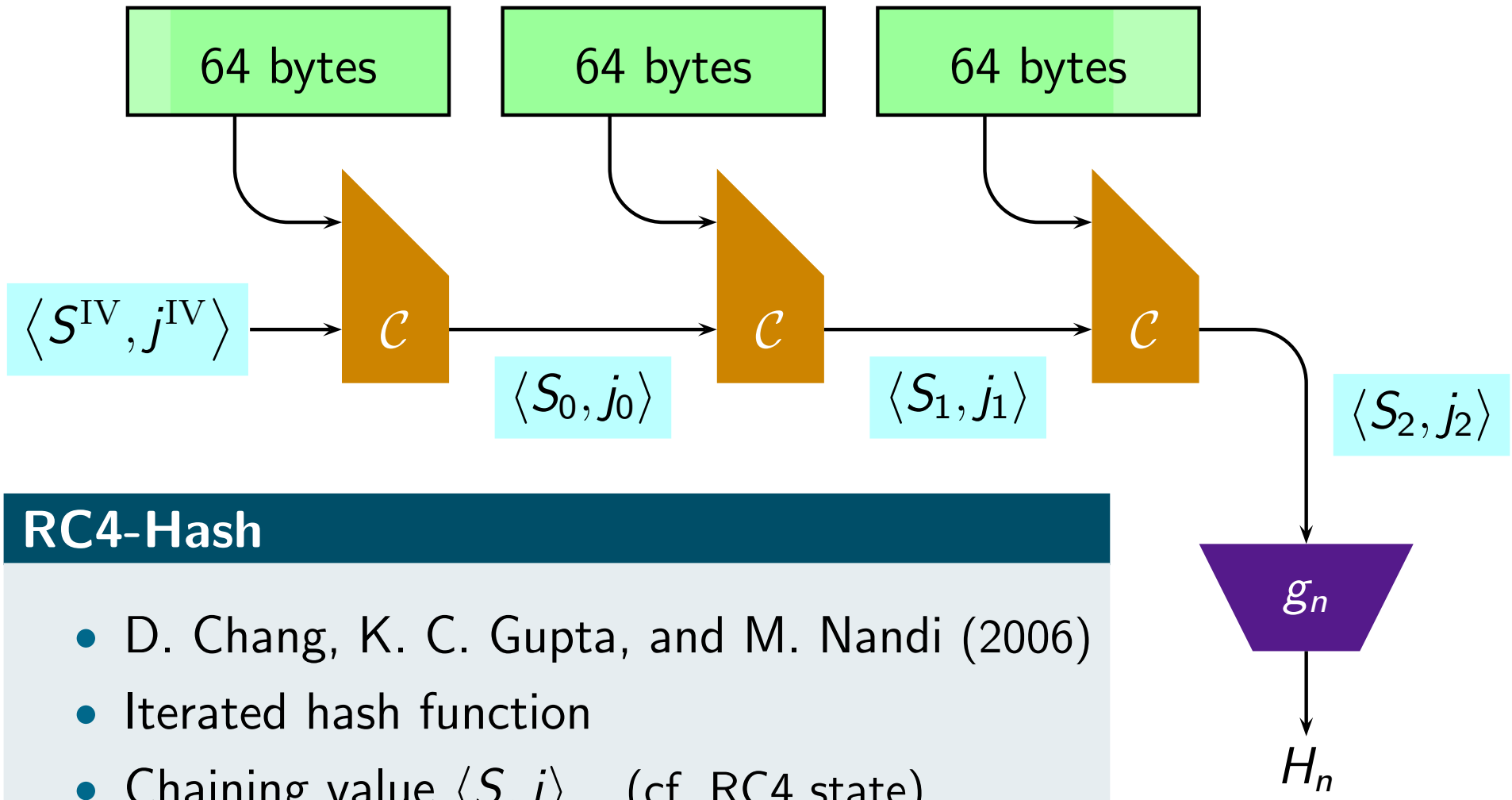
RC4-Hash



RC4-Hash

- D. Chang, K. C. Gupta, and M. Nandi (2006)
- Iterated hash function
- Chaining value $\langle S, j \rangle$ (cf. RC4 state)

RC4-Hash

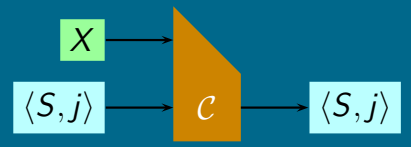


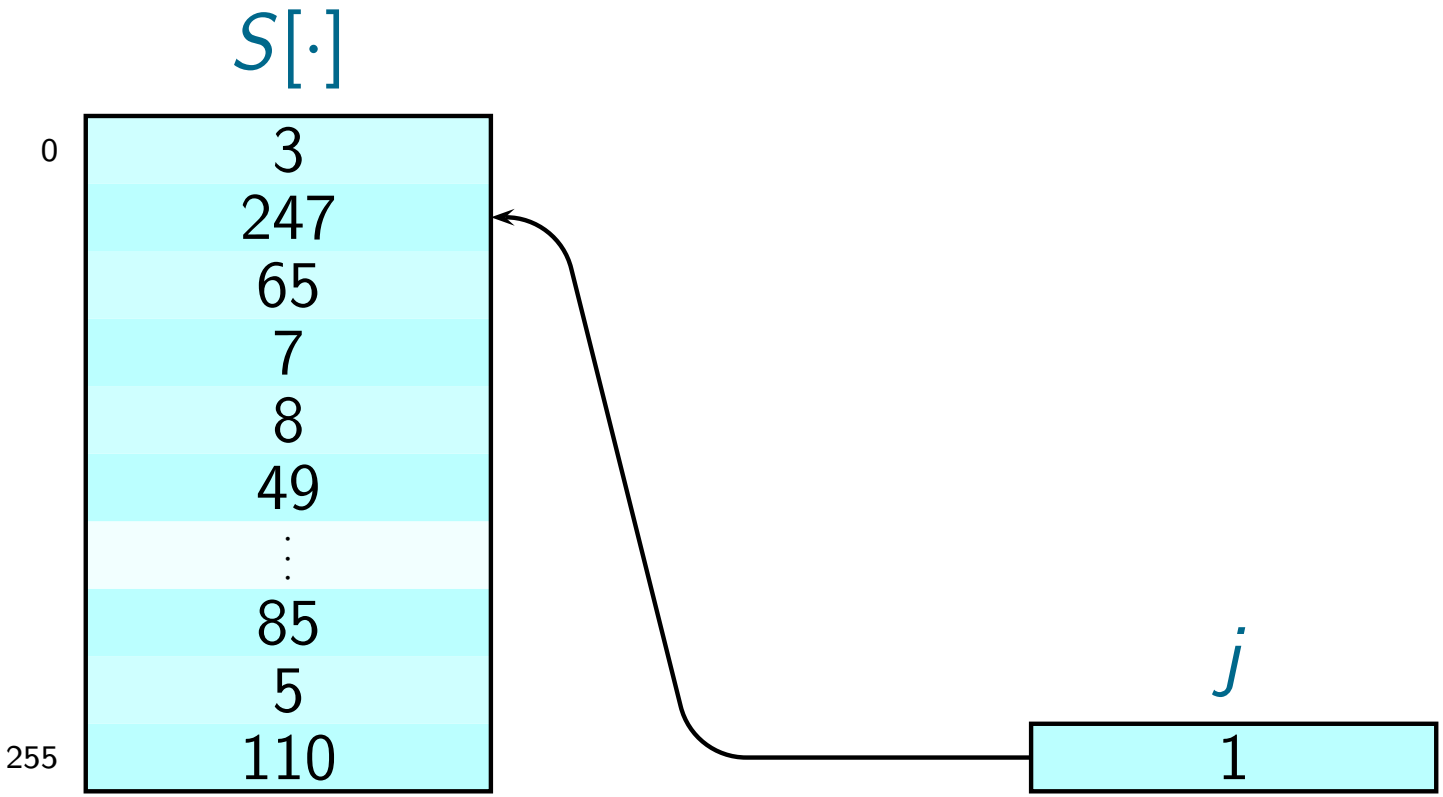
RC4-Hash

- D. Chang, K. C. Gupta, and M. Nandi (2006)
- Iterated hash function
- Chaining value $\langle S, j \rangle$ (cf. RC4 state)
- Variable output length $n = 128 - 512$bits

RC4-Hash

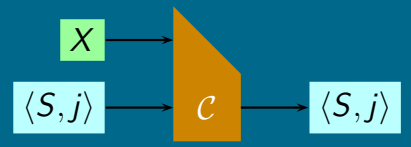
Compression Function

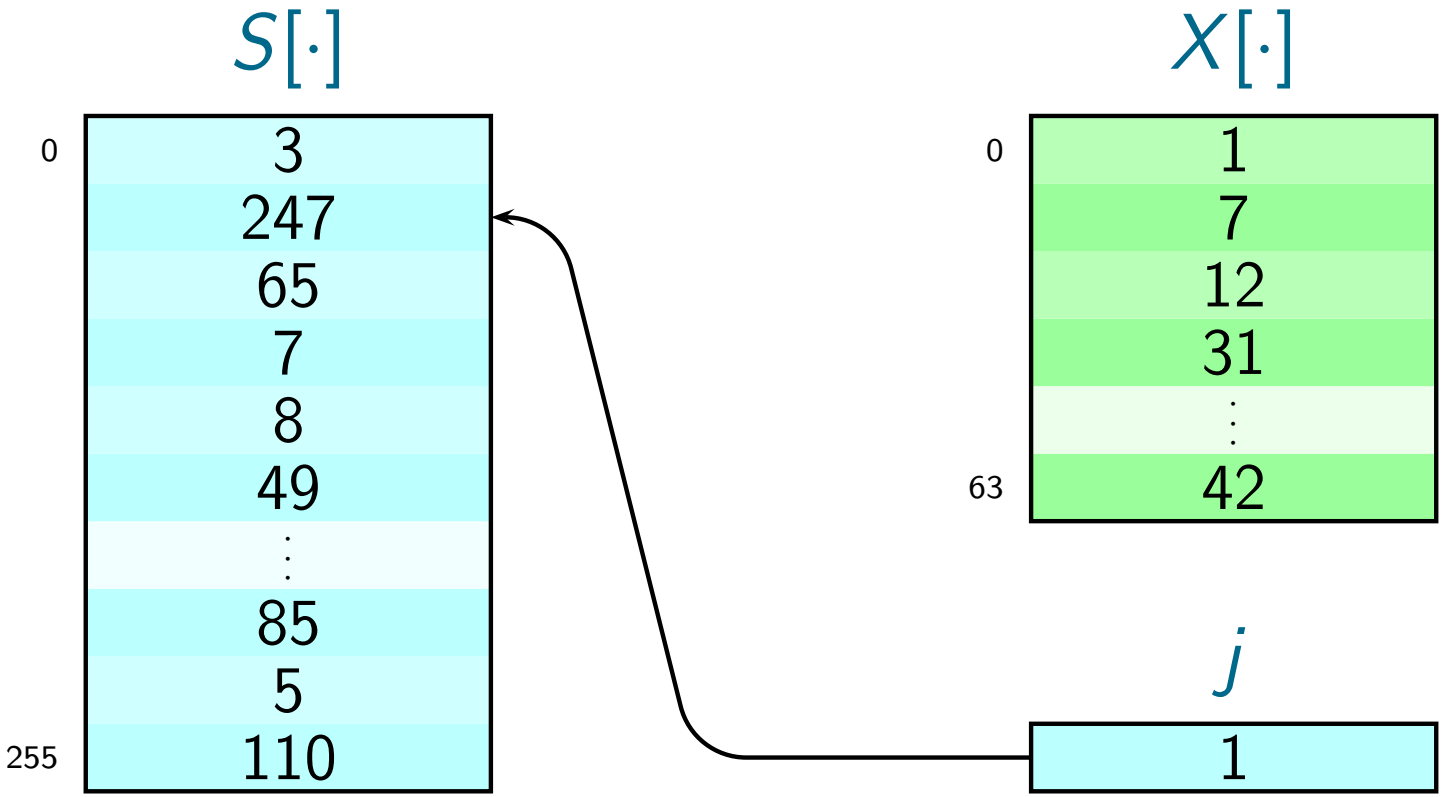




RC4-Hash

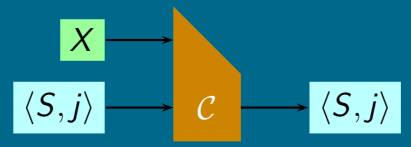
Compression Function

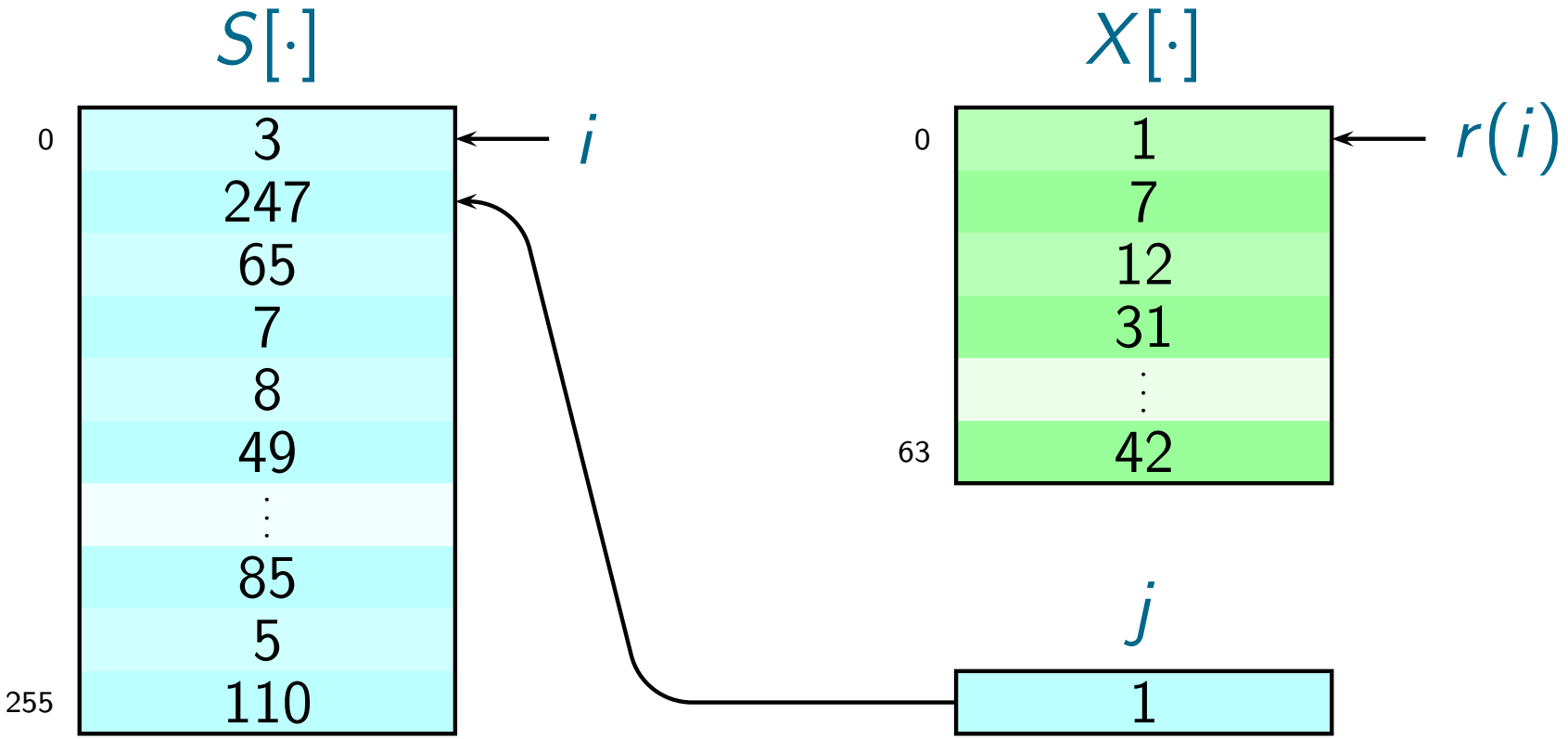




RC4-Hash

Compression Function

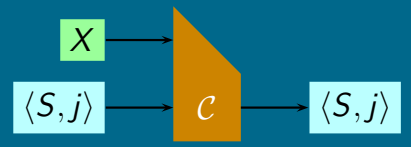


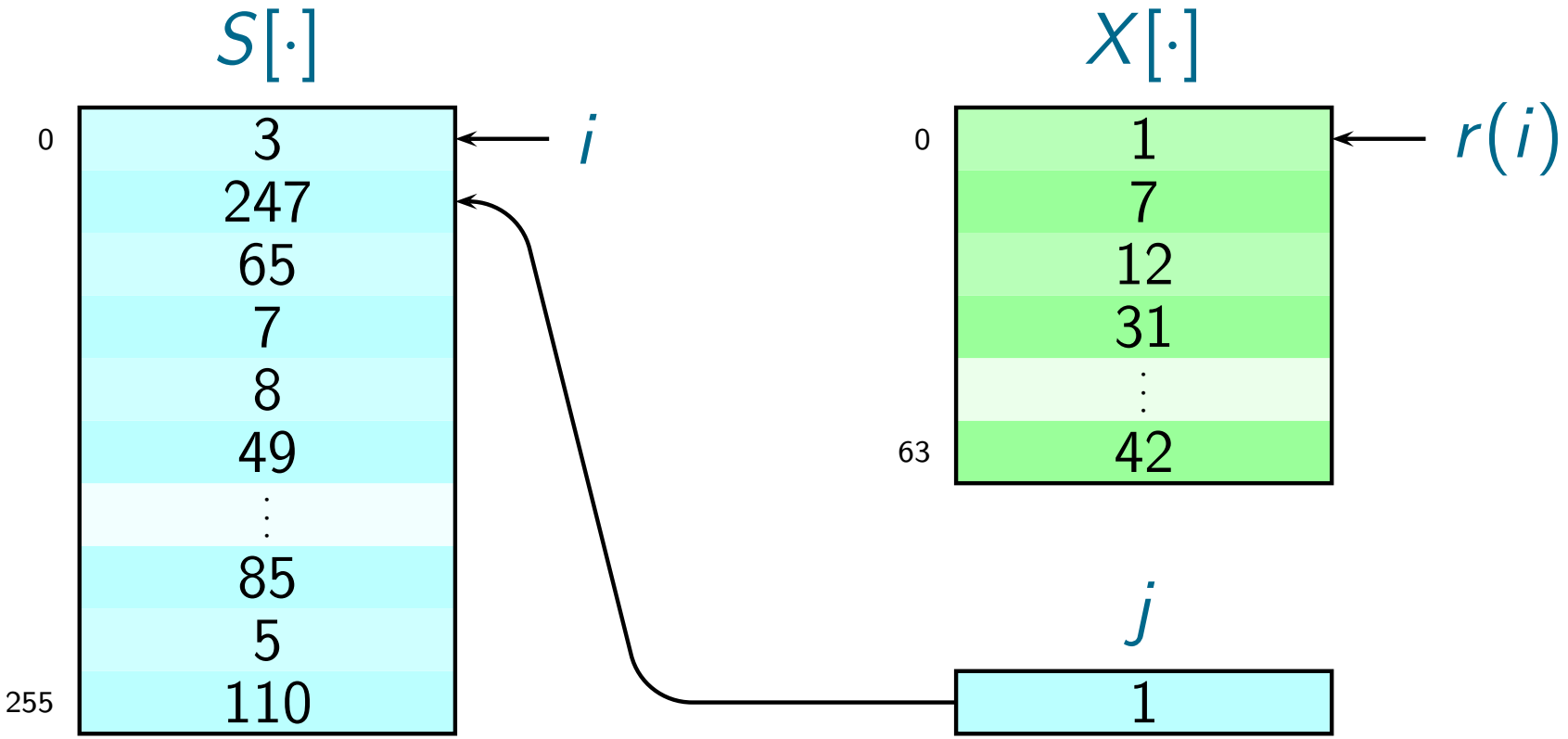


1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

RC4-Hash

Compression Function

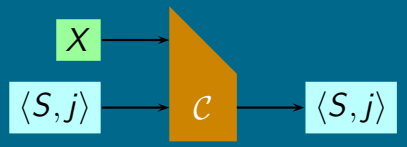


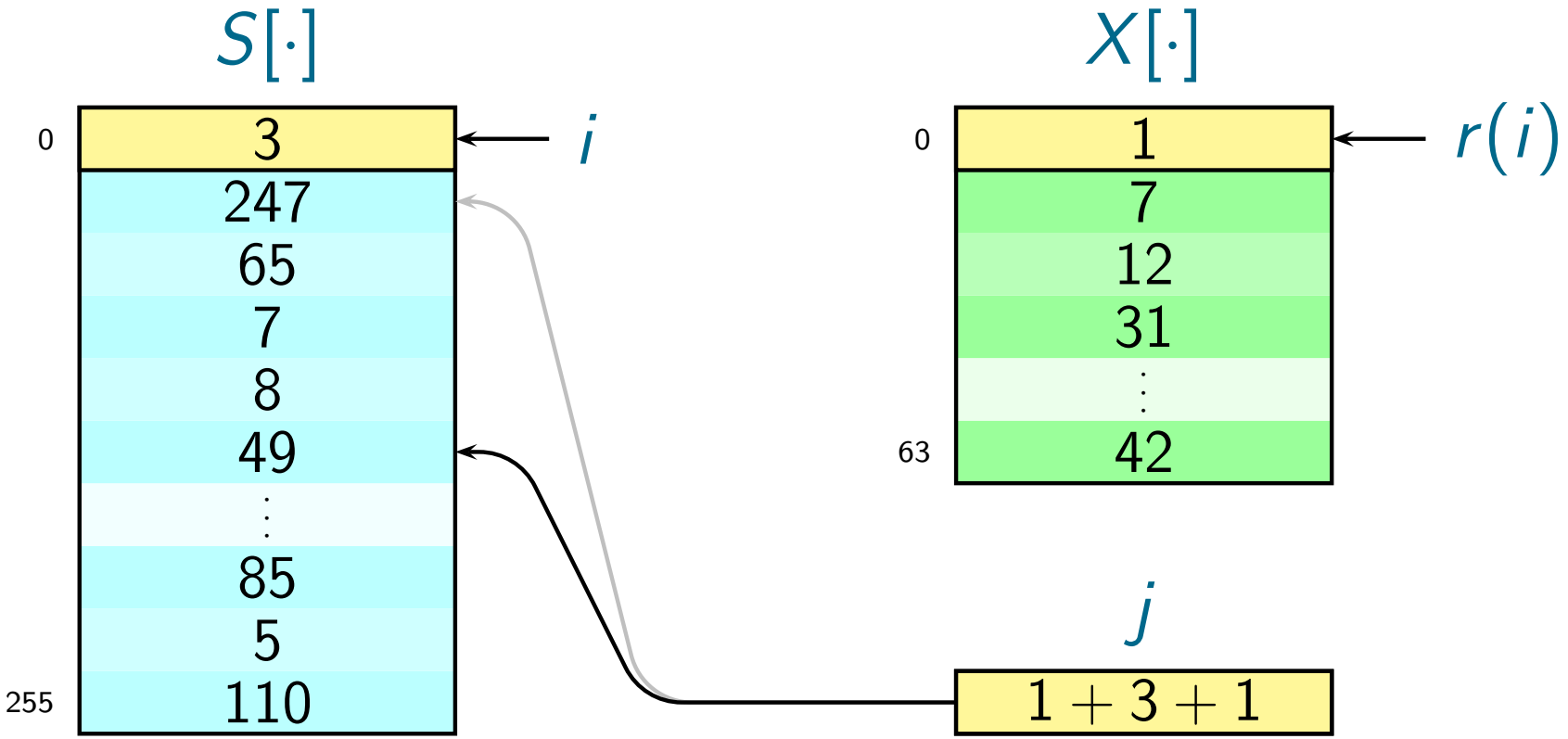


1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

RC4-Hash

Compression Function

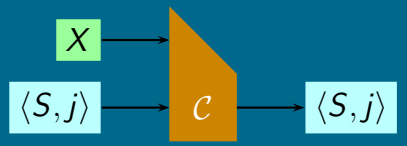


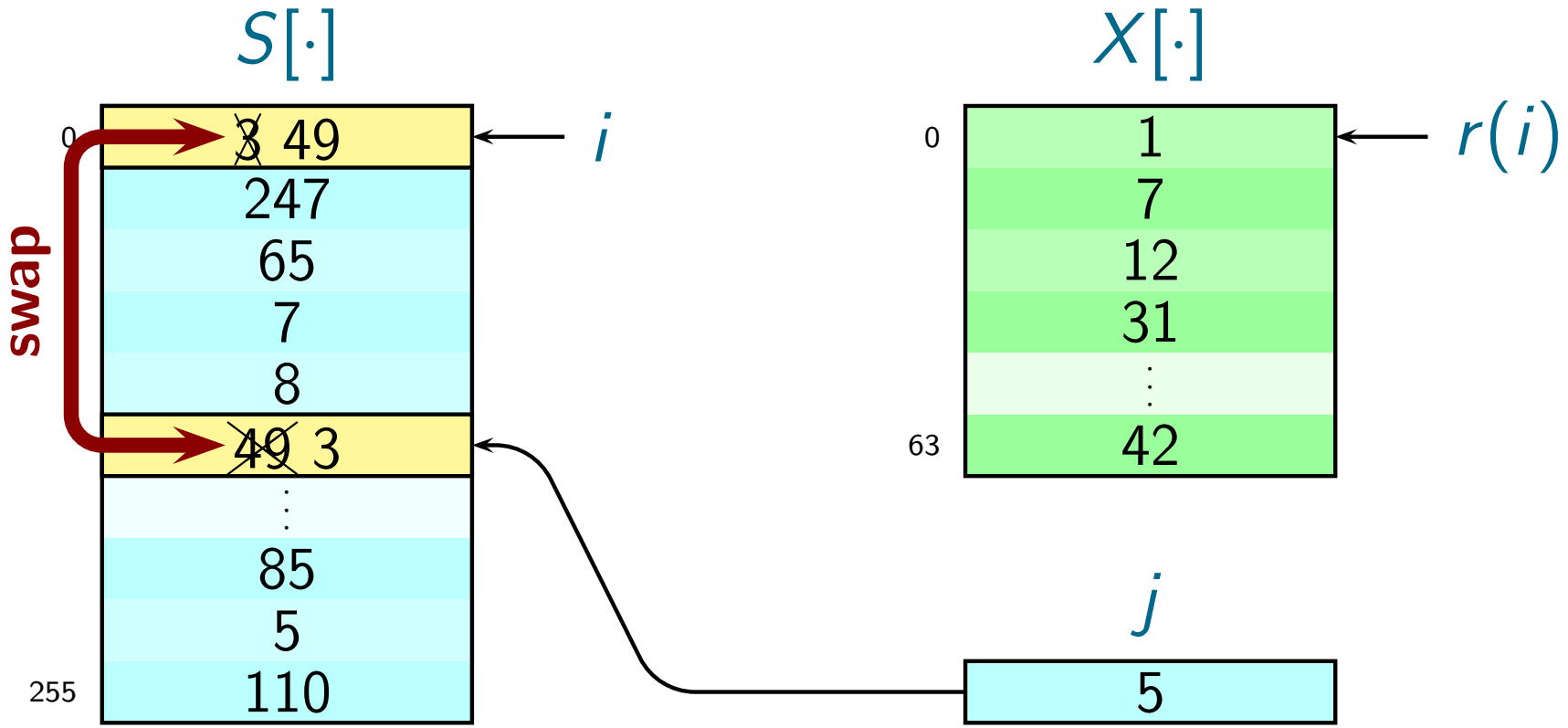


1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

RC4-Hash

Compression Function

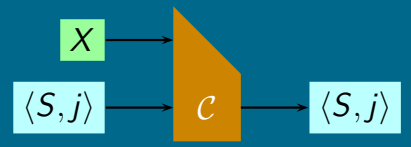


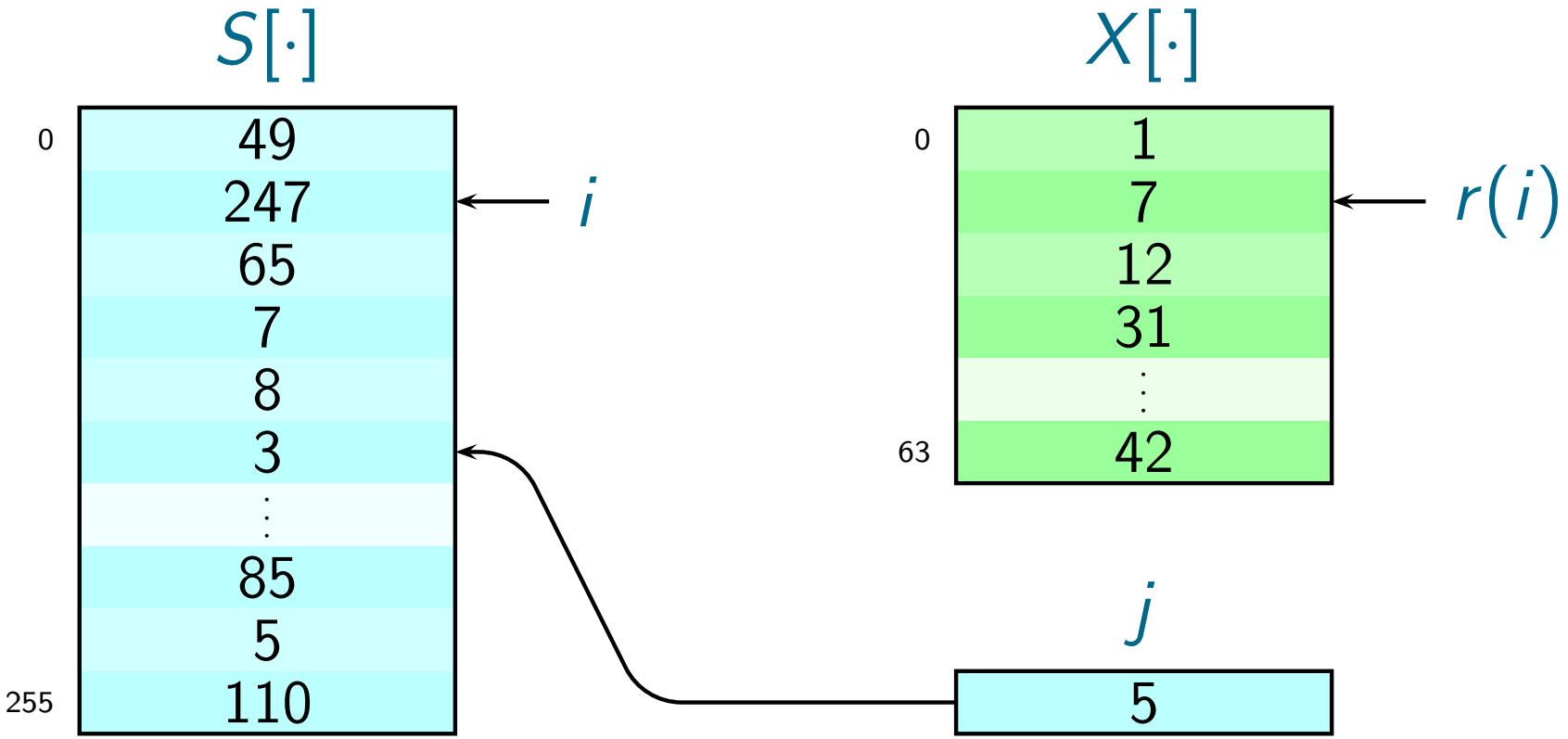


1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

RC4-Hash

Compression Function

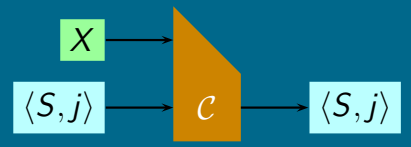


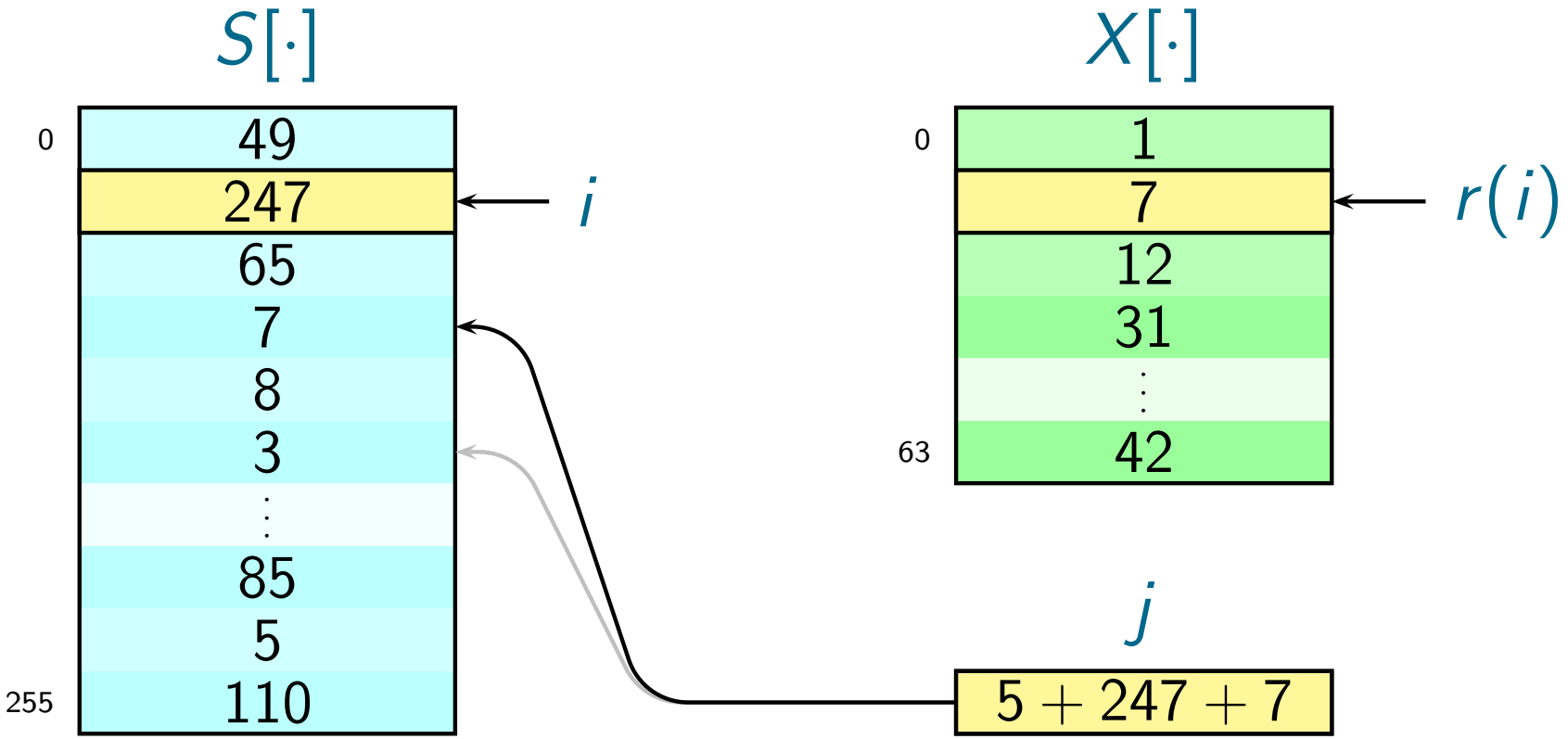


1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

RC4-Hash

Compression Function

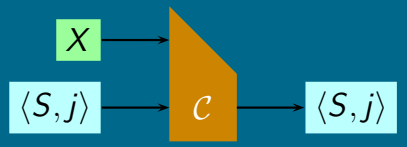


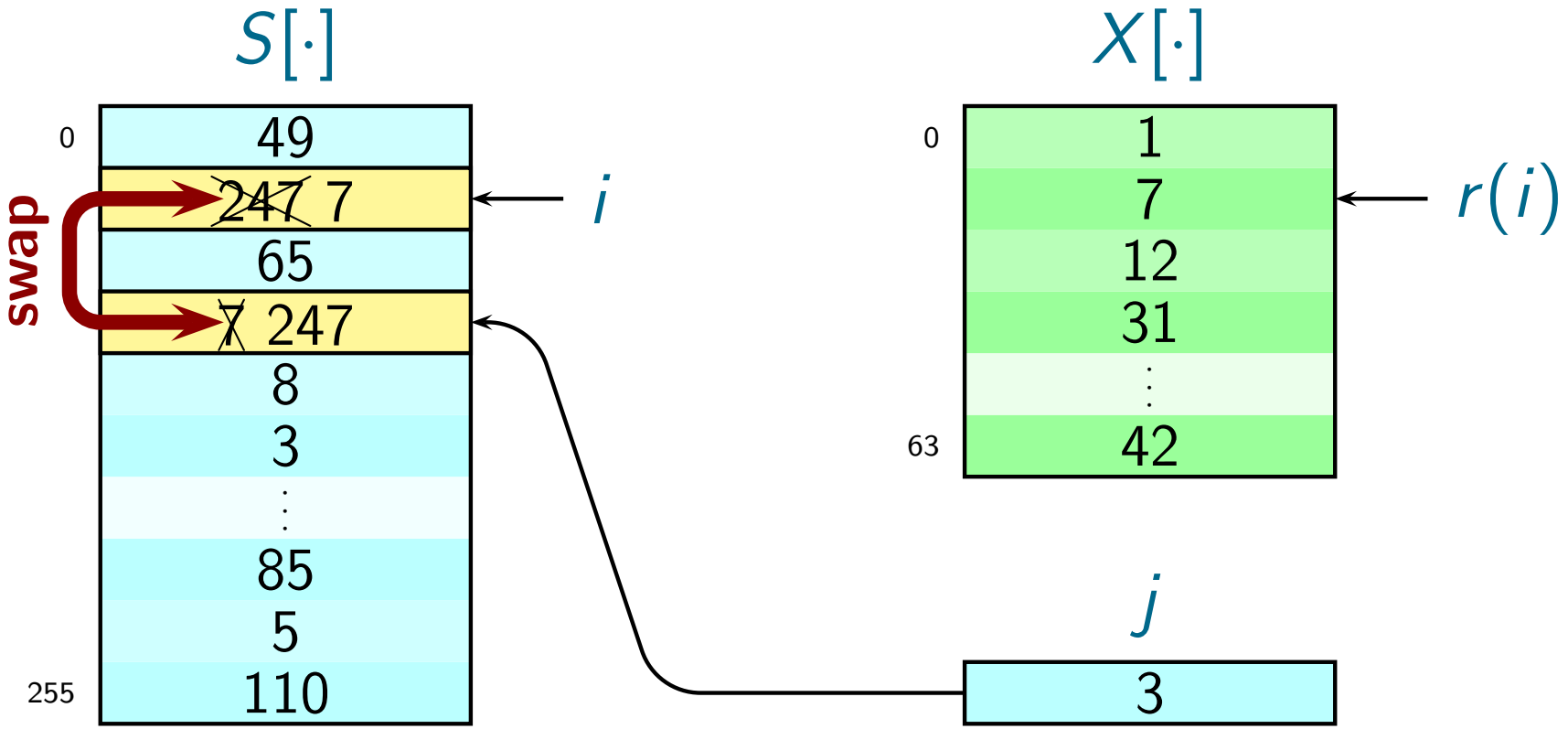


1: **for** $i = 0$ to 255 **do**
2: $j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: swap $S[i], S[j]$
4: **end for**

RC4-Hash

Compression Function

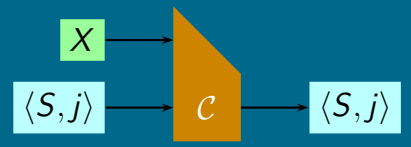


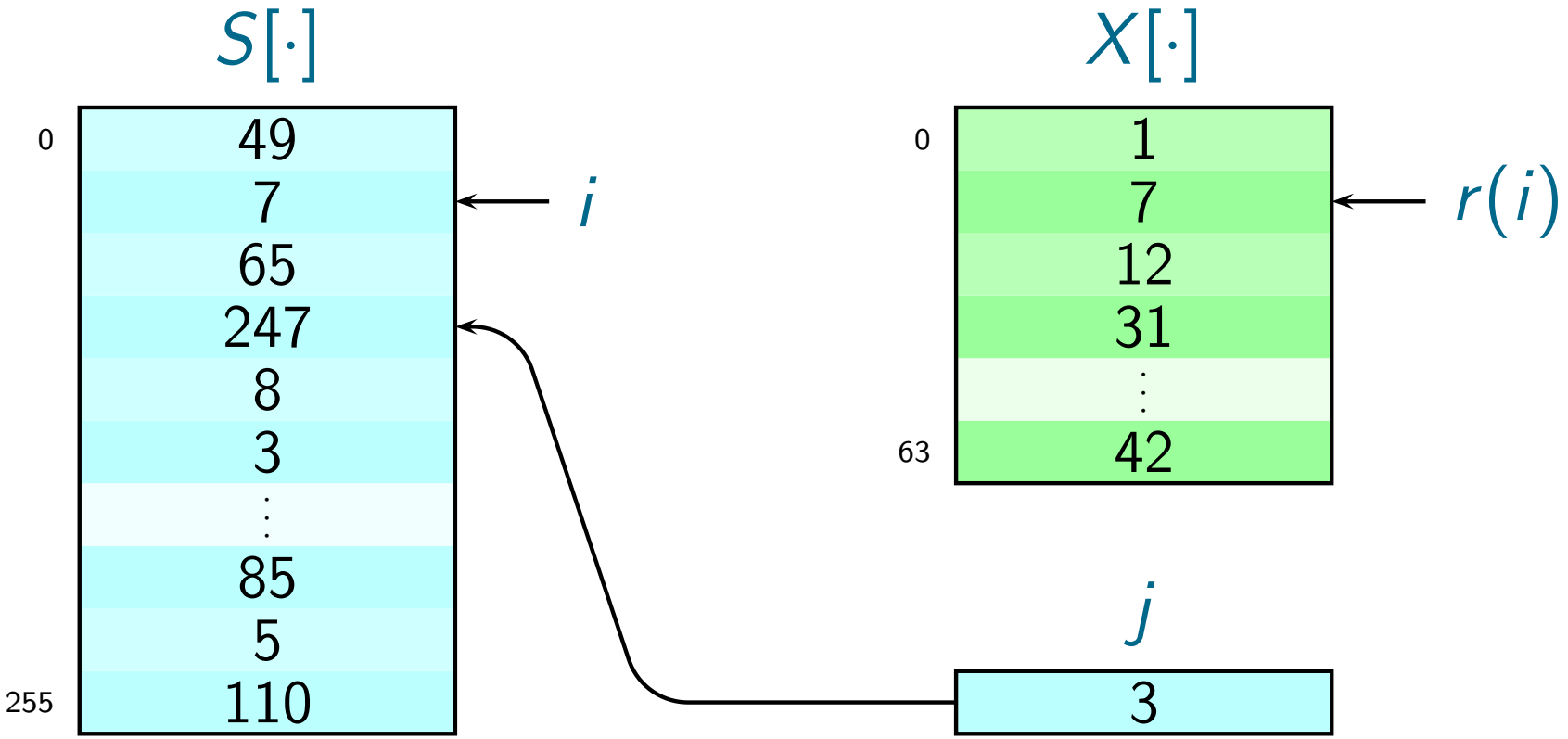


1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

RC4-Hash

Compression Function



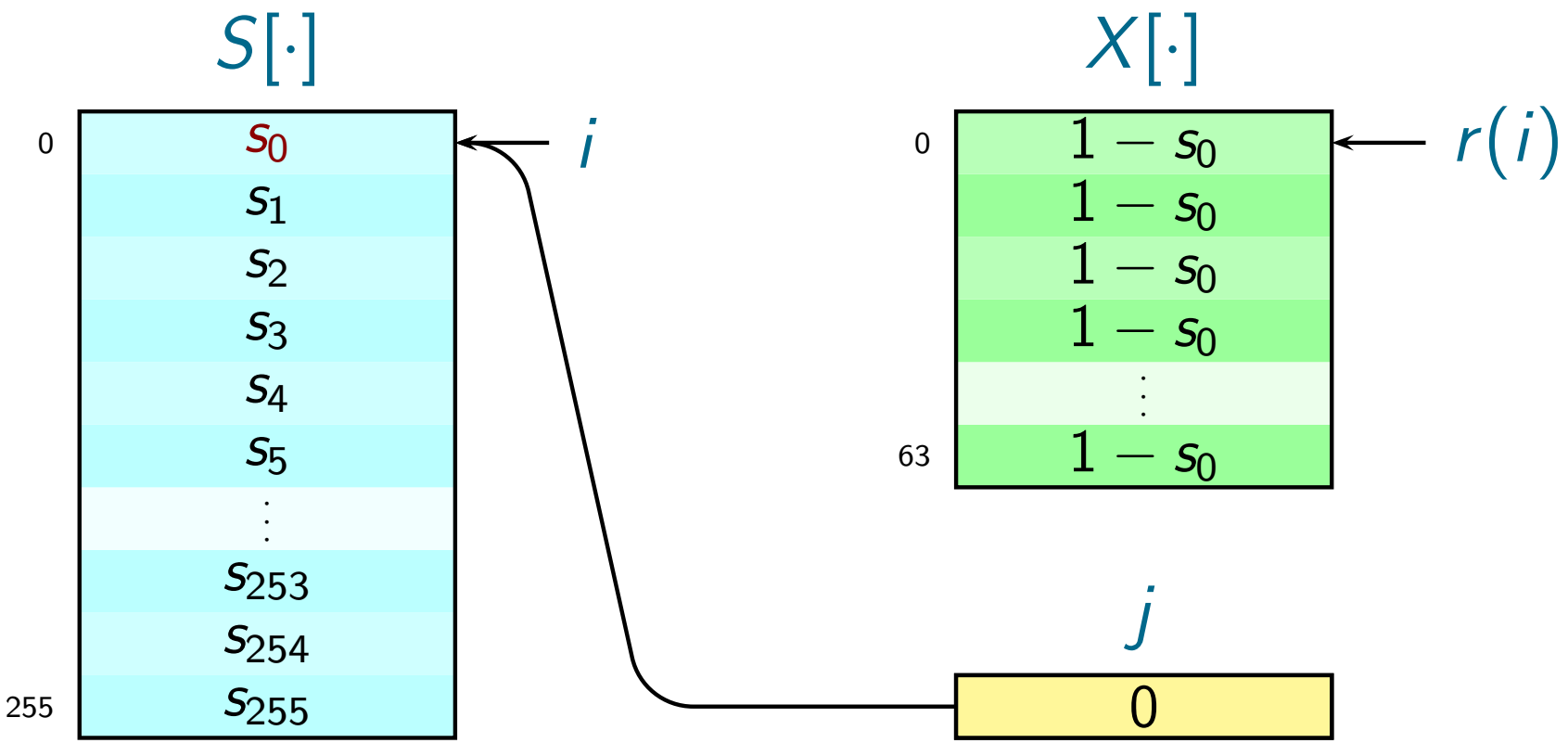


1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Outline

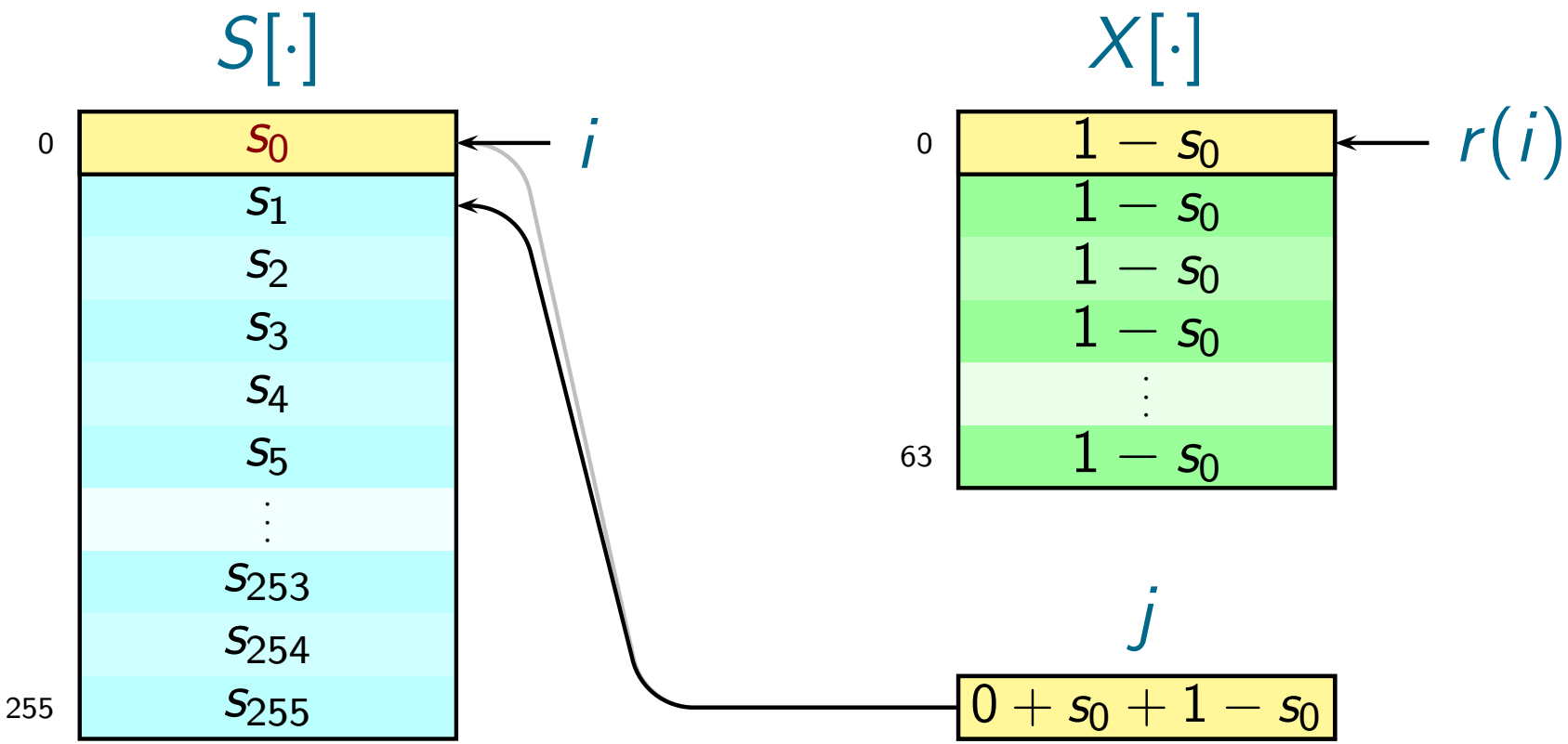
1. Introduction
   - Cryptographic Hash Functions
   - RC4-Hash
2. **Fixed Points for RC4-Hash**
   - Type I
   - Type II
3. Collisions for RC4-Hash
4. Conclusion

Partial State Rotations of Type I



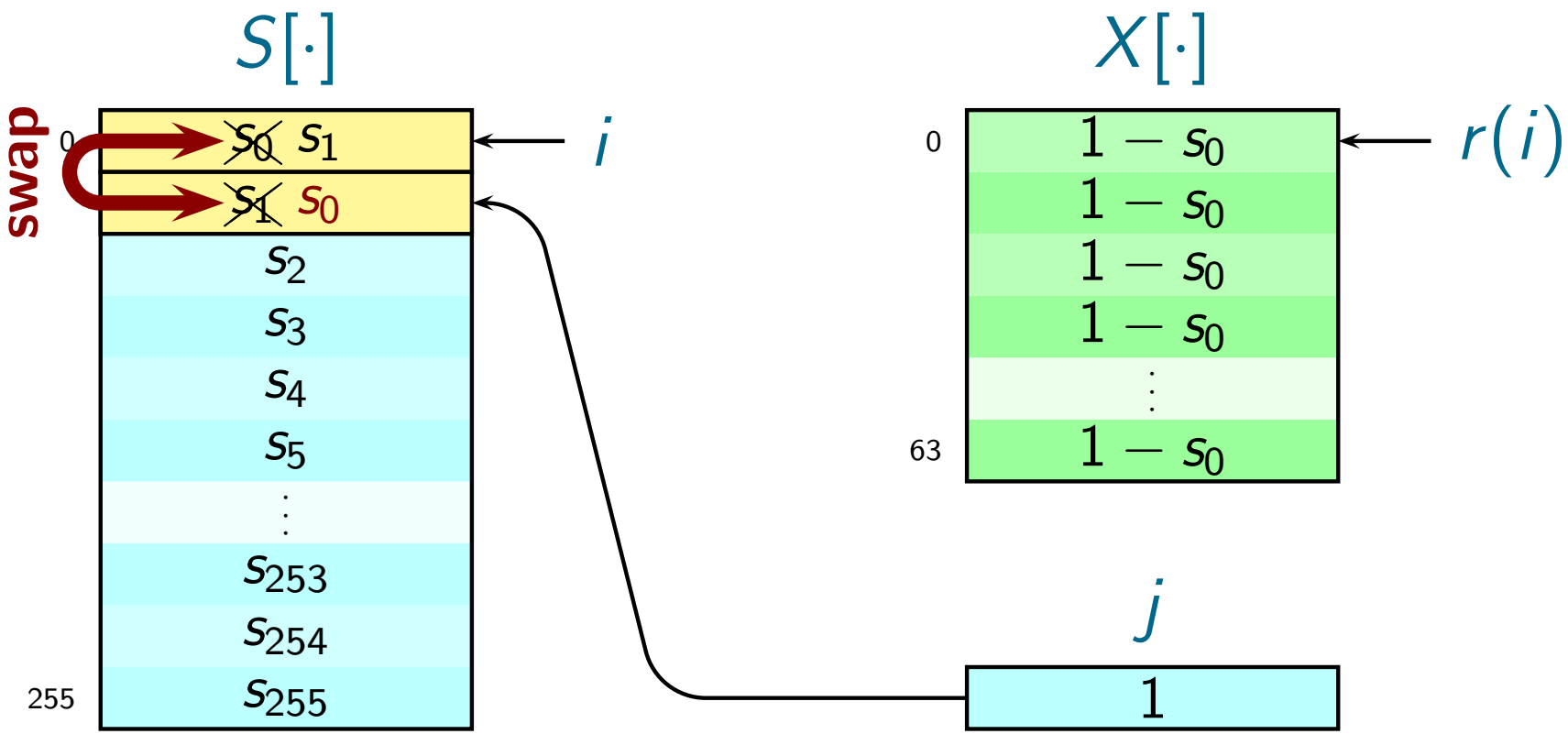
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type I



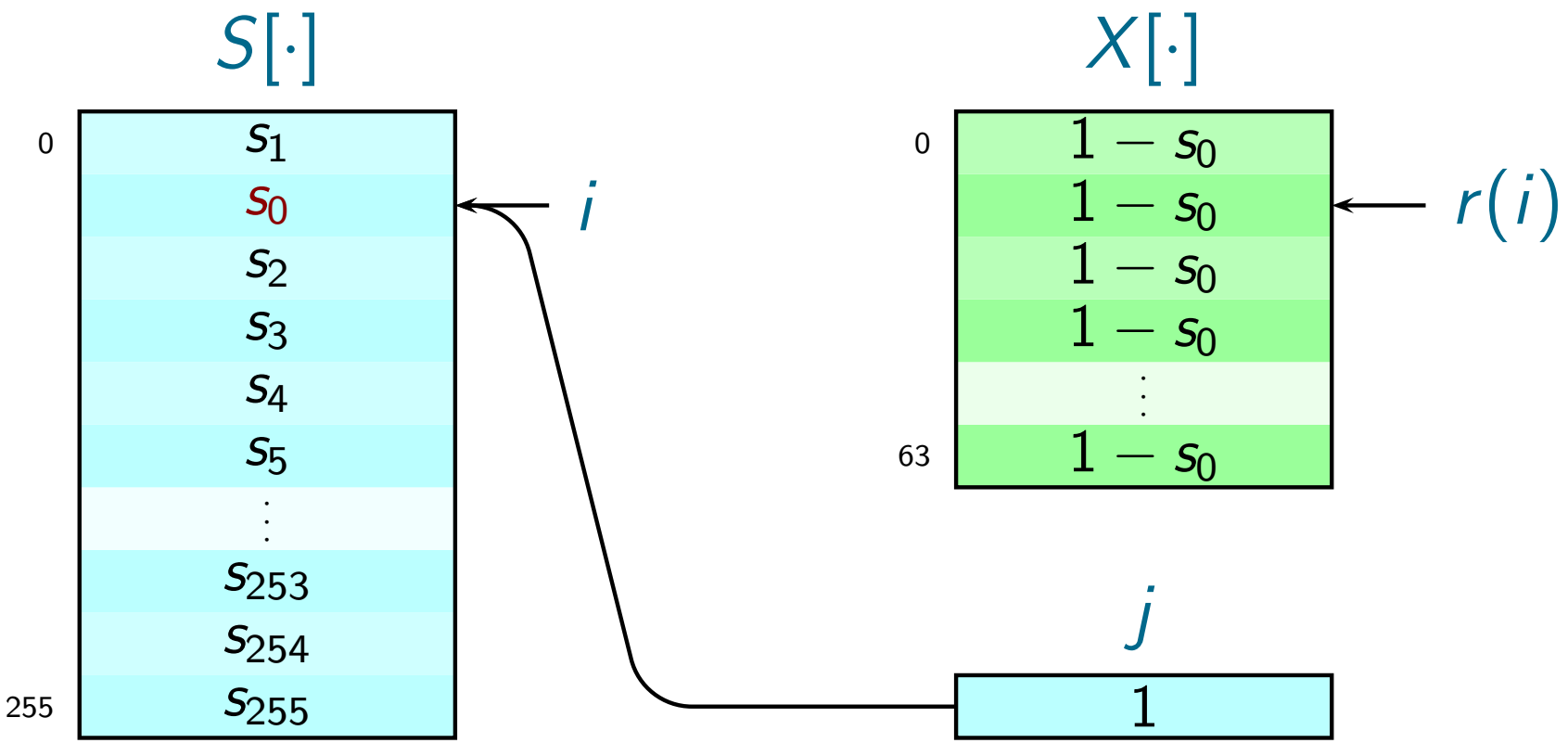
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type I



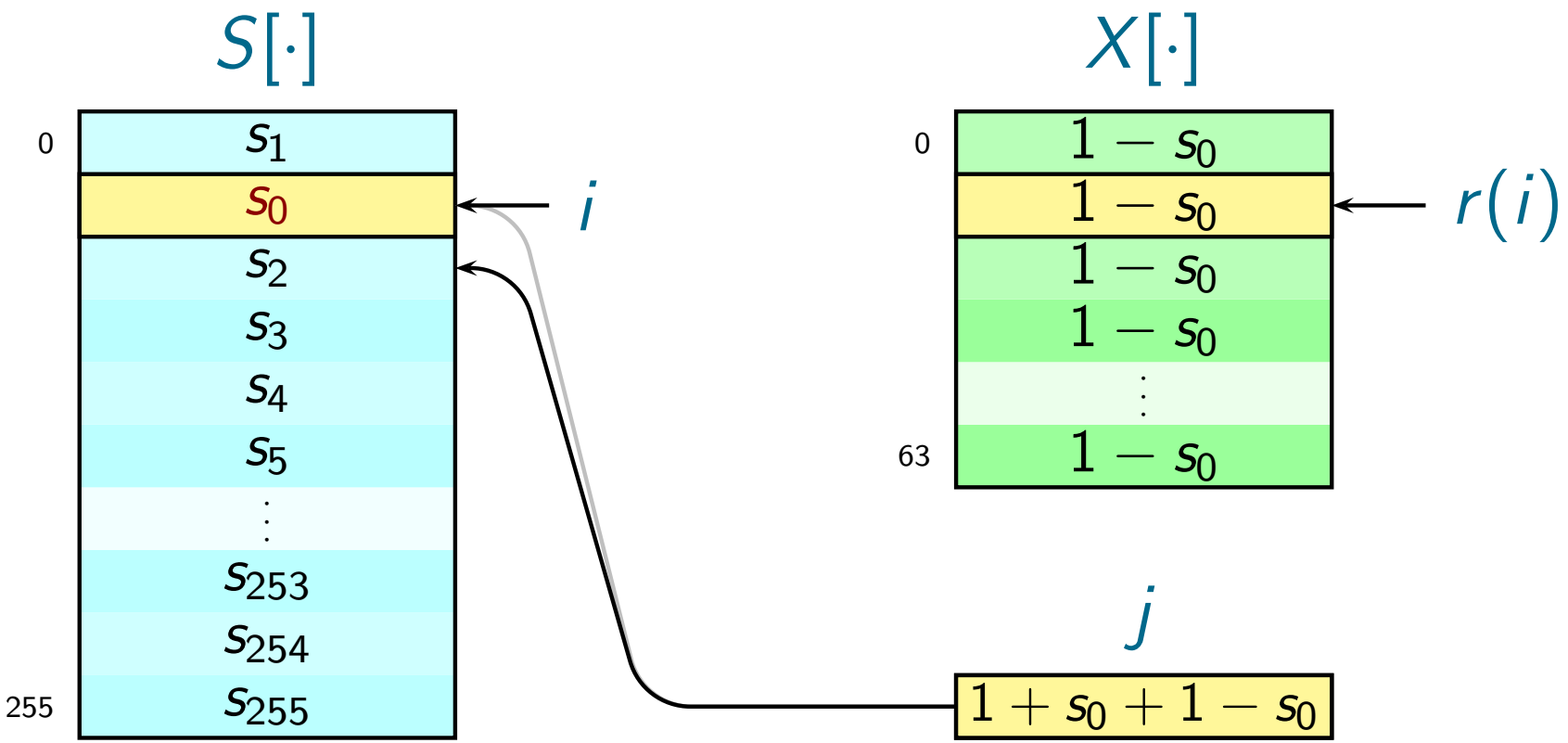
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type I



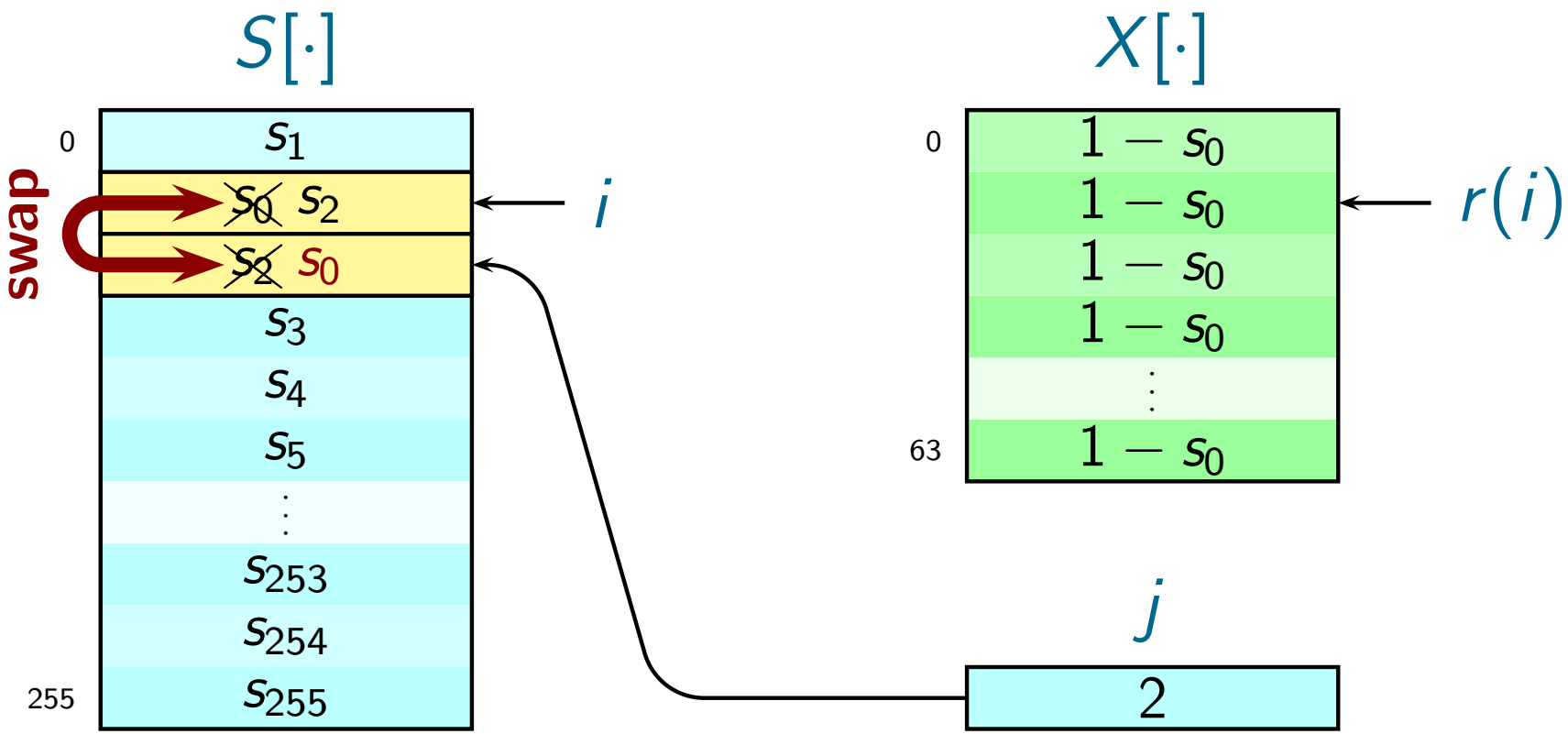
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type I



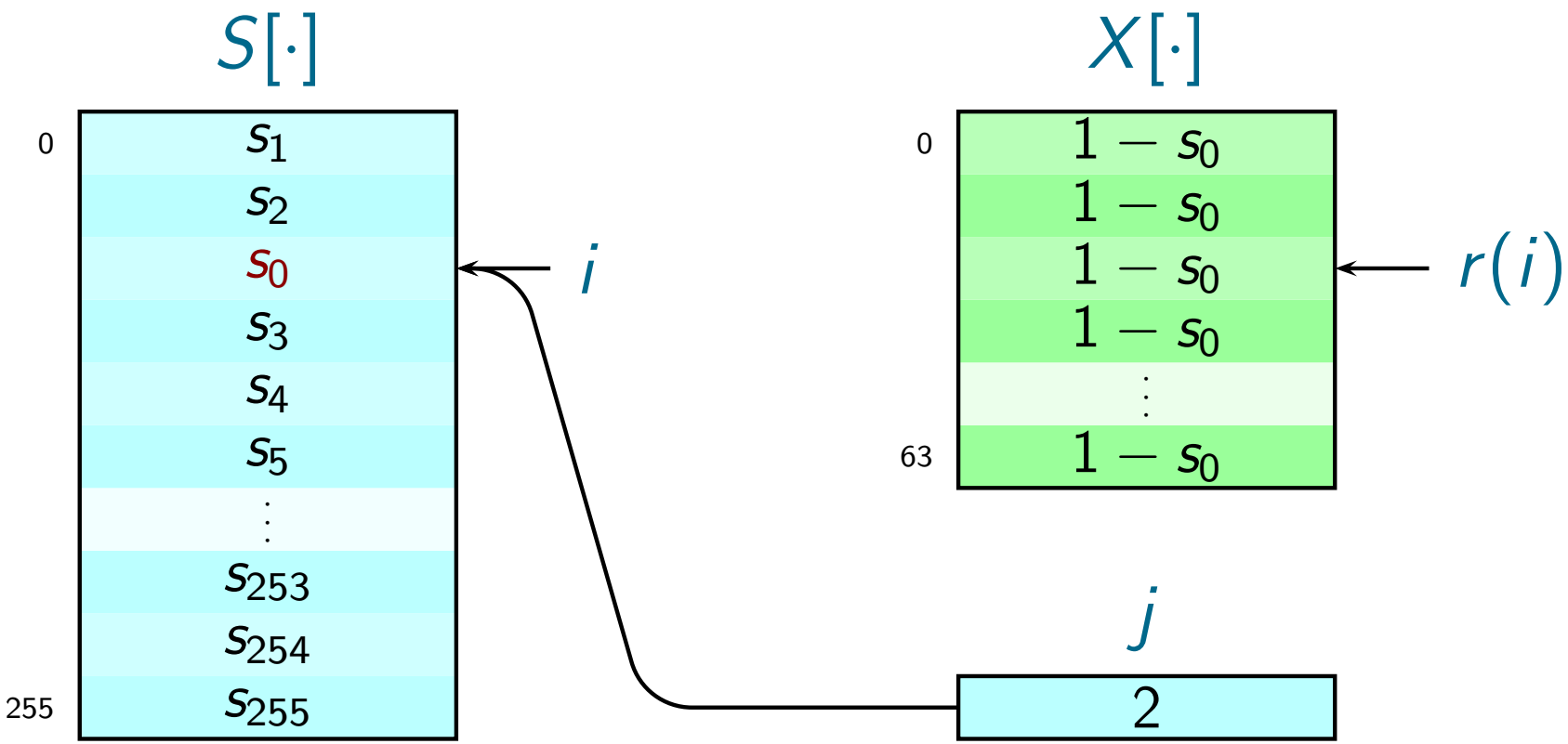
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type I



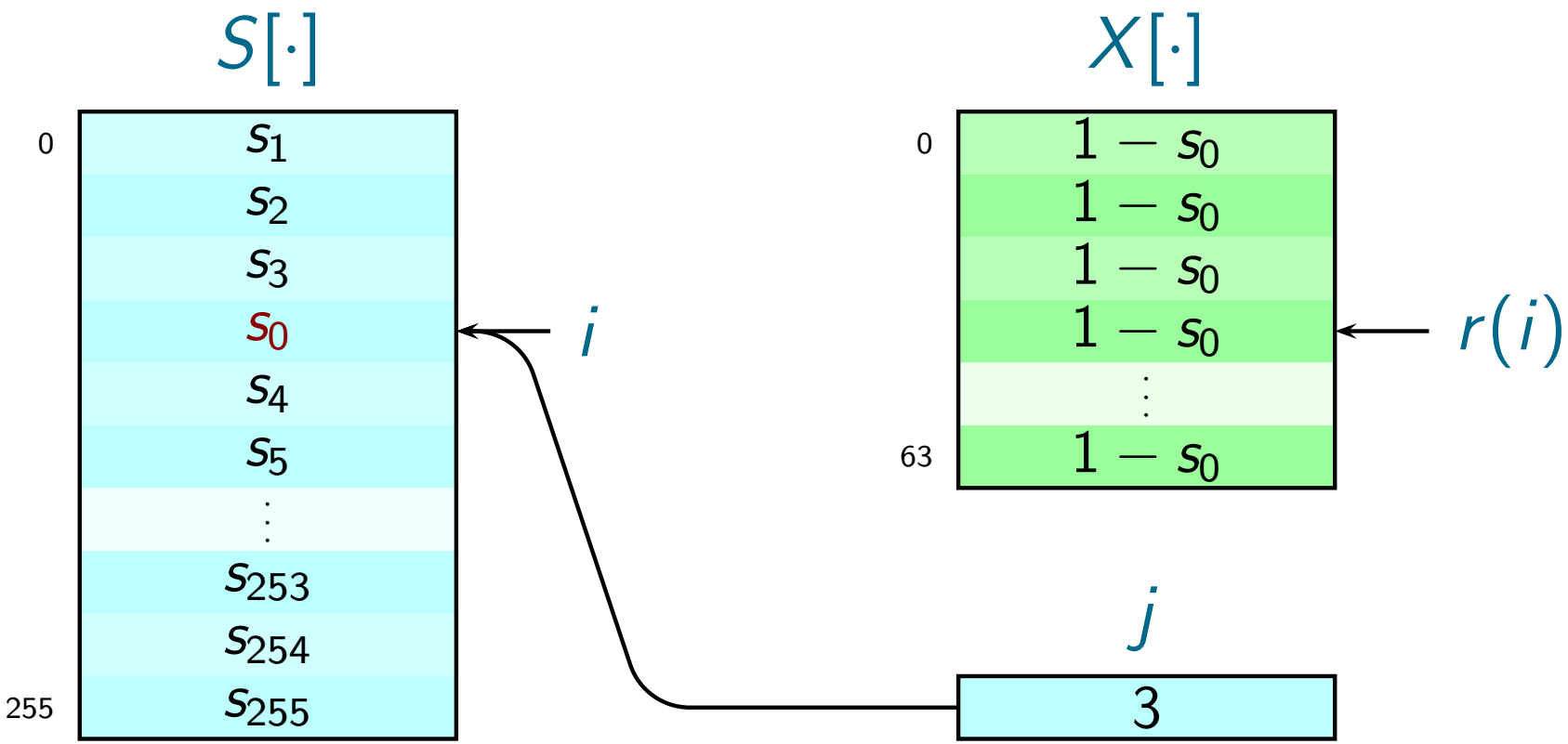
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type I



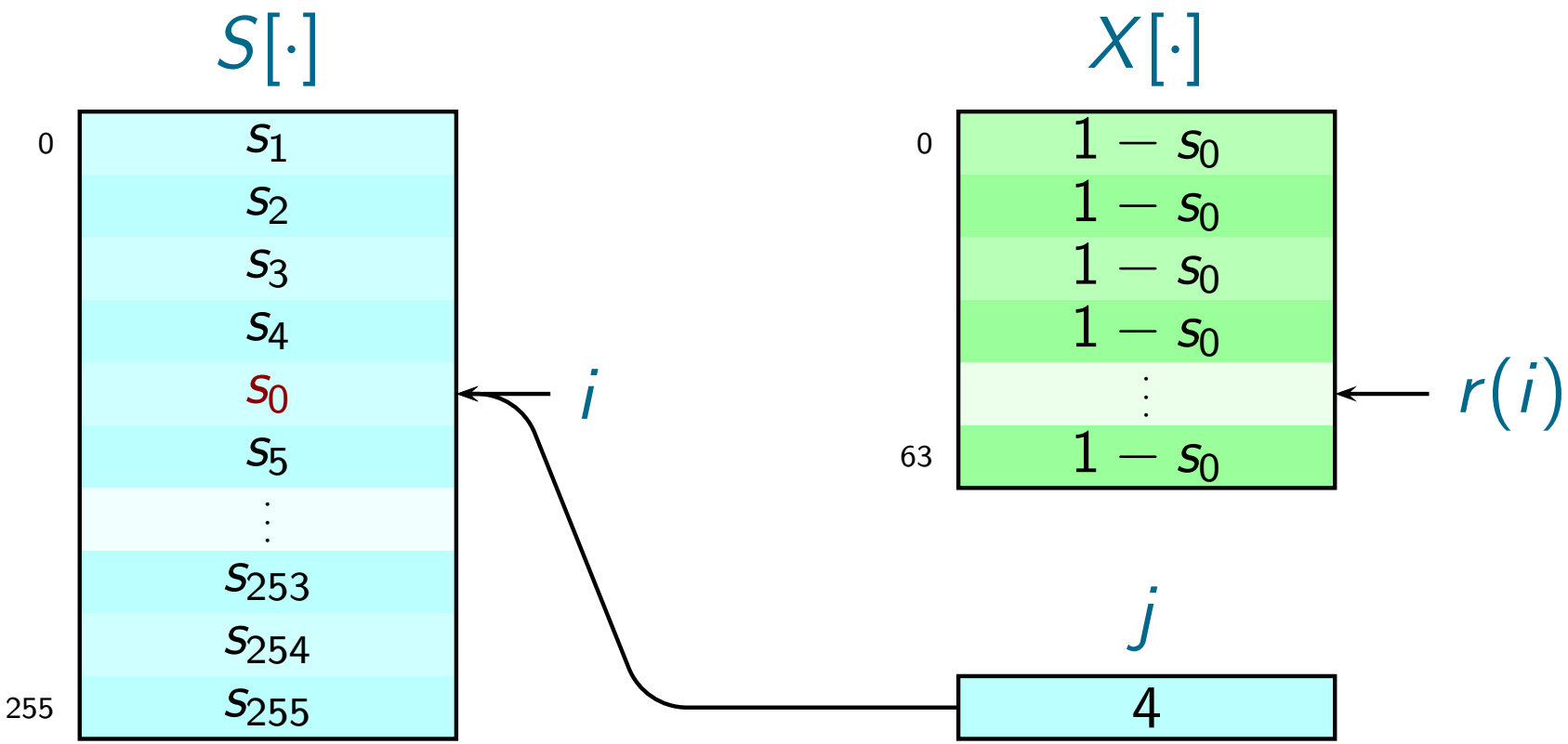
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type I



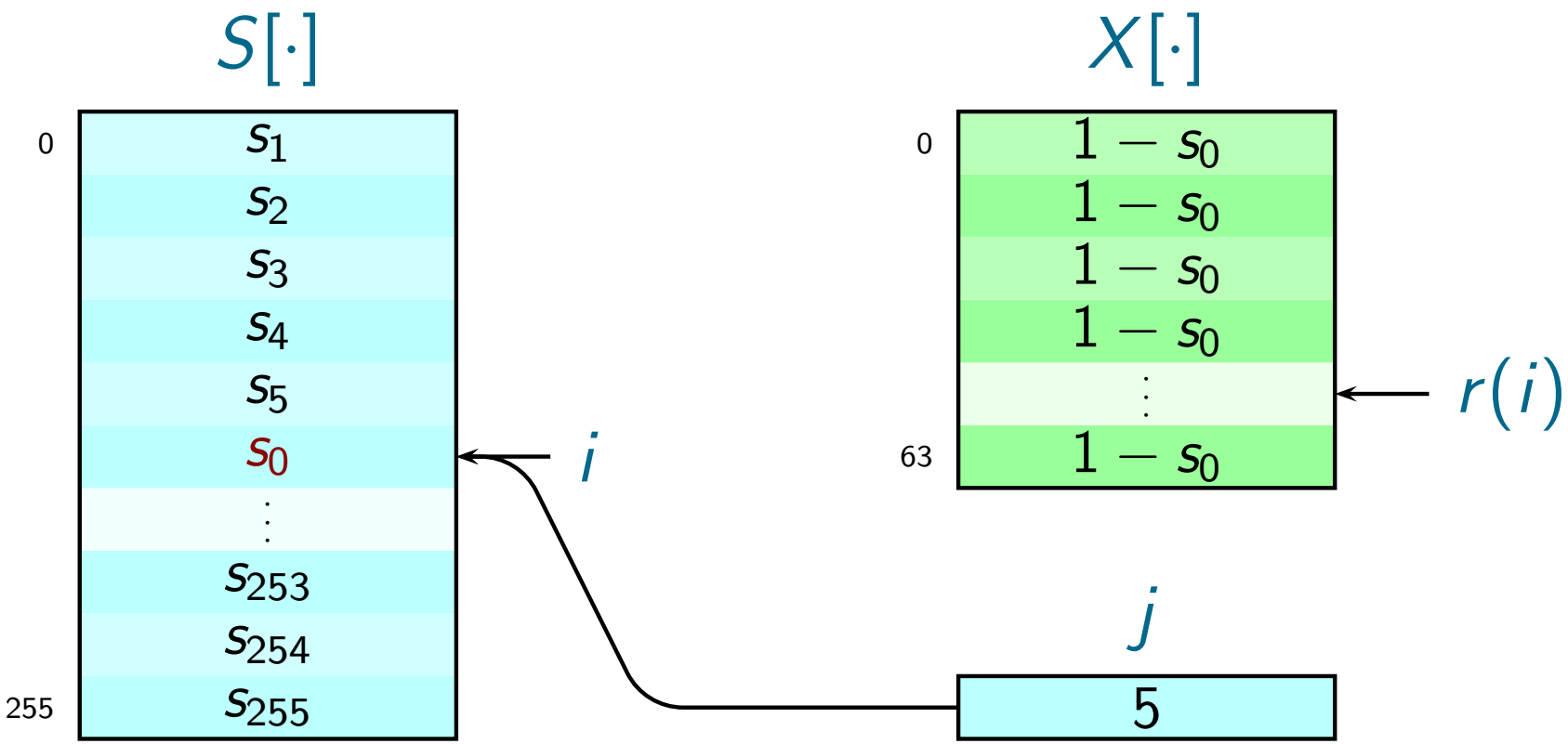
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type I



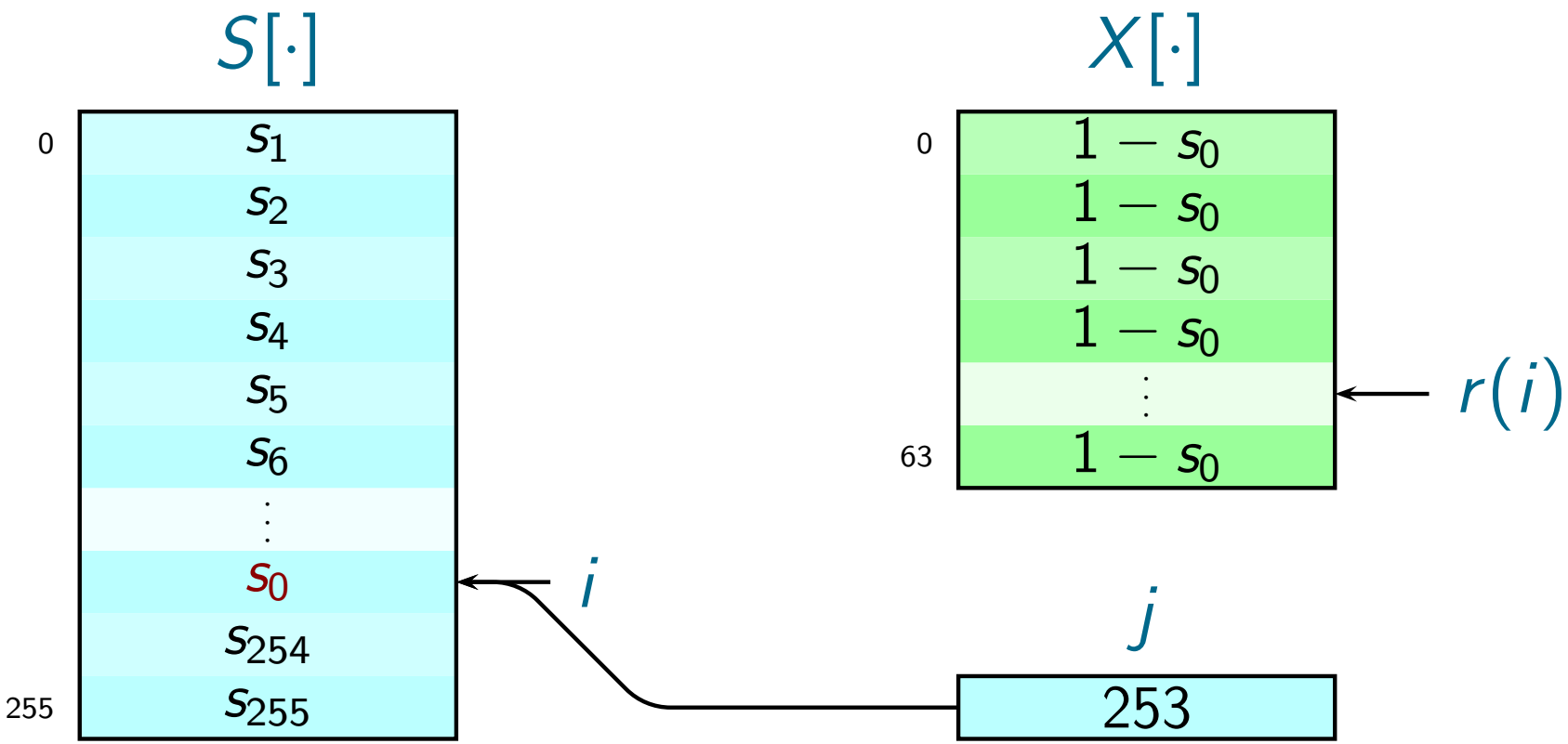
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type I



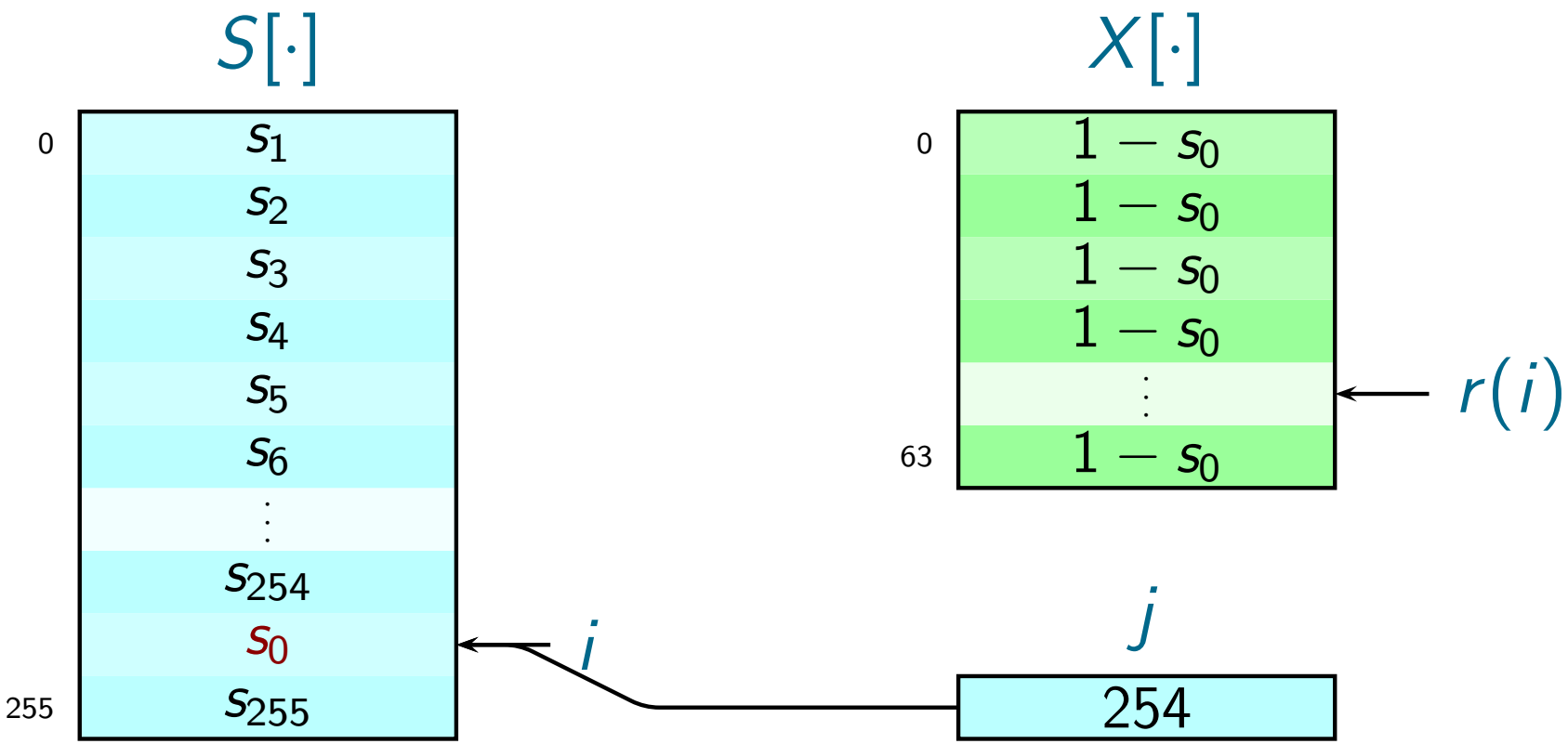
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type I



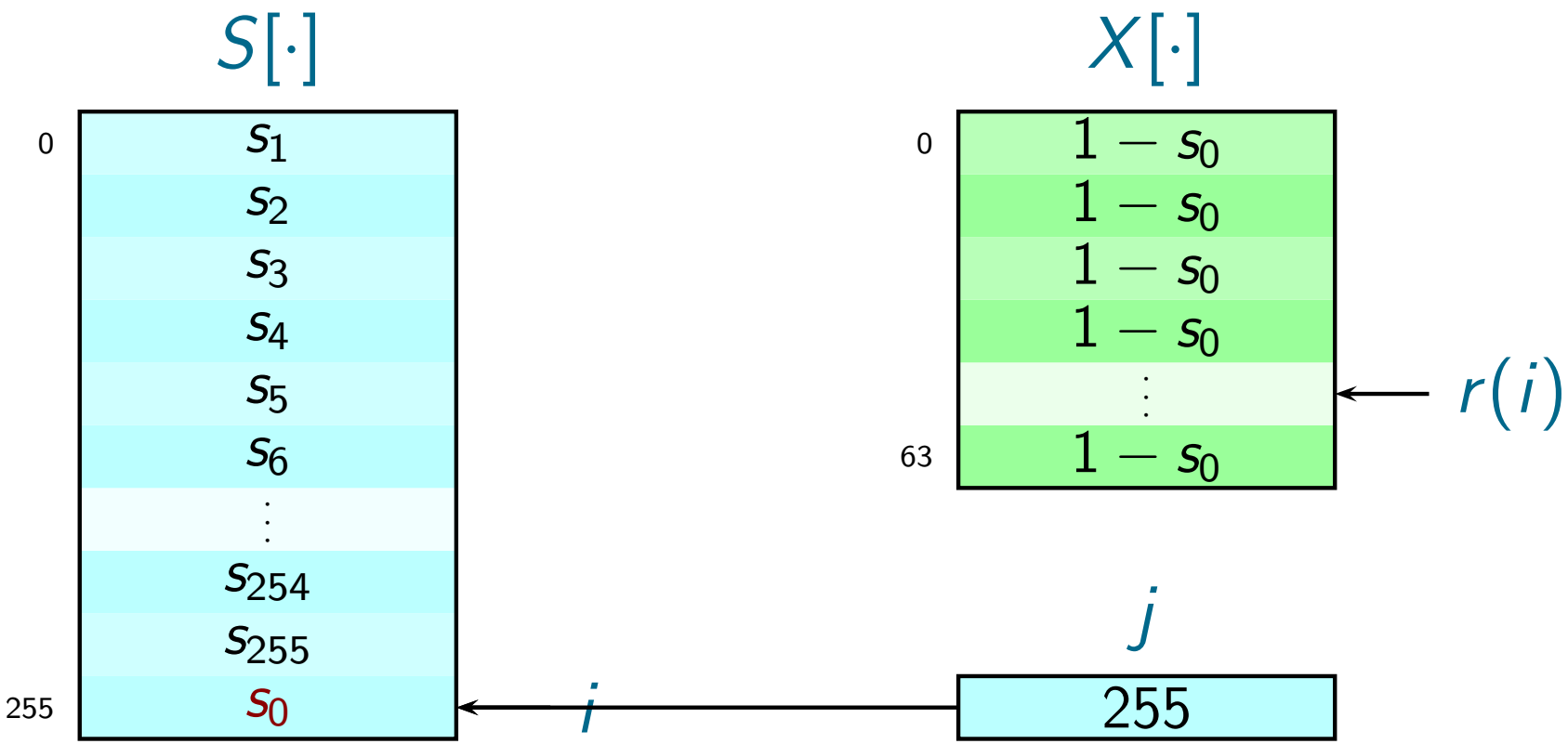
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type I



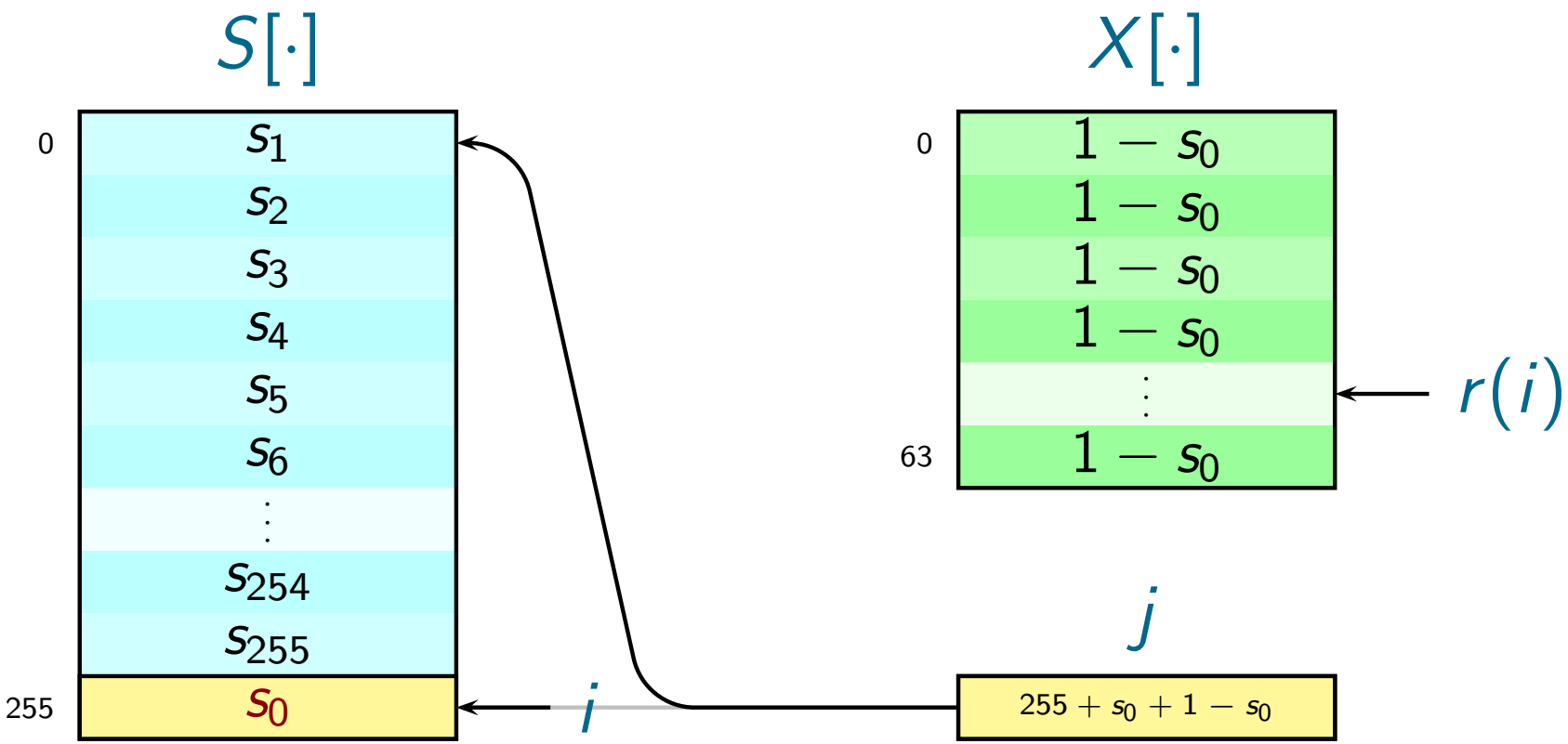
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type I



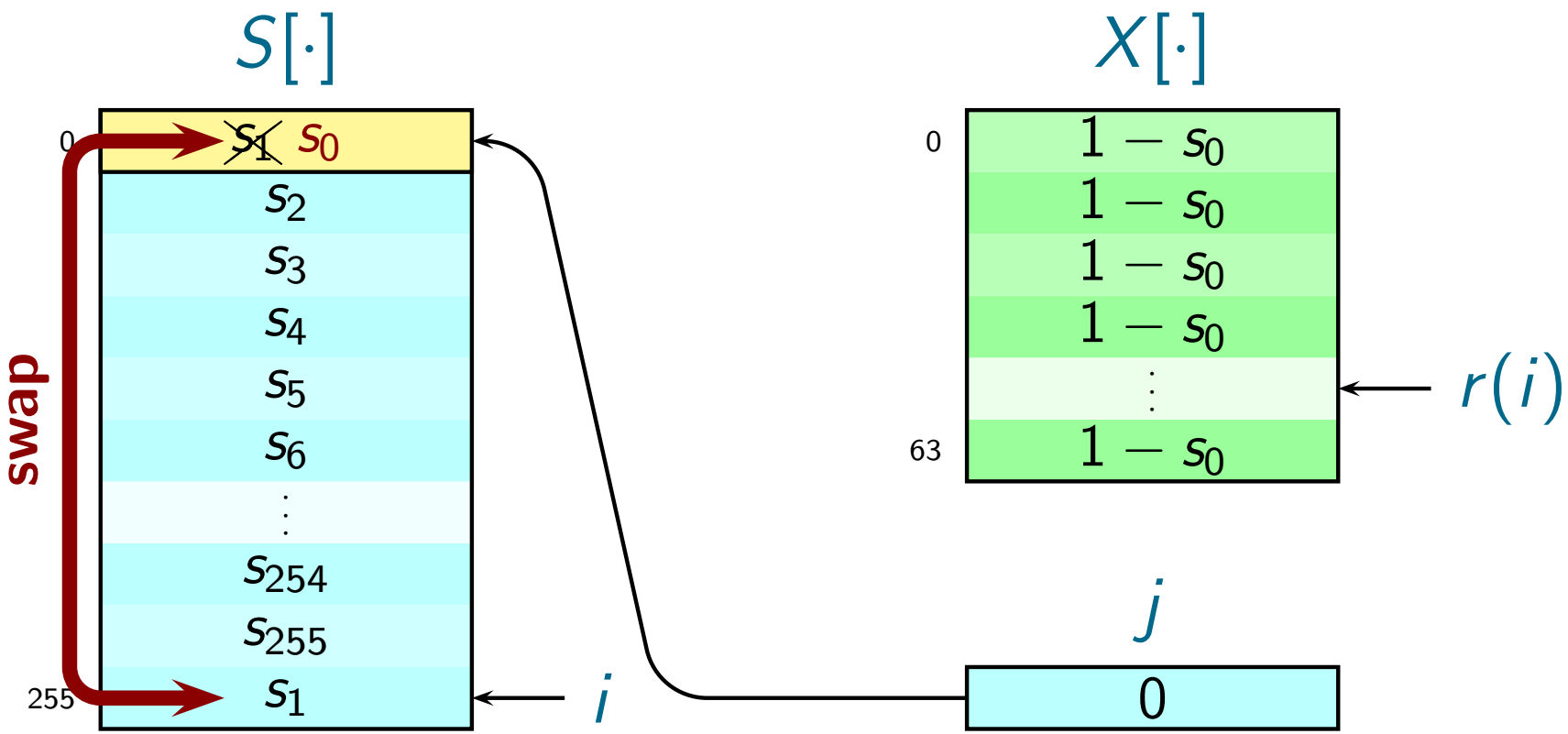
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type I



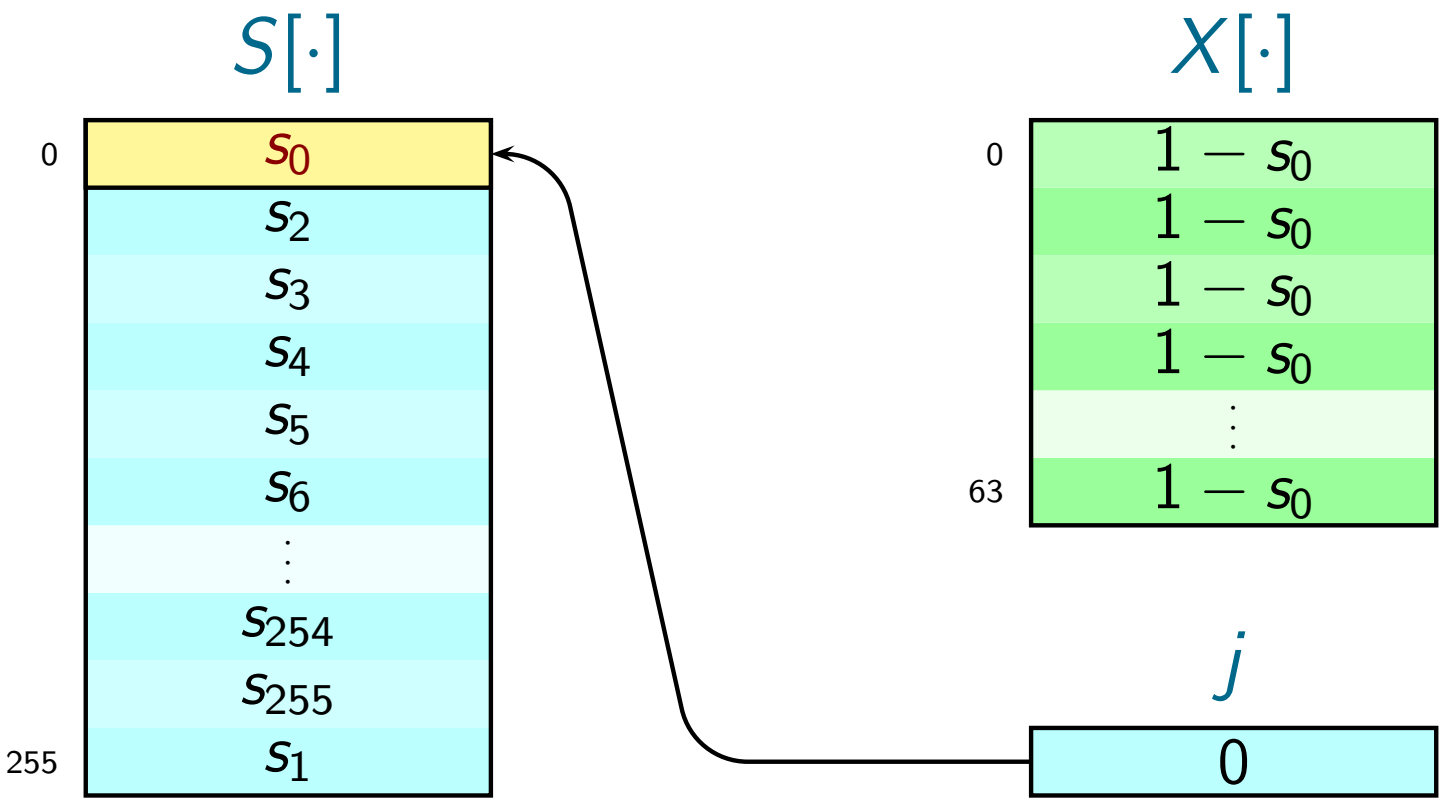
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type I



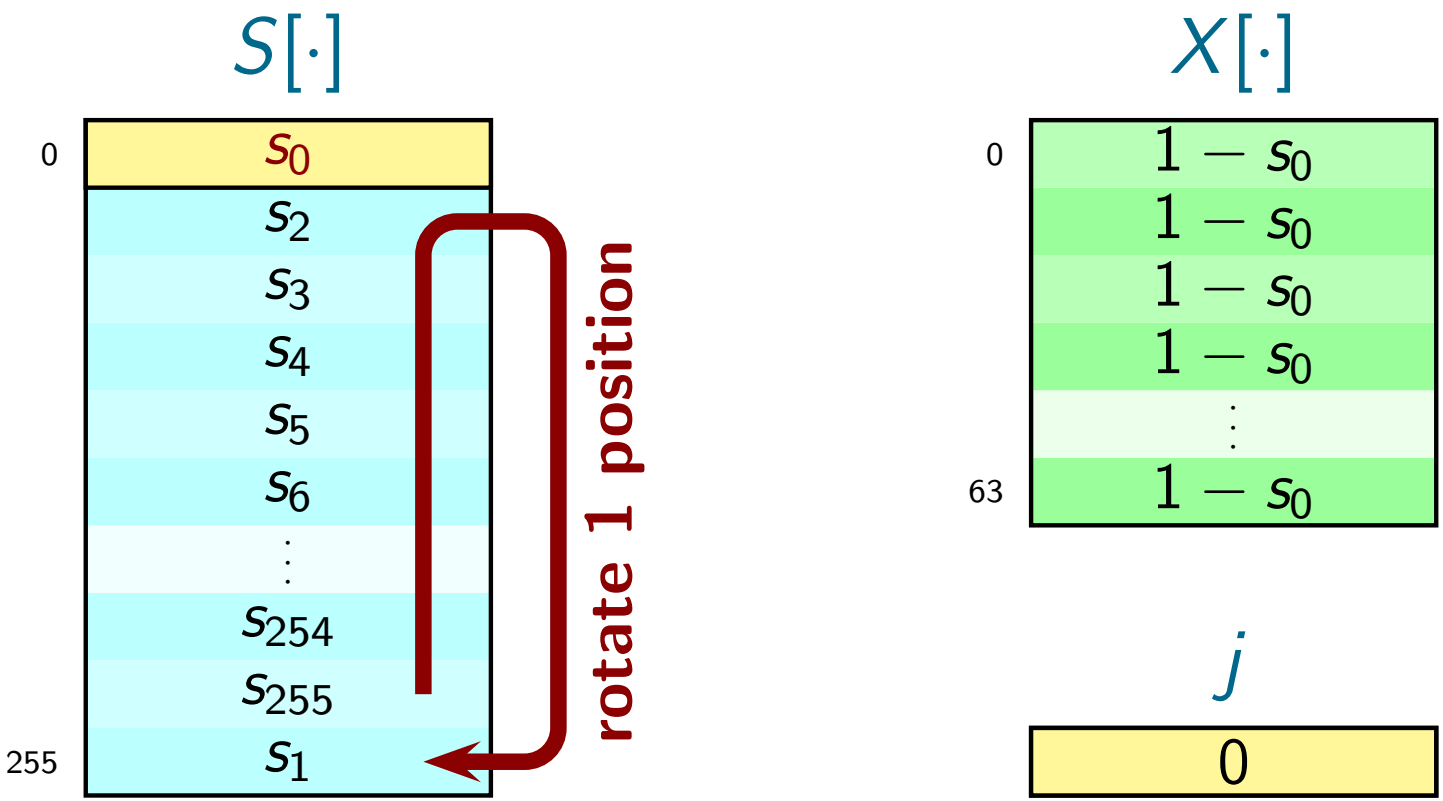
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type I



1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

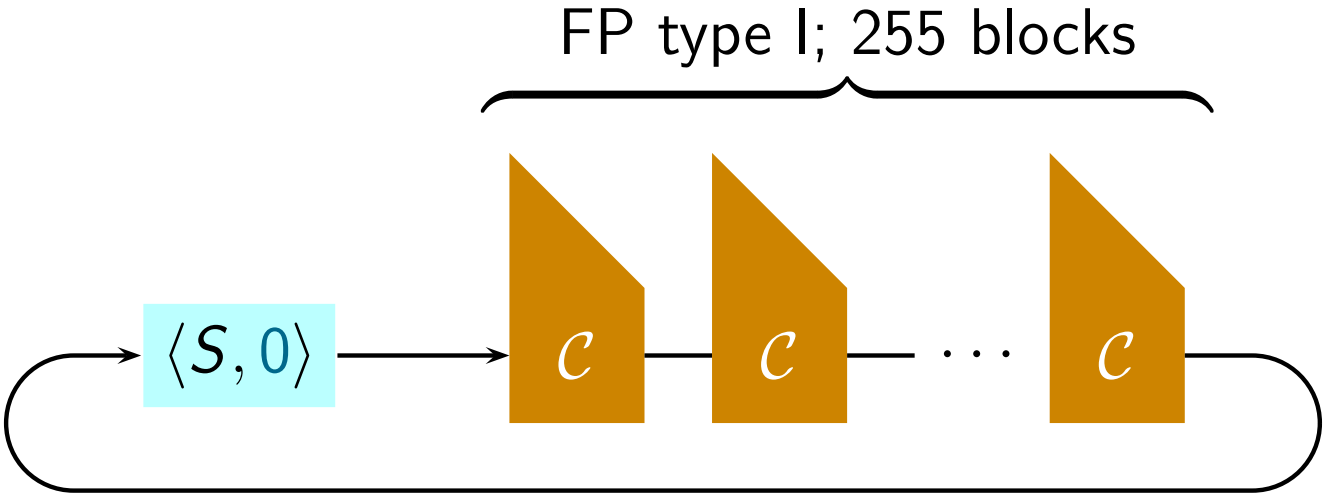
Partial State Rotations of Type I



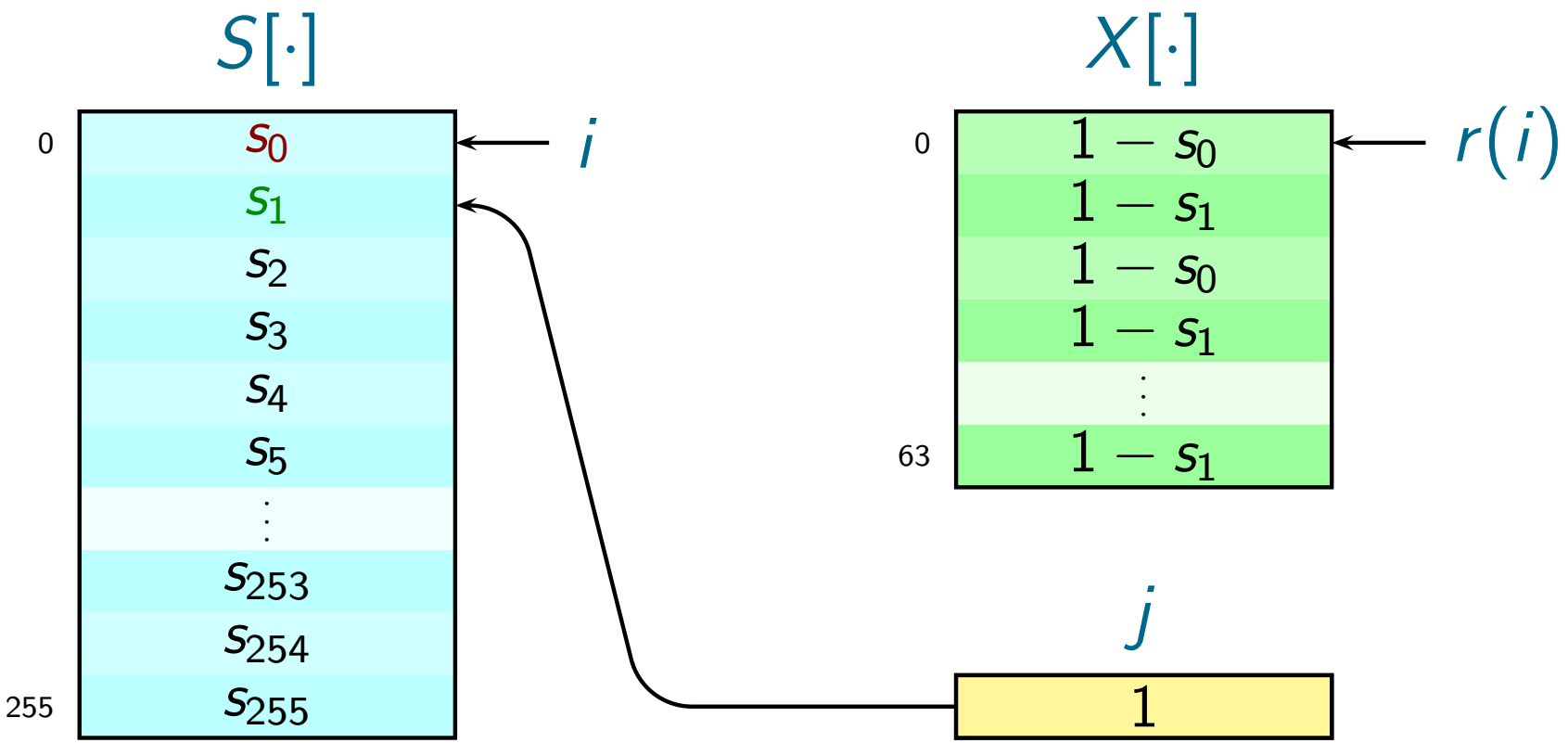
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Fixed Points of Type I

- Repeat **255 times** partial state rotation of type I
- **Only condition**: $j = 0$
- **Cost?** One subtraction mod256

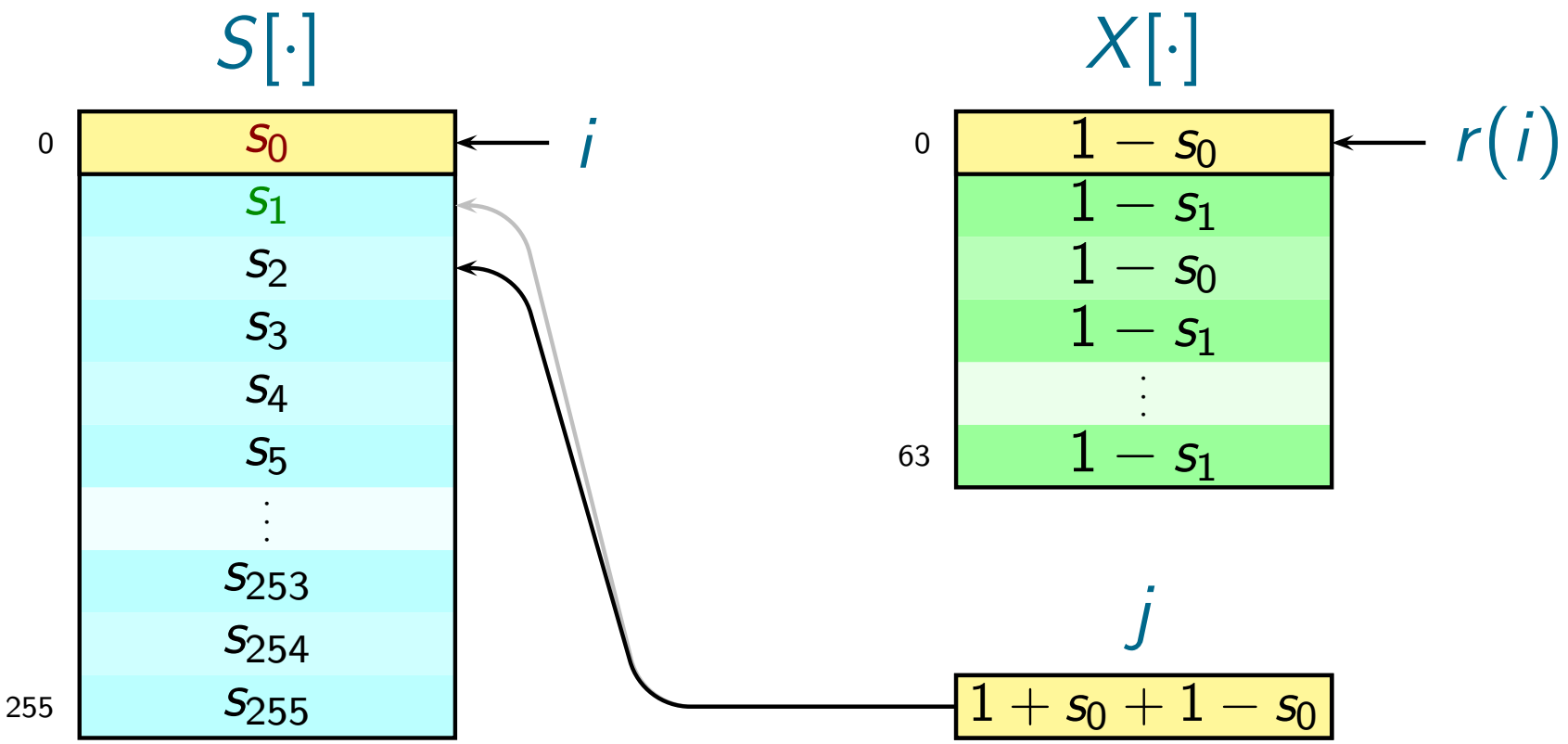


Partial State Rotations of Type II



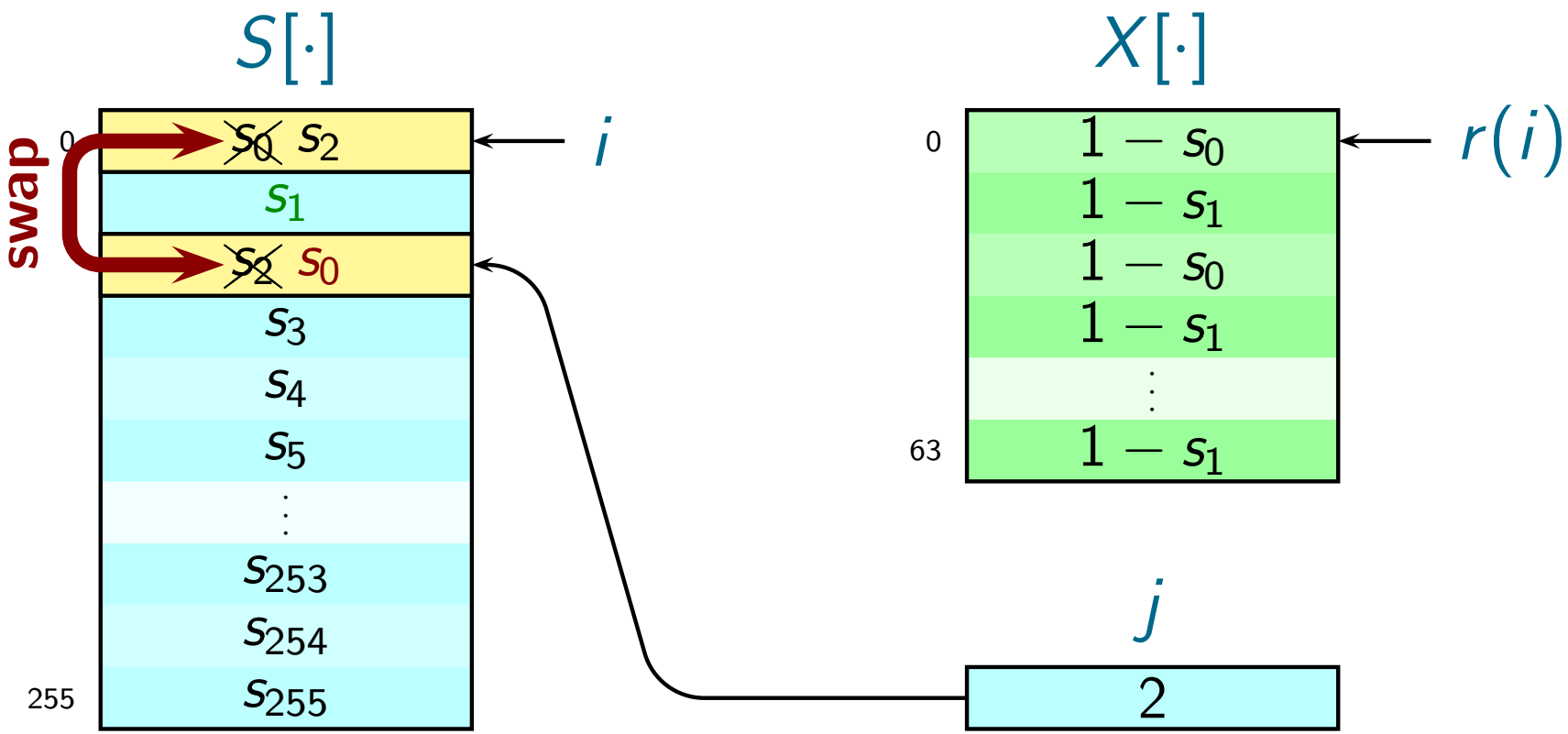
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type II



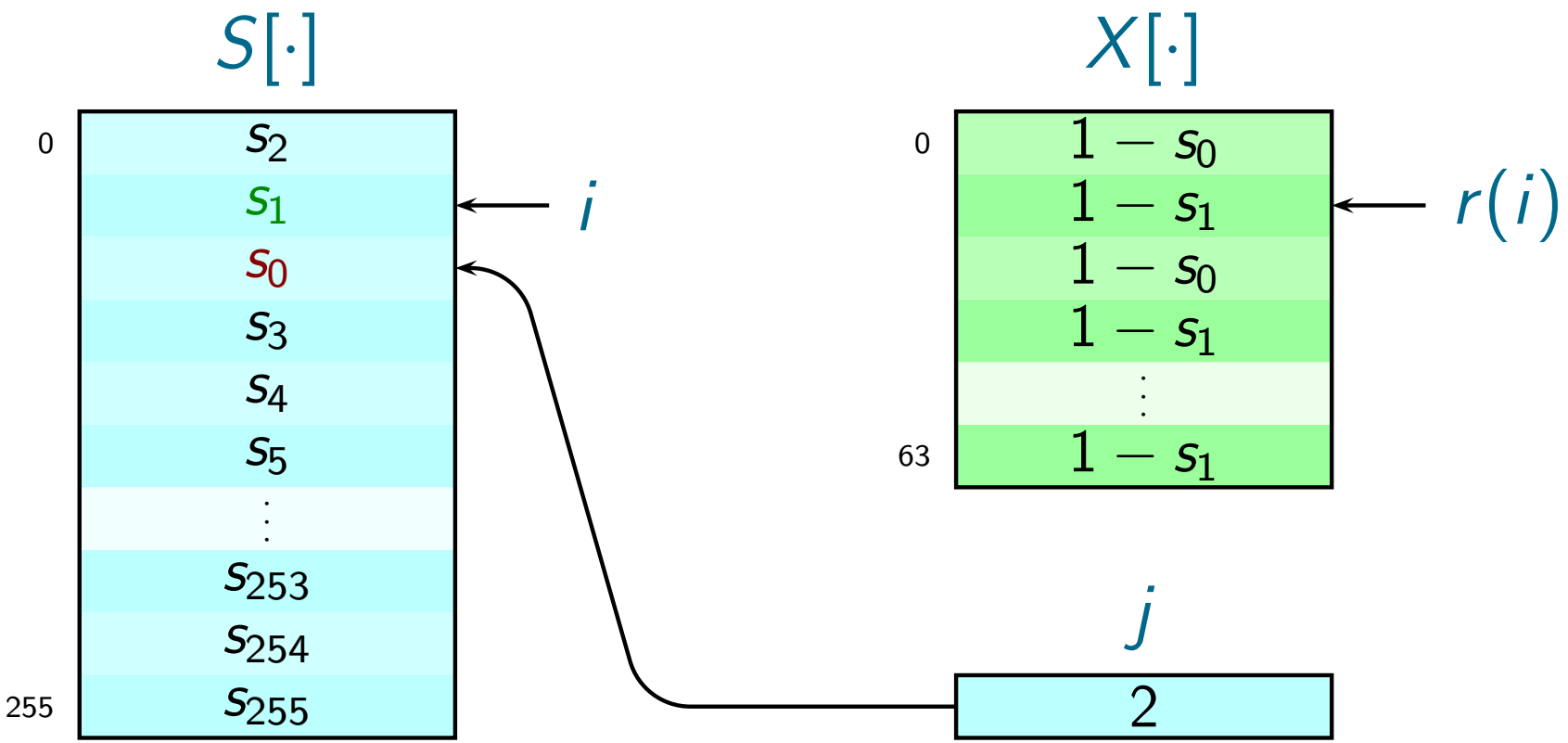
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type II



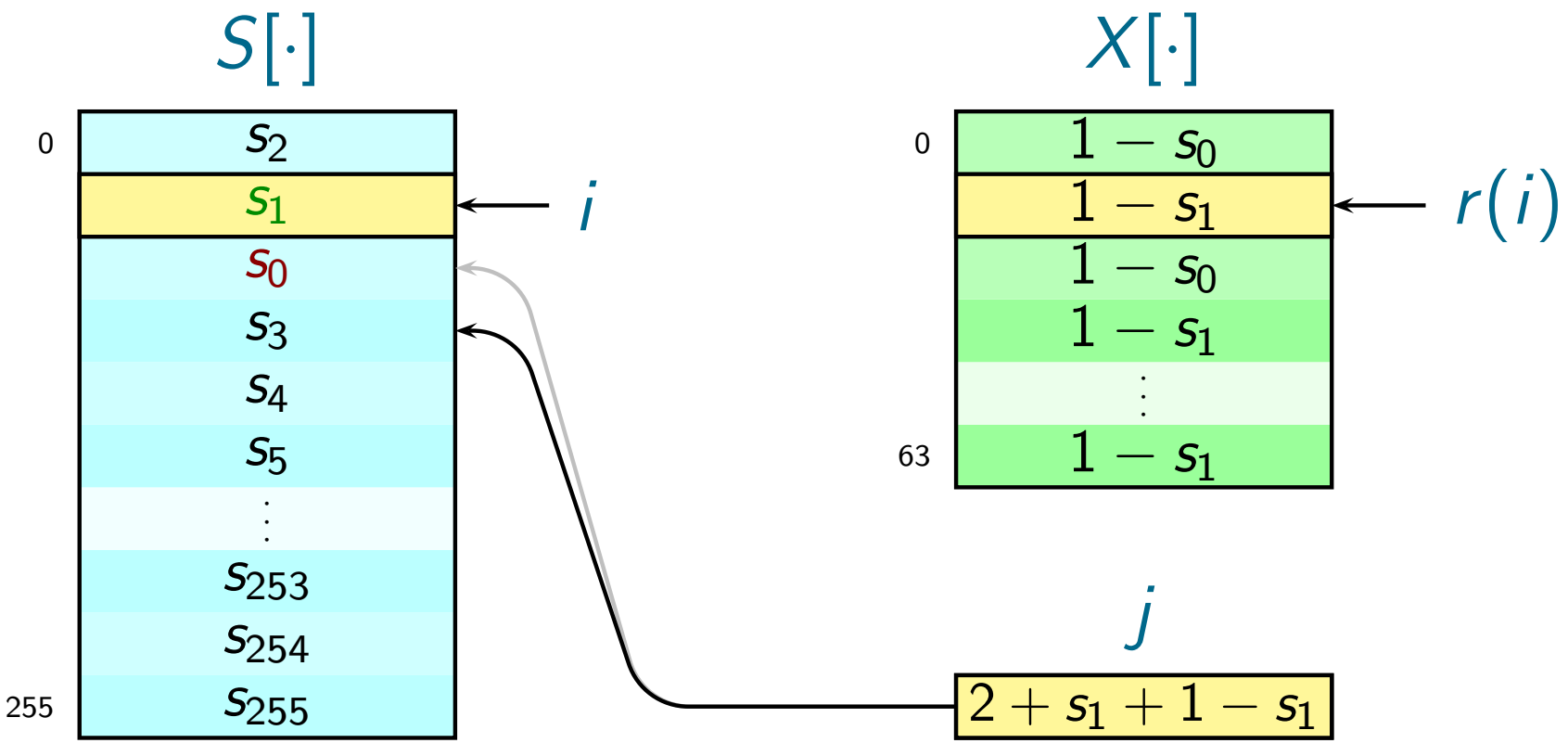
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type II



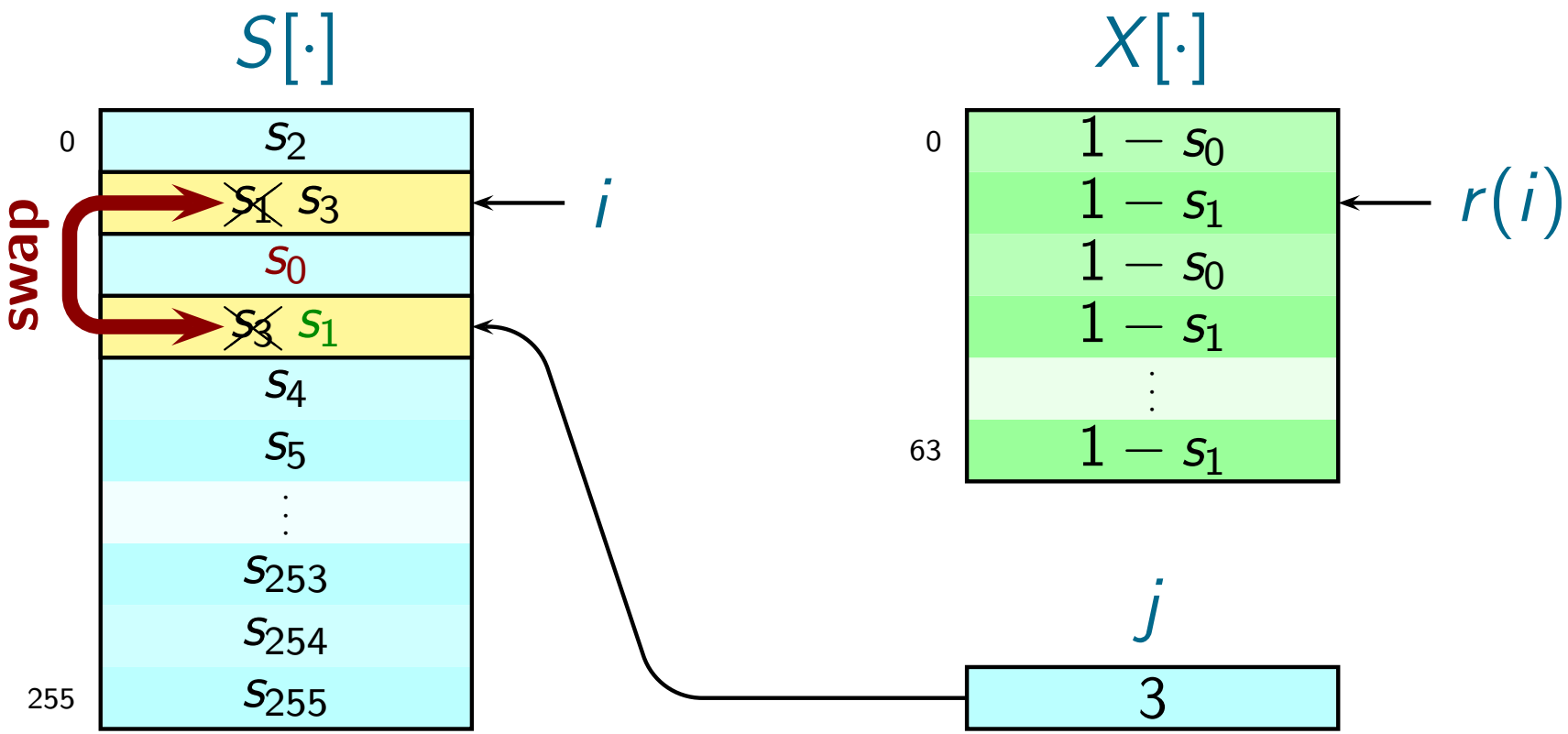
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type II



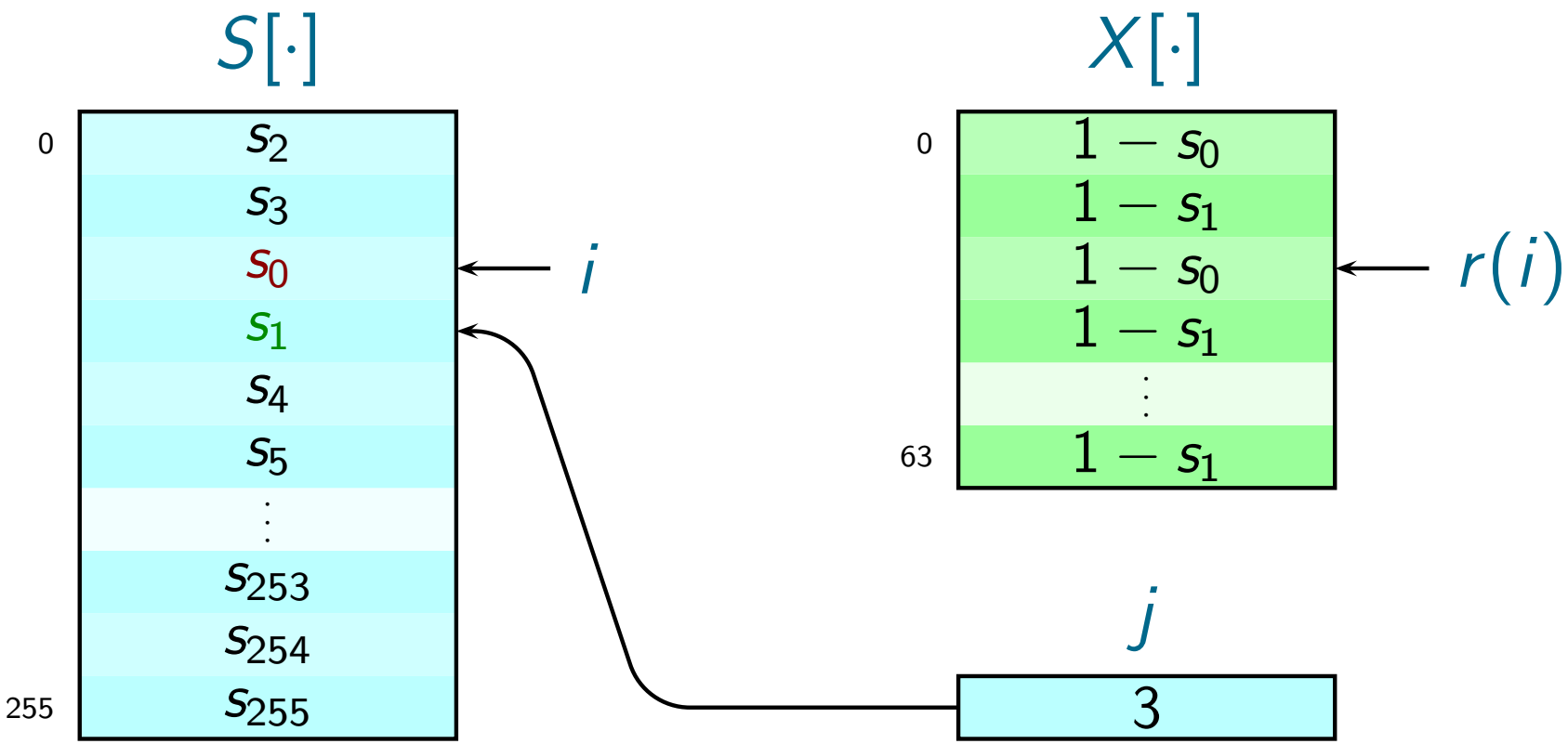
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type II



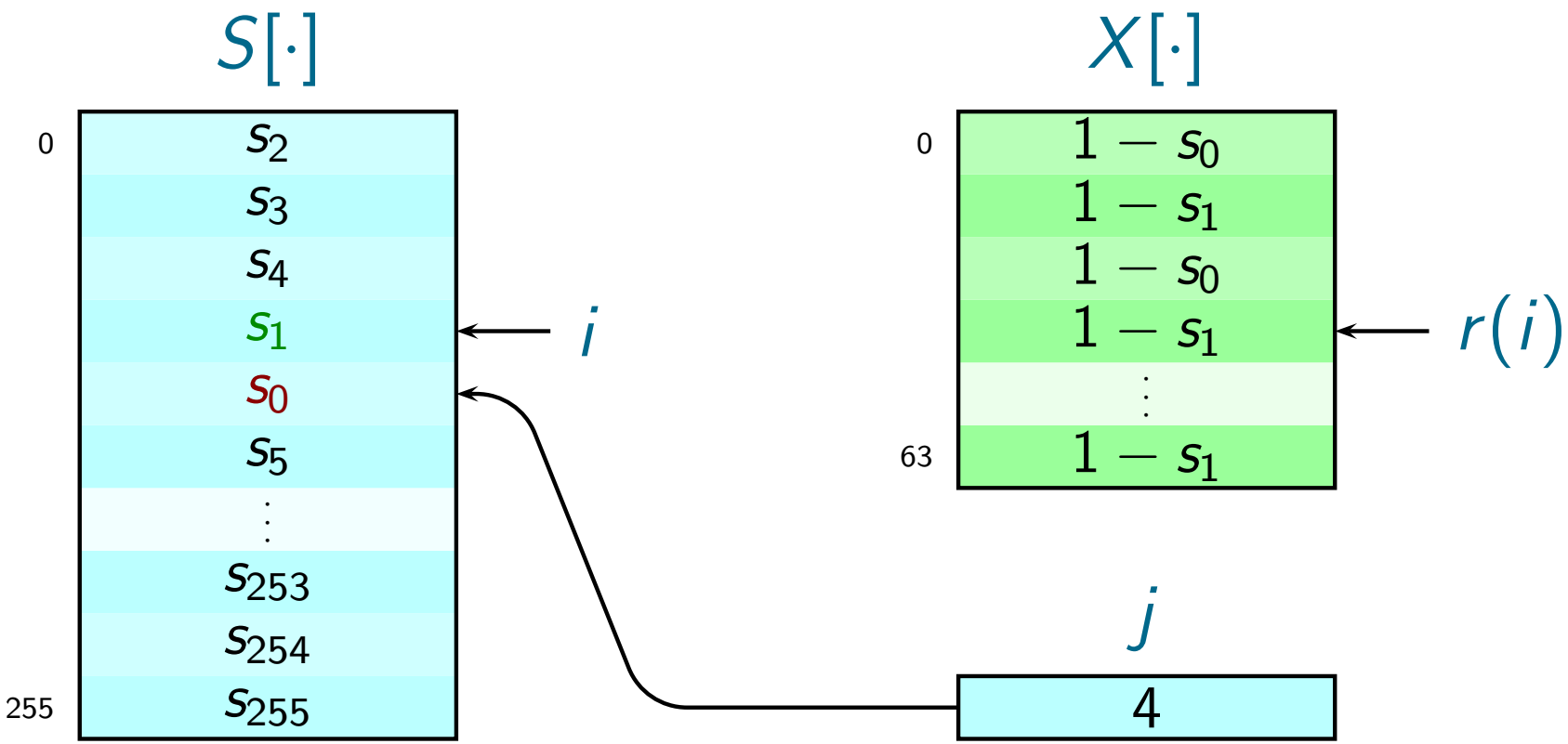
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type II



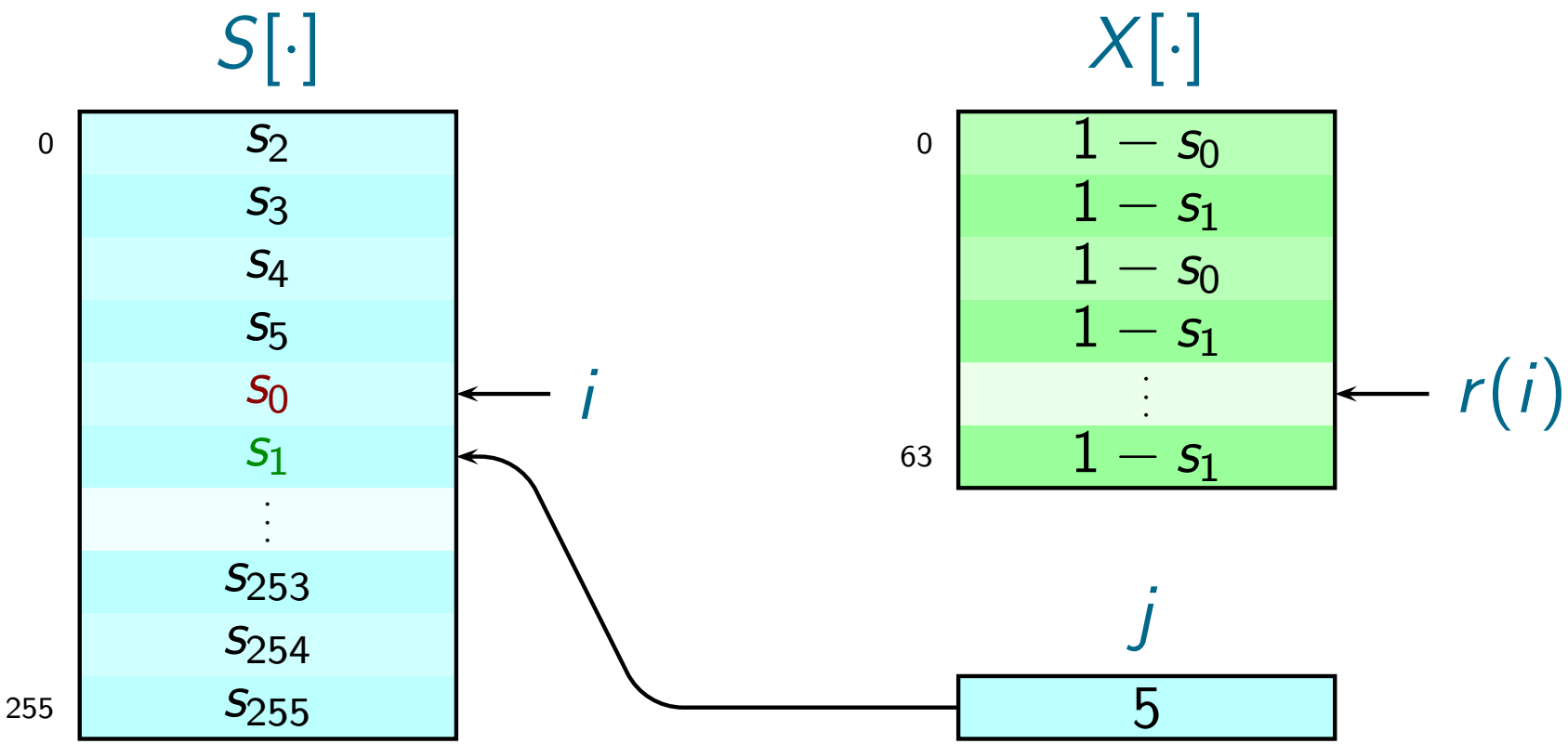
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type II



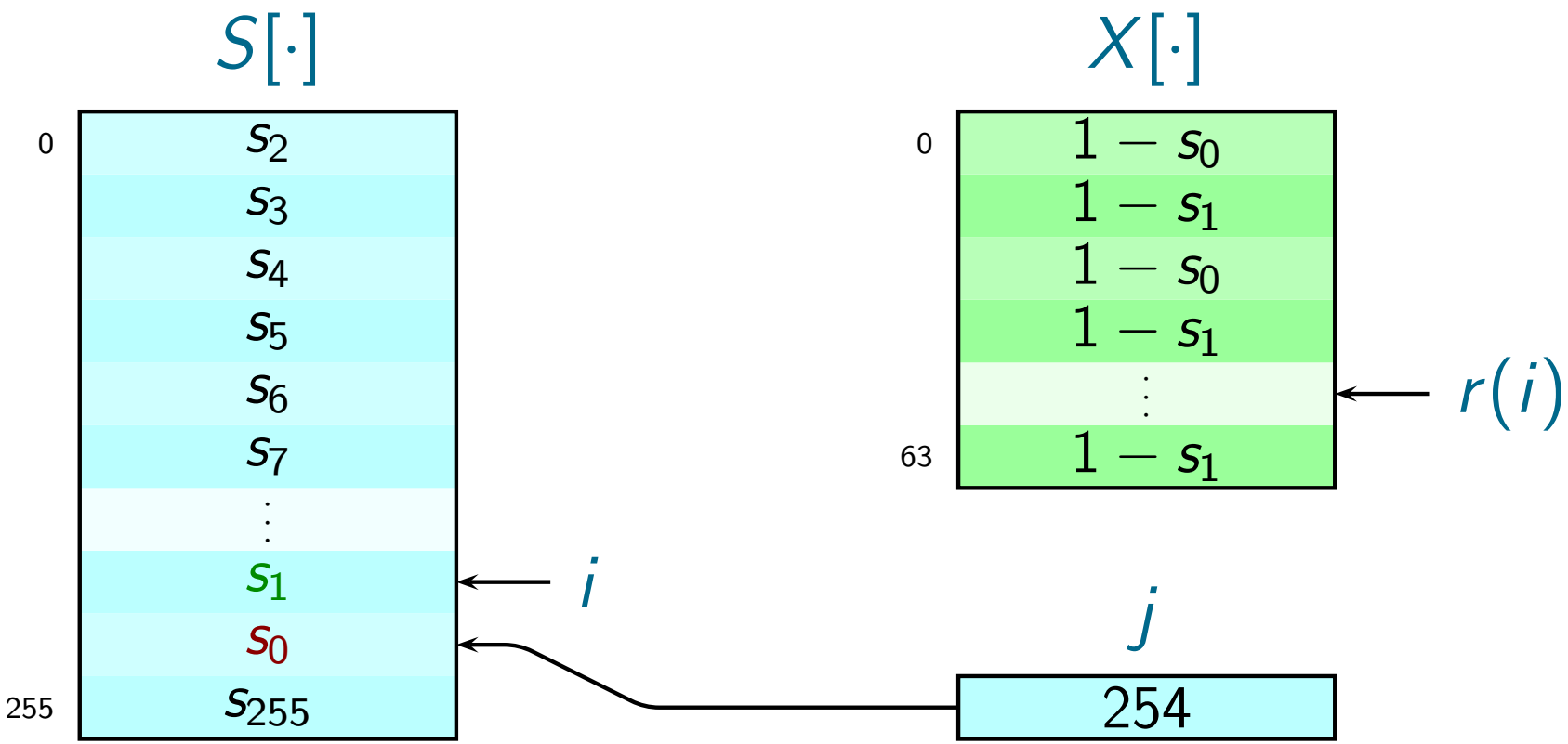
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type II



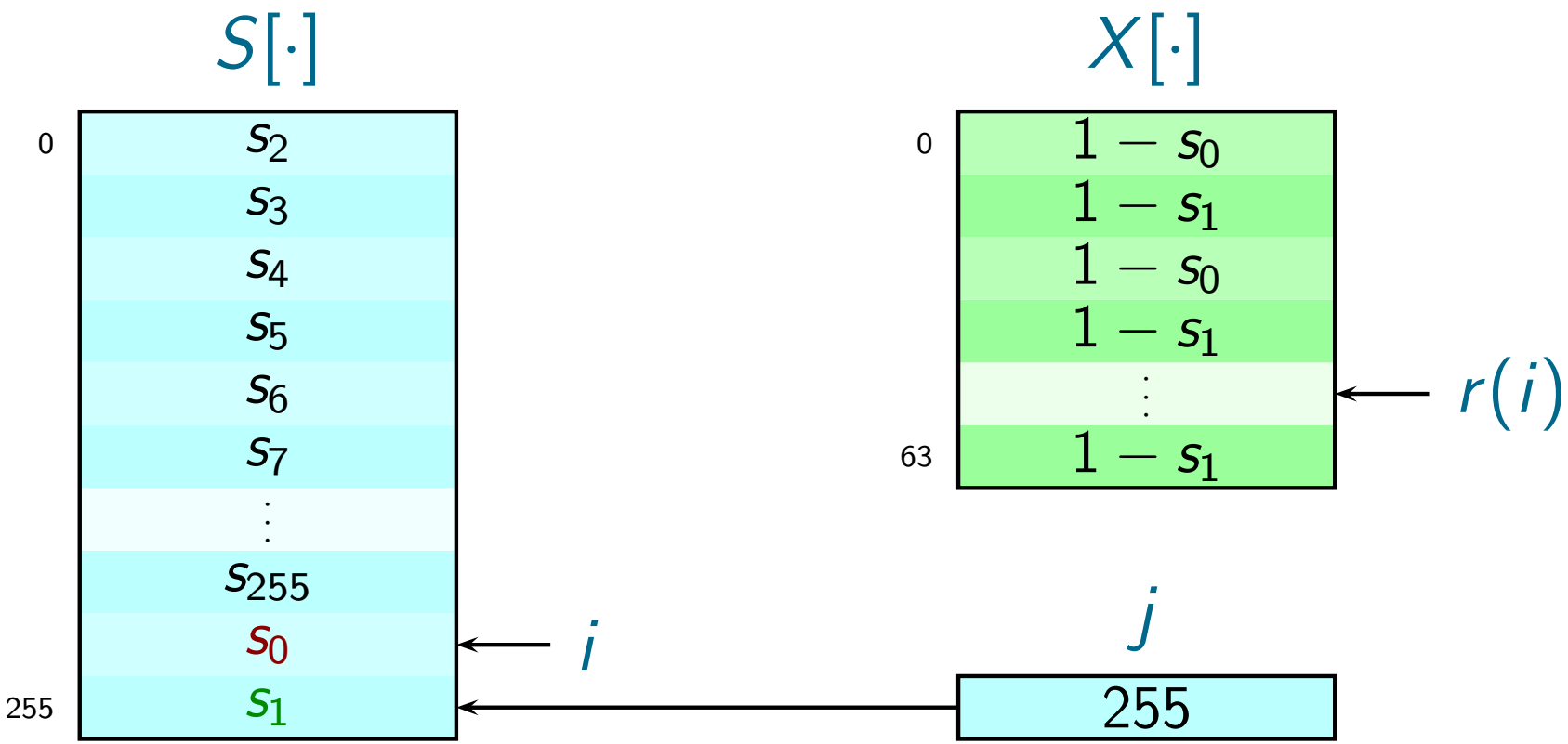
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type II



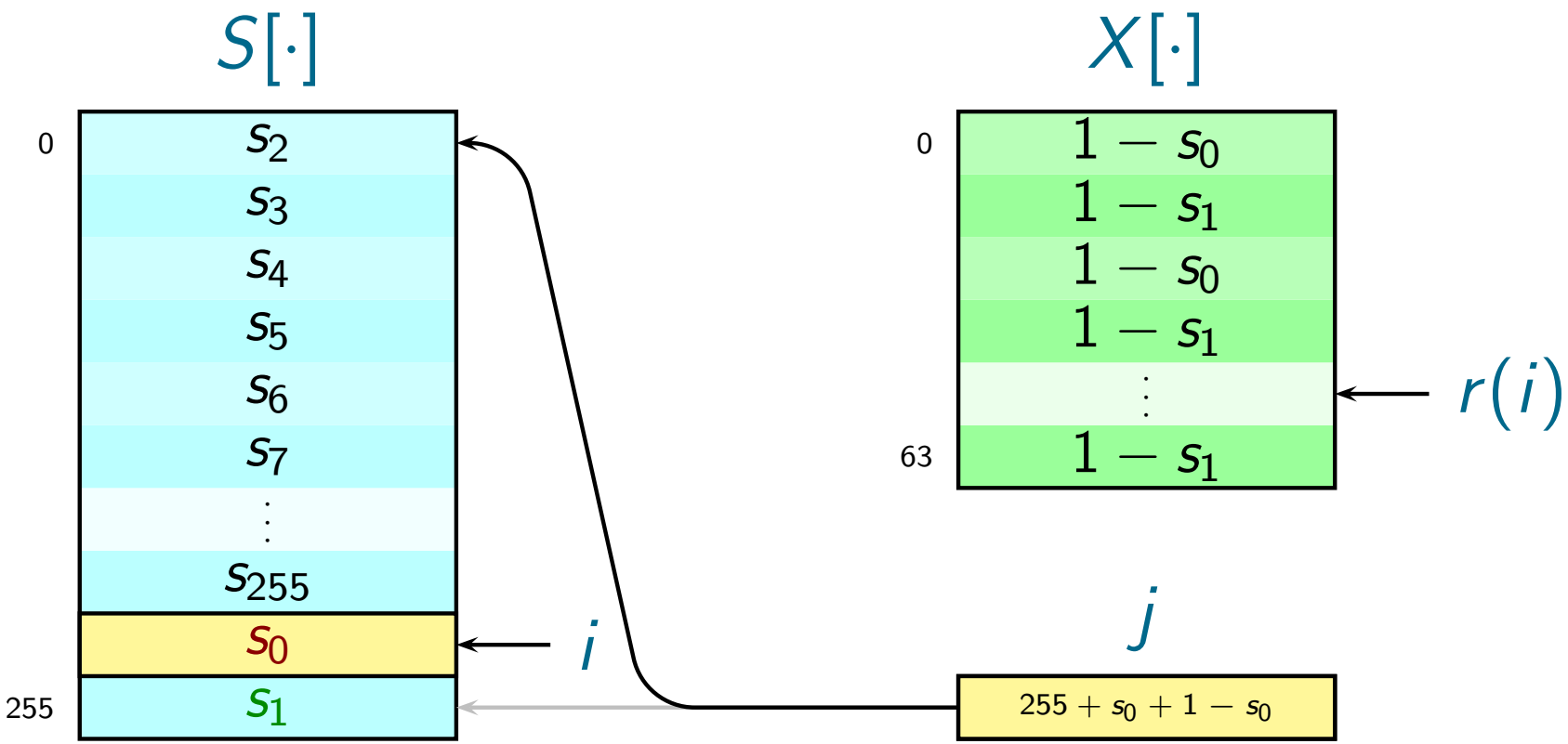
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type II



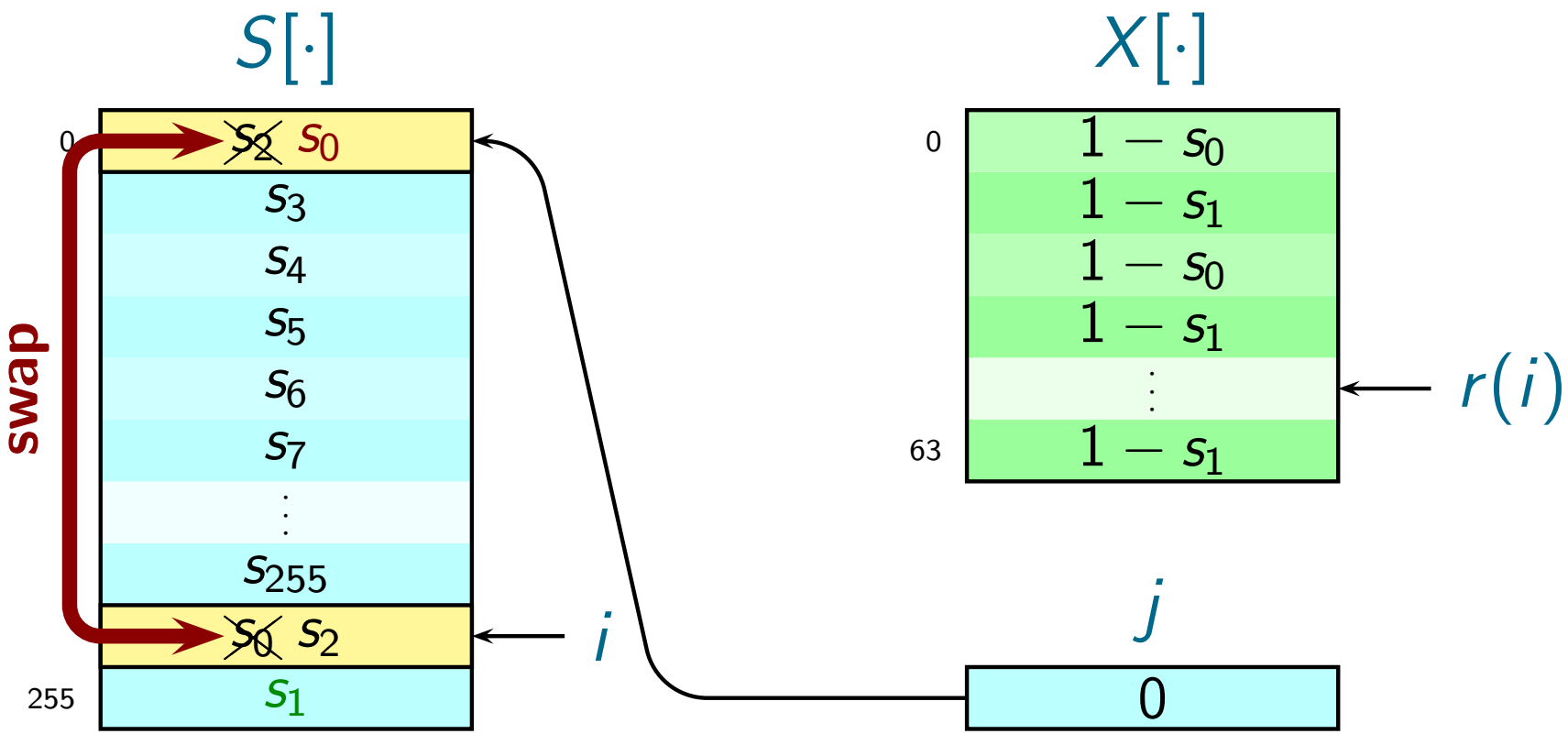
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type II



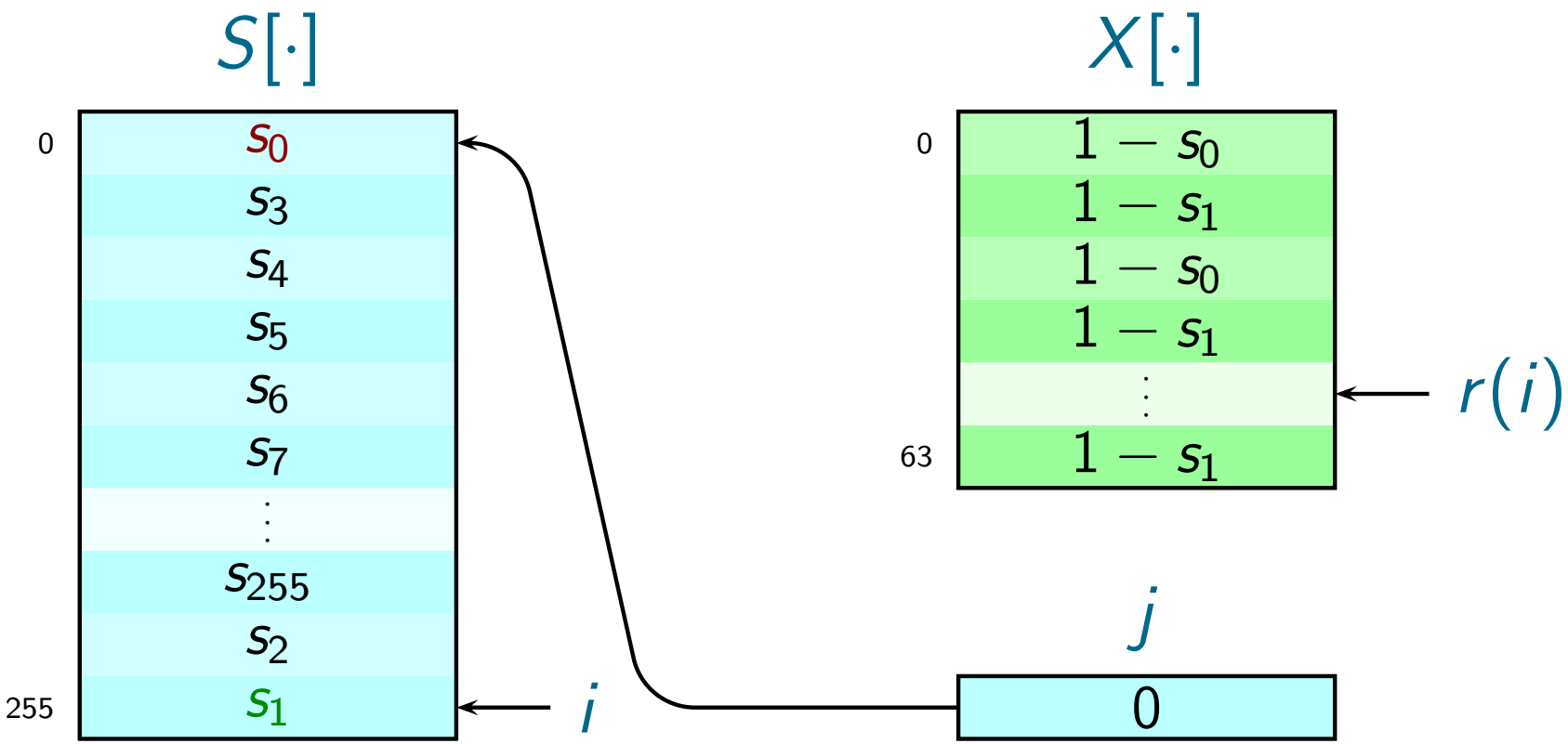
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type II



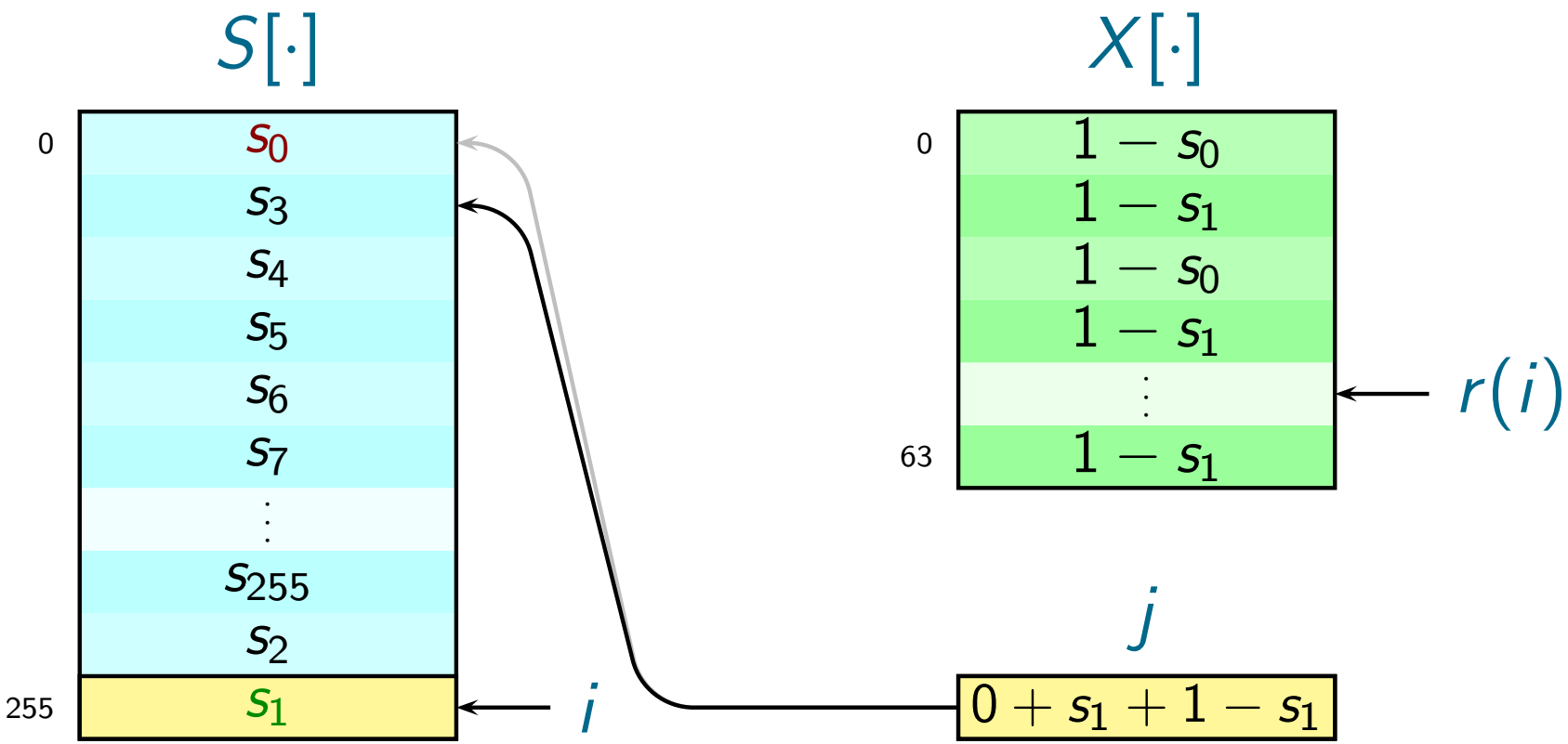
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type II



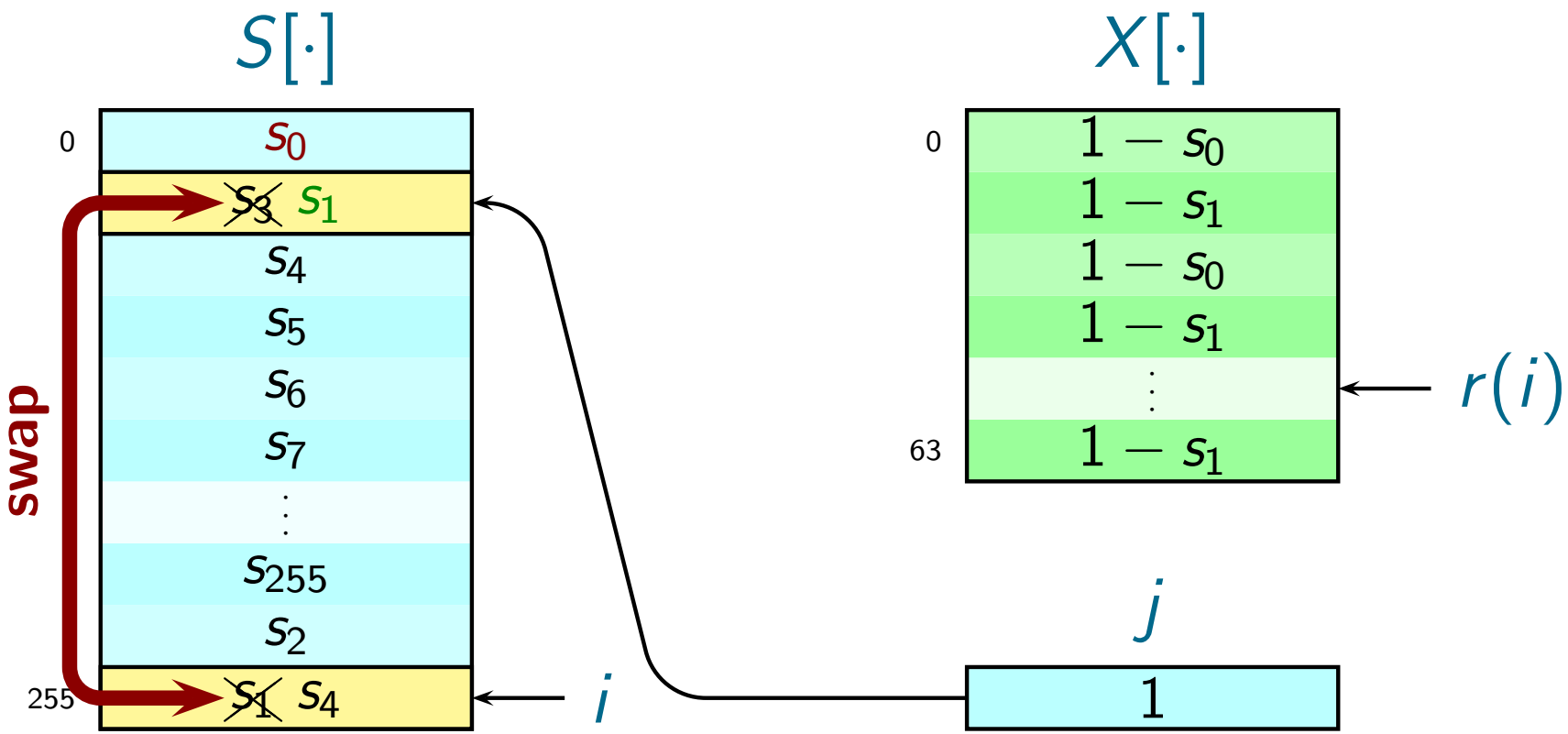
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type II



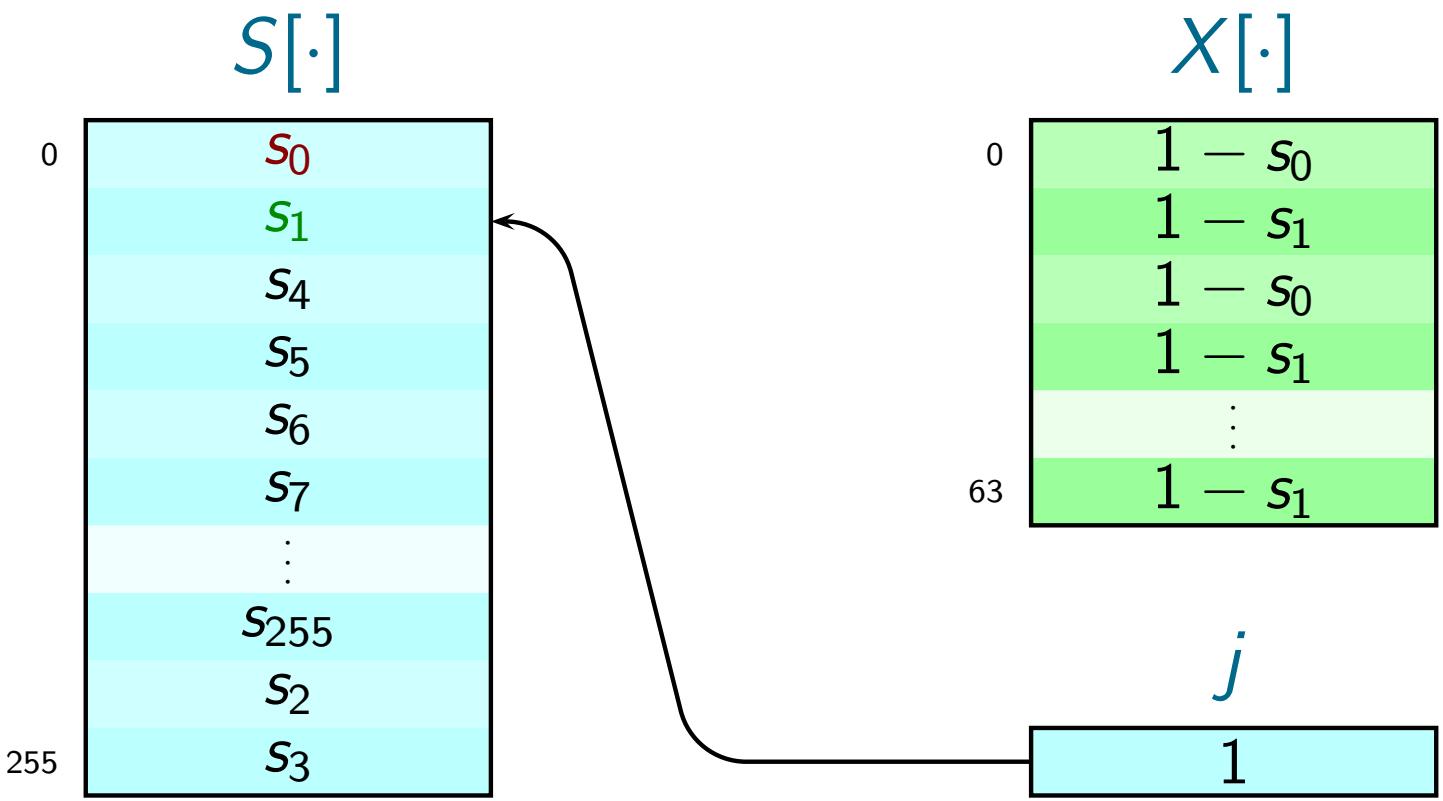
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type II



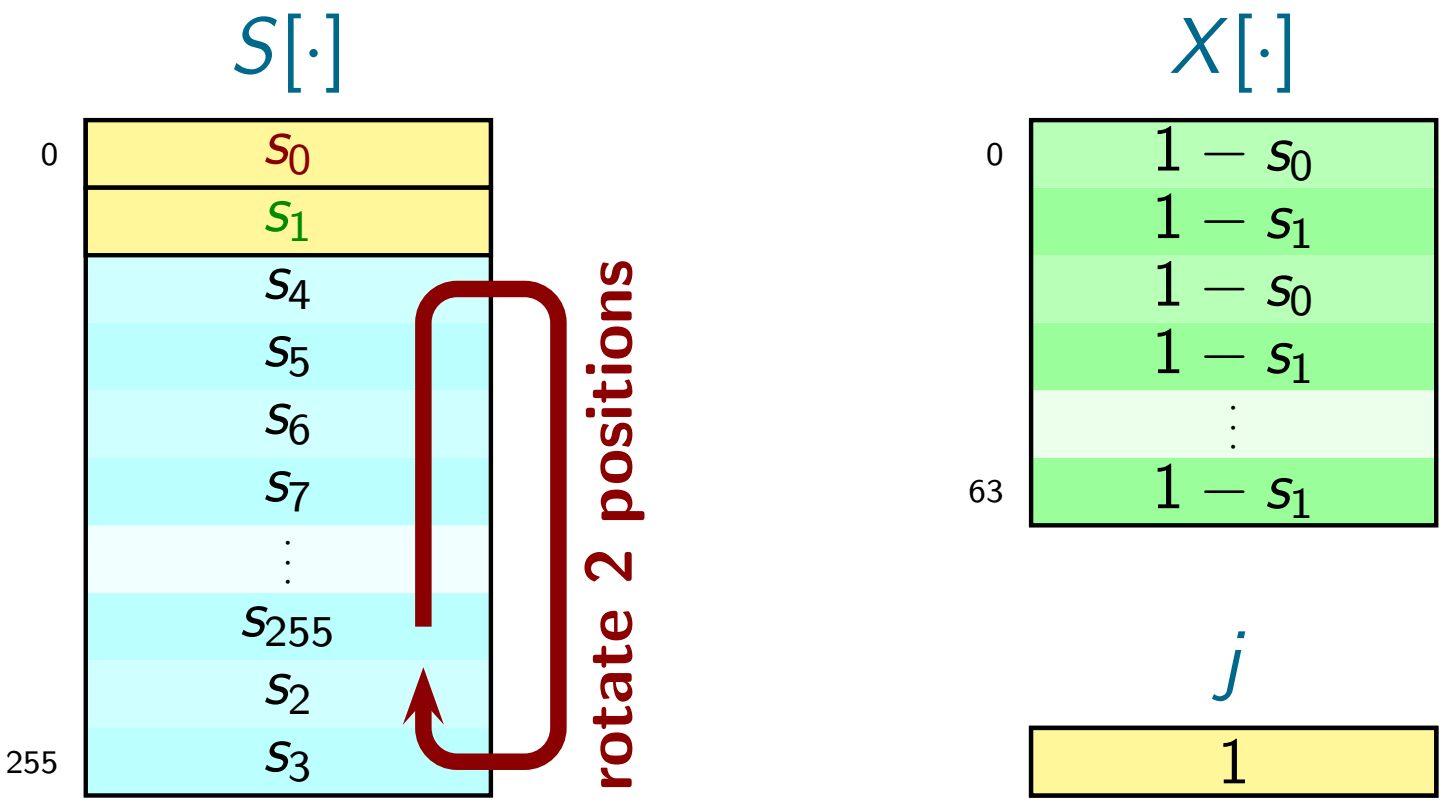
1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

Partial State Rotations of Type II



1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

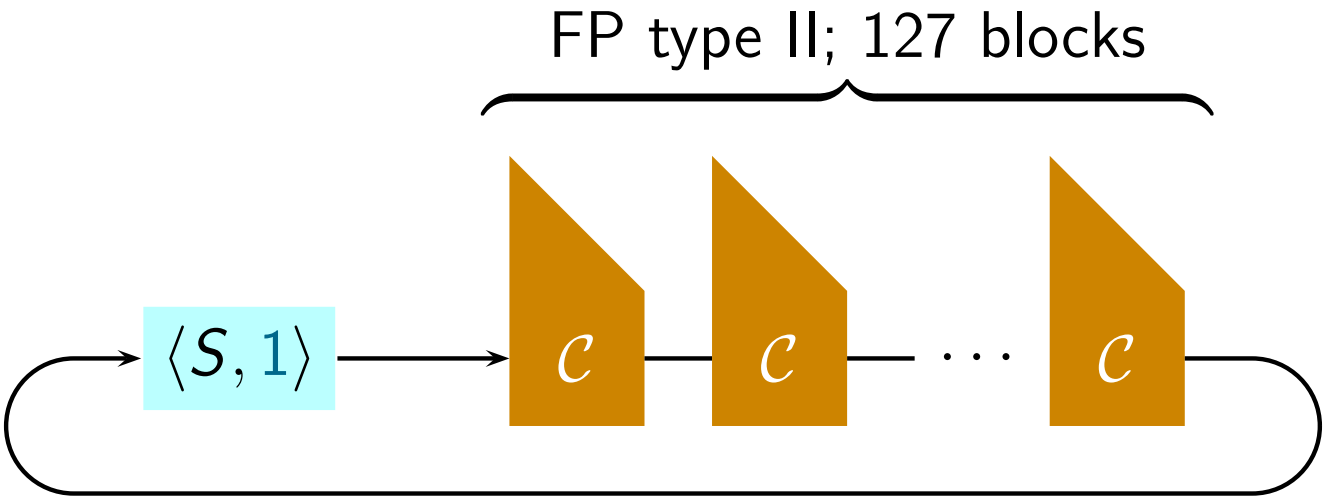
Partial State Rotations of Type II



1: **for** $i = 0$ to 255 **do**
2: $\quad j \leftarrow j + S[i] + X[r(i)] \bmod 256$
3: $\quad$ swap $S[i], S[j]$
4: **end for**

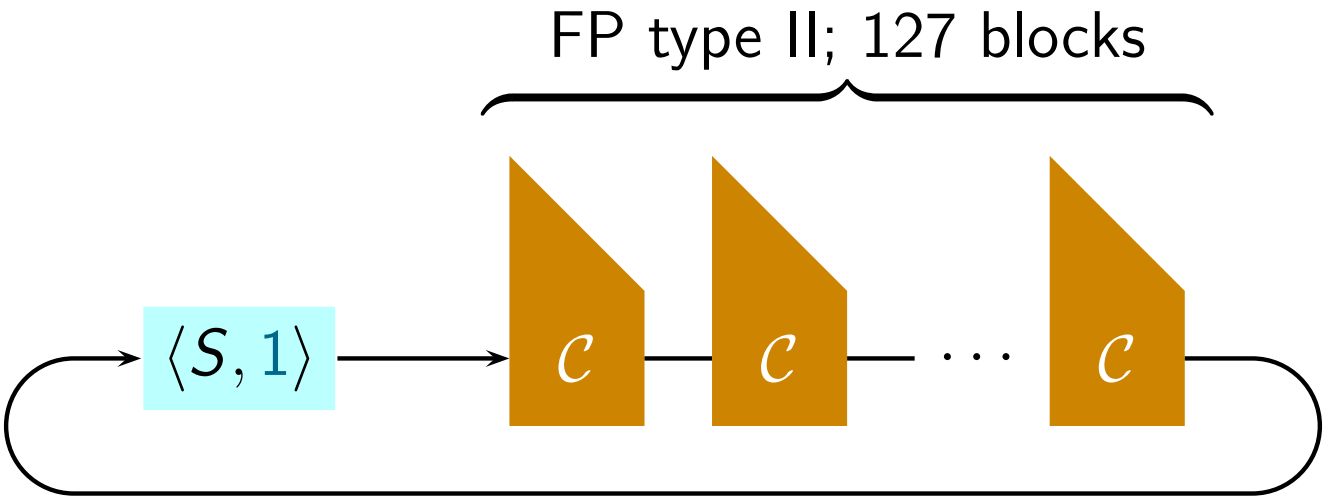
Fixed Points of Type II

- Repeat **127 times** partial state rotation of type II
- **Only condition**: $j = 1$
- **Cost?** Two subtractions mod256



Fixed Points of Type II

- Repeat **127 times** partial state rotation of type II
- **Only condition**: $j = 1$
- **Cost?** Two subtractions mod256



Type III and beyond?

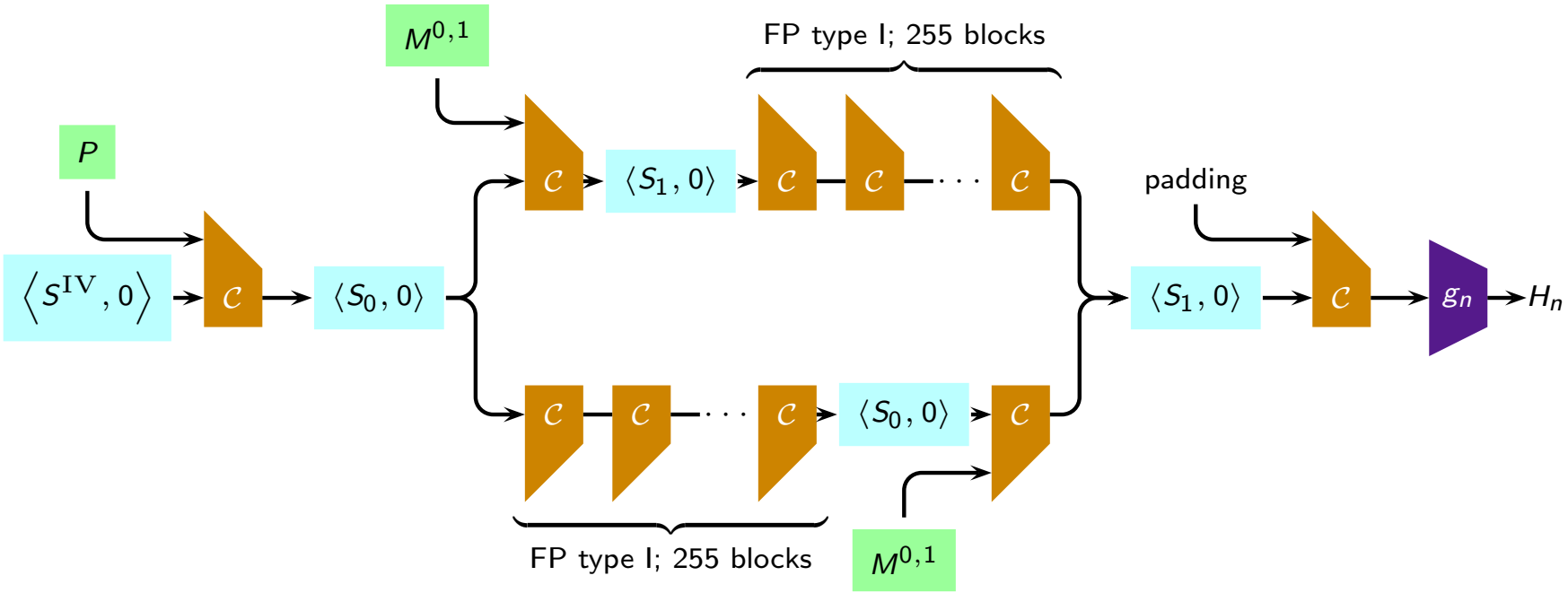
- **Not possible** due to message reordering $r(\cdot)$

Outline

1. Introduction
   - Cryptographic Hash Functions
   - RC4-Hash
2. Fixed Points for RC4-Hash
   - Type I
   - Type II
3. **Collisions for RC4-Hash**
4. Conclusion

Collisions for RC4-Hash

Using Fixed Points of Type I

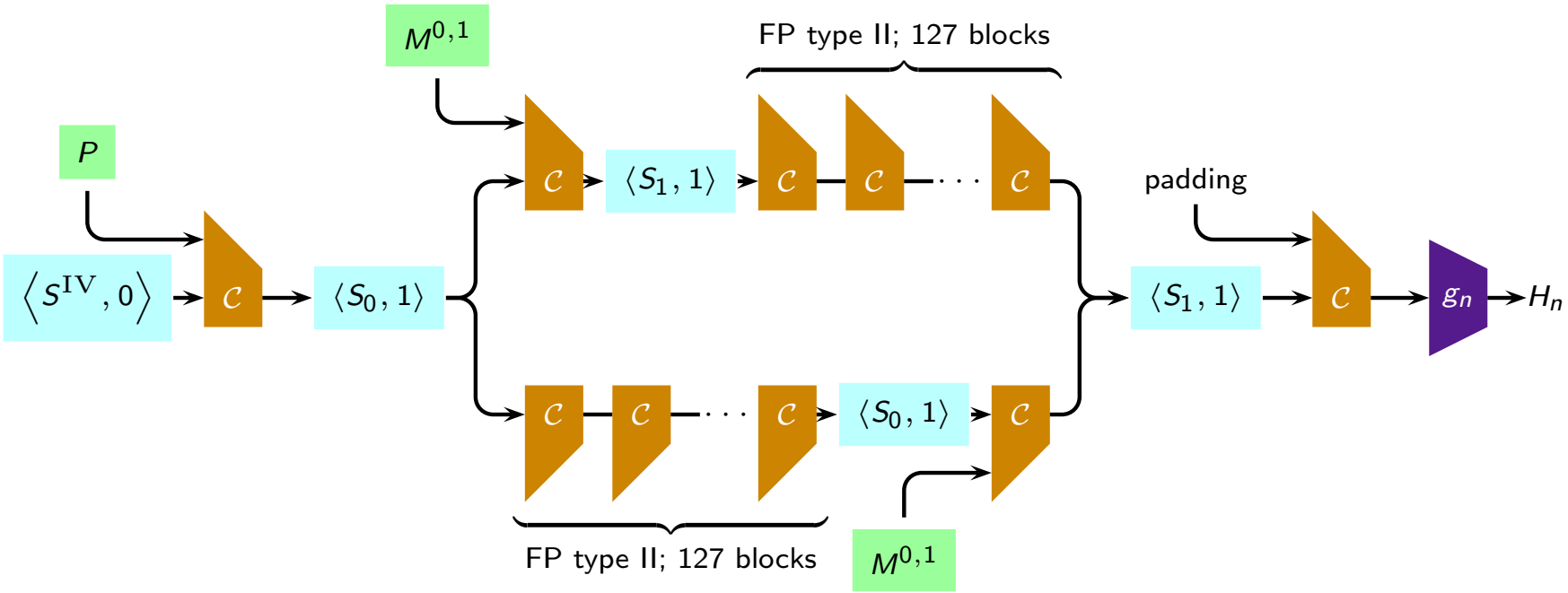


Time Complexity $\approx 2^9$ evaluations of $\mathcal{C}$

- Prefix block P $\approx 2^8$
- Message block $M^{0,1}$ $\approx 2^8$

Collisions for RC4-Hash

Using Fixed Points of Type II

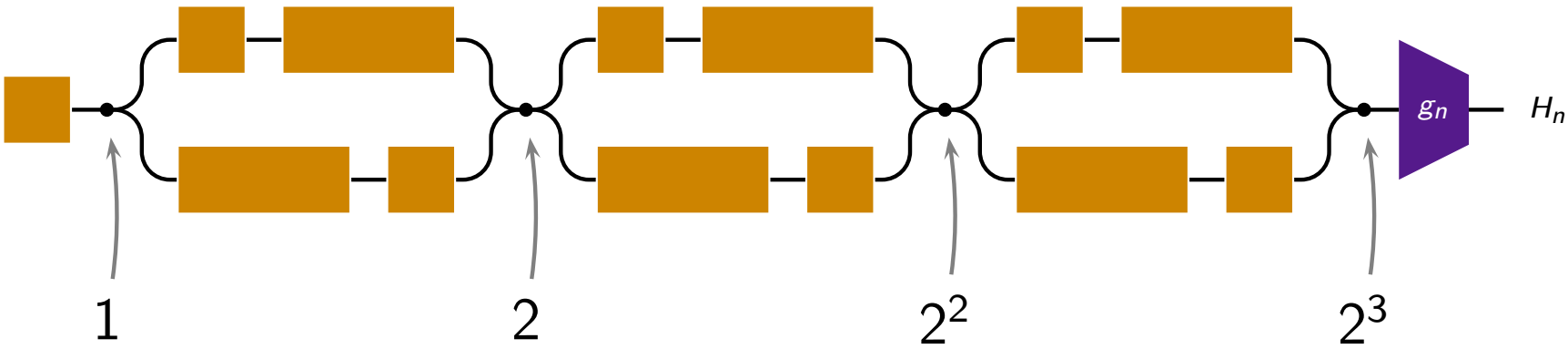


Time Complexity $\approx 2^9$ evaluations of $\mathcal{C}$

- Prefix block P $\approx 2^8$
- Message block $M^{0,1}$ $\approx 2^8$

Collisions for RC4-Hash

Multicollisions



Multicollisions

- 2^k colliding messages
- Technique of A. Joux
- Time complexity $\approx 2^7 + k \cdot 2^8$

Outline

1. Introduction
   - Cryptographic Hash Functions
   - RC4-Hash
2. Fixed Points for RC4-Hash
   - Type I
   - Type II
3. Collisions for RC4-Hash
4. **Conclusion**

Conclusion

RC4-Hash is not collision resistant

- Easy construction of **fixed points**
- Practical **collision attack**
  - Time complexity $\approx 2^9$
- **Multicollisions**
  - 2^k colliding messages for $\approx 2^7 + k \cdot 2^8$

Conclusion

RC4-Hash is not collision resistant

- Easy construction of **fixed points**
- Practical **collision attack**
  - Time complexity $\approx 2^9$
- **Multicollisions**
  - 2^k colliding messages for $\approx 2^7 + k \cdot 2^8$