UMassAmherst

Cryptanalysis of Two Lightweight RFID Authentication Schemes

Benessa Defend and Kevin Fu, UMass Amherst
Ari Juels, RSA Laboratories

defend@cs.umass.edu
http://www.cs.umass.edu/~defend
http://www.rfid-cusp.org

RSA
The Security Division of EMC

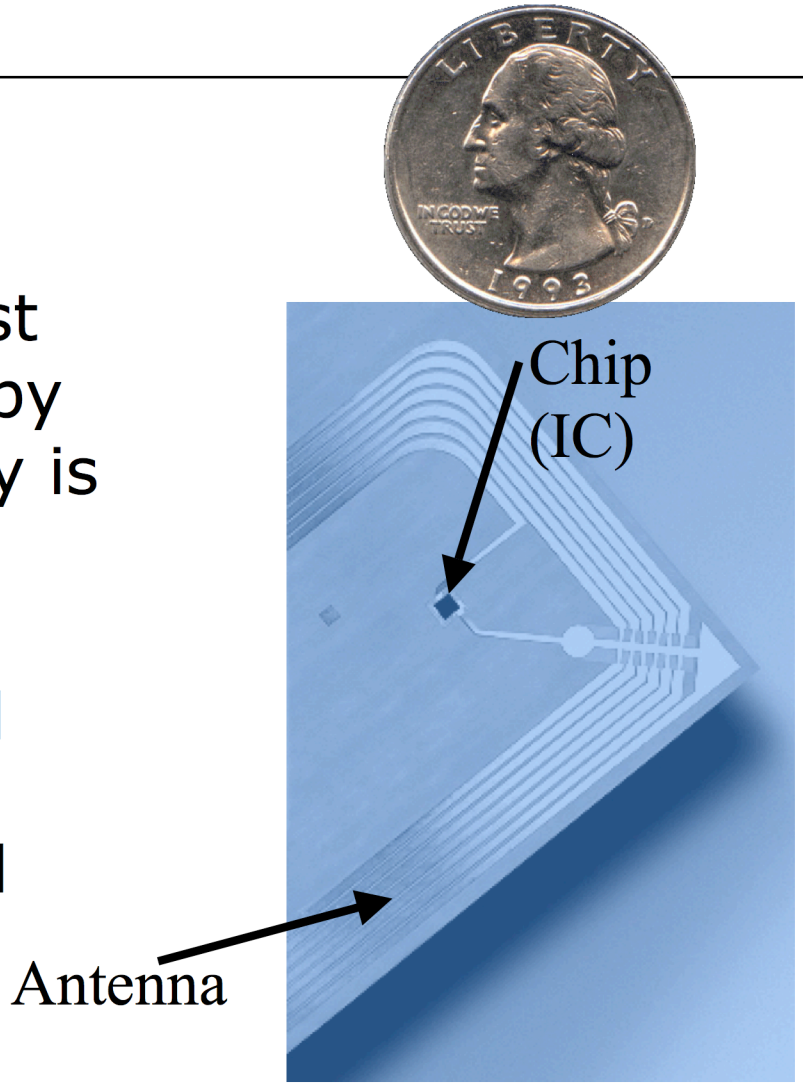
RFID
Consortium for Security and Privacy

Outline

- Introduction to RFID
- Original low-cost RFID authentication protocol
- Implementation Results
- Repeated Keys Attack
- Nibble Attack
- Suggestions for future protocols

Introduction to RFID

- Radio Frequency IDentification
- A radio signal is broadcast by a reader, interpreted by a transponder and a reply is returned to the reader.
- Tags are either active (possess a self-contained power source) or passive (powered solely by signal received from reader)



Applications for RFID Tags

- Person Identification
  - Hospitals
  - Homes
  - Businesses
- Livestock Management
- Inventory Control
- Building Access Control

Why Authentication for RFID Tags?

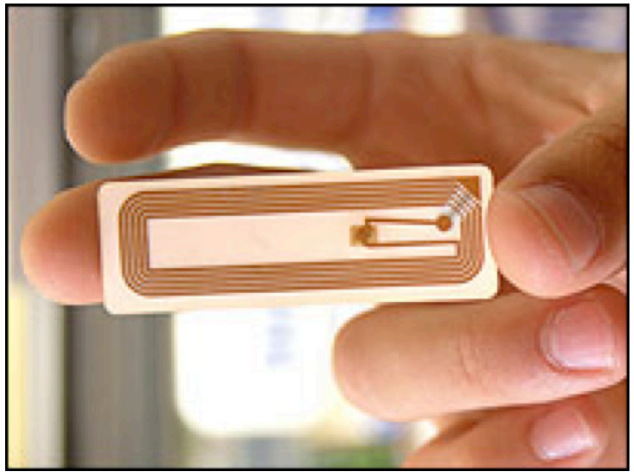
- Prevention of unauthorized communication
- Authorization of a tag by a reader
  - Detection of cloned/spoofed tags
  - Person identification
  - Theft prevention
  - Inventory control

Why Authentication for RFID Tags?

- Prevention of unauthorized communication
- Authorization of a tag by a reader
  - Detection of cloned/spoofed tags
  - Person identification
  - Theft prevention
  - Inventory control

Low Cost vs. Higher Cost

	Low Cost	**Higher Cost**
Storage	Few 100 bits	Few kB
Computational Capabilities	XOR, simple operations	RSA, AES, Triple DES
Cost	Few cents	Few dollars



Outline

- Introduction to RFID
- Original low-cost RFID authentication protocol
- Implementation Results
- Repeated Keys Attack
- Nibble Attack
- Suggestions for future protocols

Vajda and Buttyán Protocol 1

- Challenge/Response Protocol
- Uses XOR operations -> lightweight
- Reader and Tag share secret
- Session key computed by "evolving" previous key

"Lightweight Authentication Protocols for Low-Cost RFID Tags" by I. Vajda and L. Buttyan. In UBICOMP, 2003."

Vajda and Buttyán Protocol 1

Vajda and Buttyán Protocol 1

Vajda and Buttyán Protocol 1

Vajda and Buttyán Protocol 1

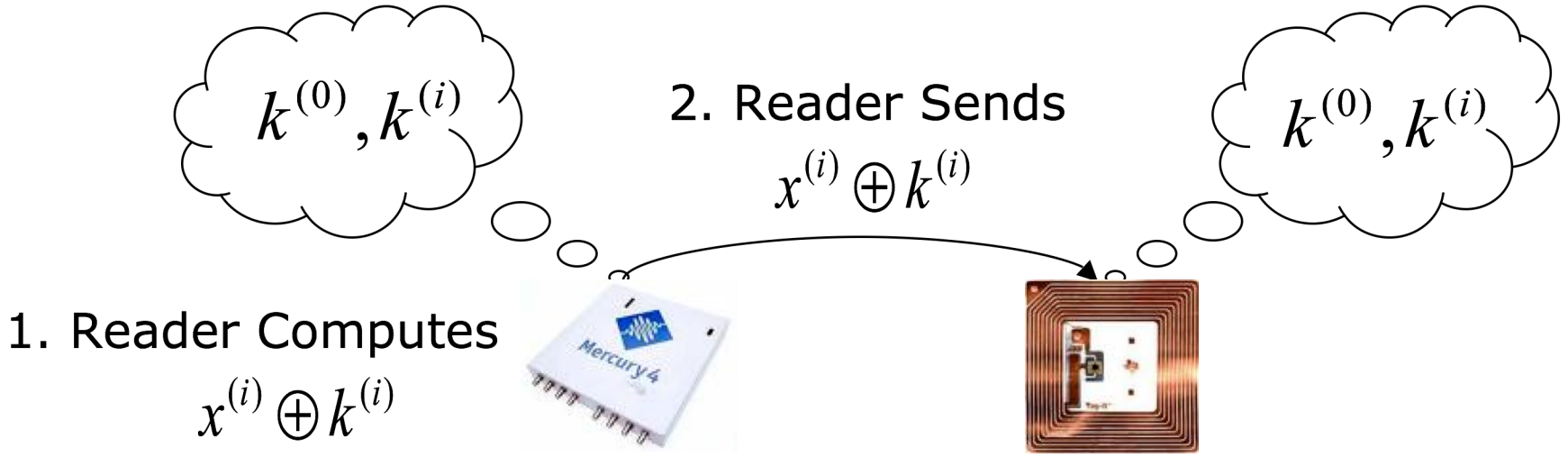
Vajda and Buttyán Protocol 1

Vajda and Buttyán Protocol 1

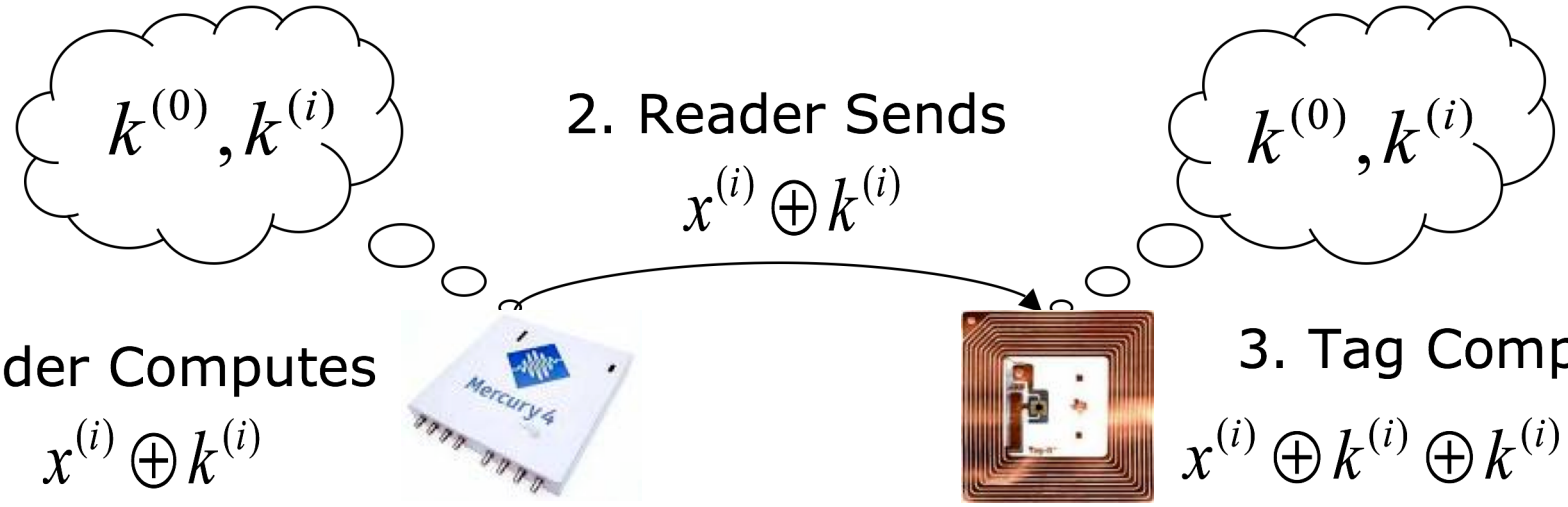
1. Reader Computes

$$x^{(i)} \oplus k^{(i)}$$

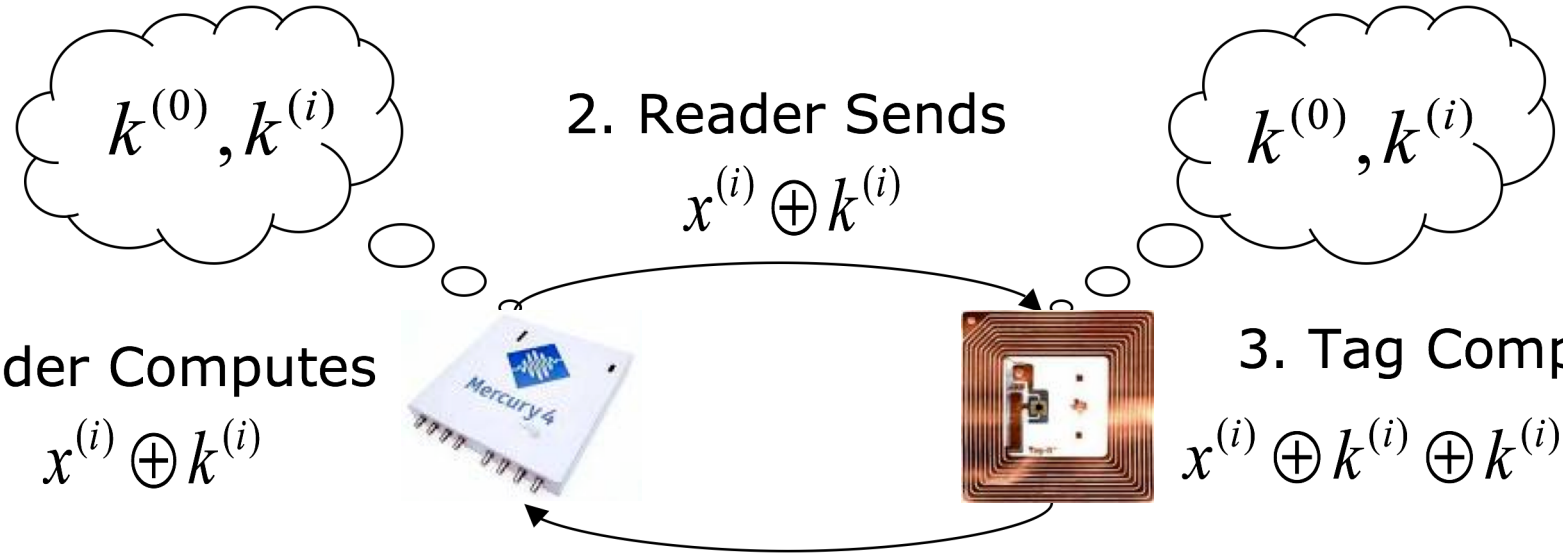
Vajda and Buttyán Protocol 1



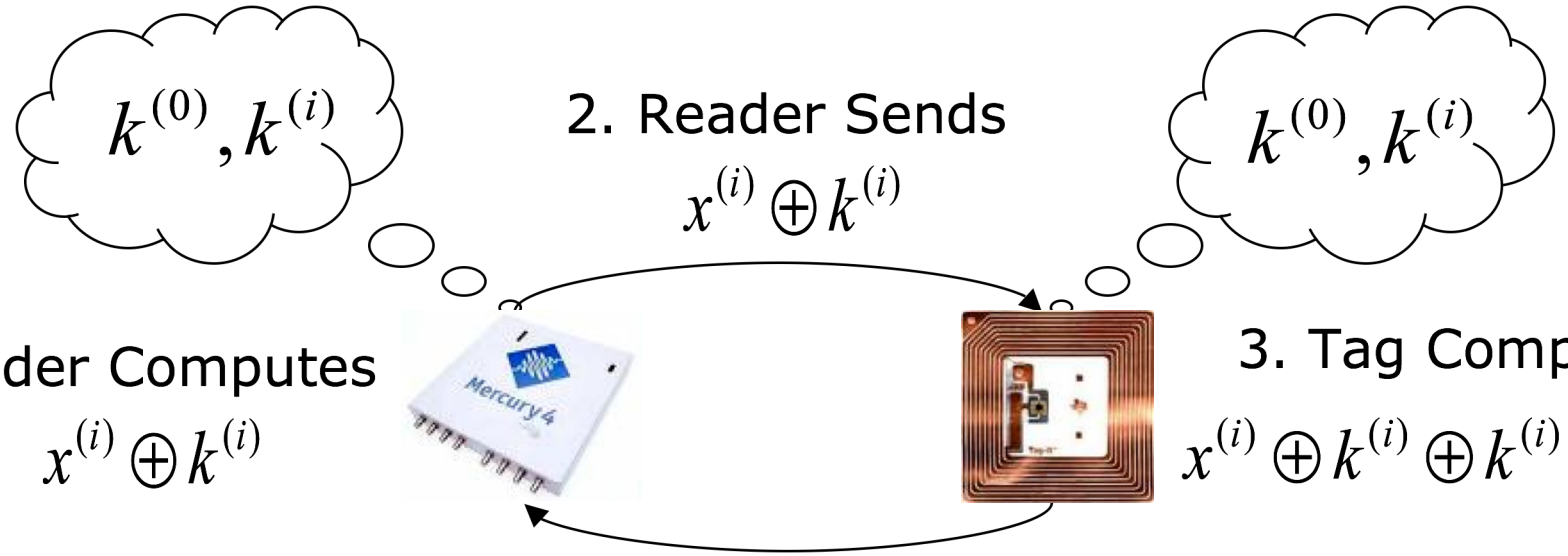
Vajda and Buttyán Protocol 1



Vajda and Buttyán Protocol 1



Vajda and Buttyán Protocol 1

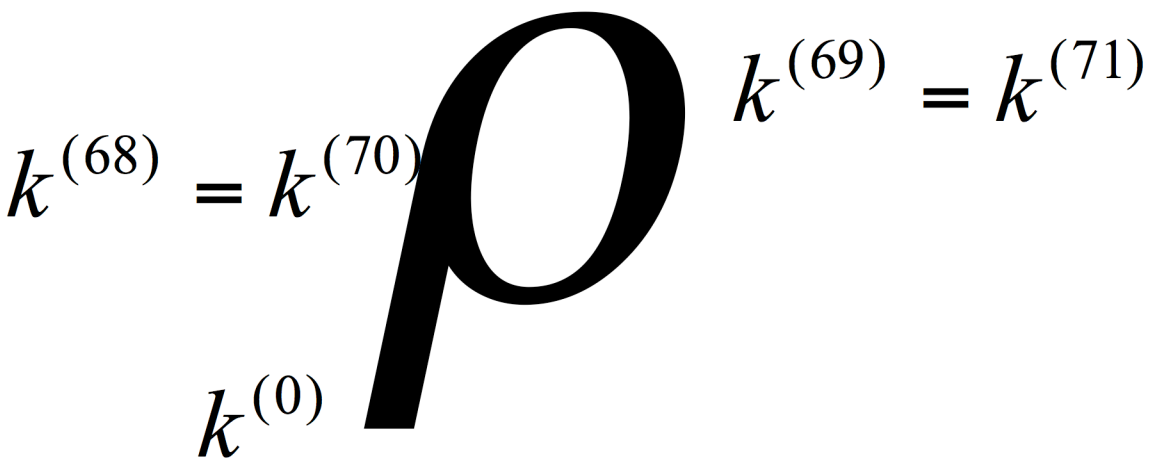


Outline

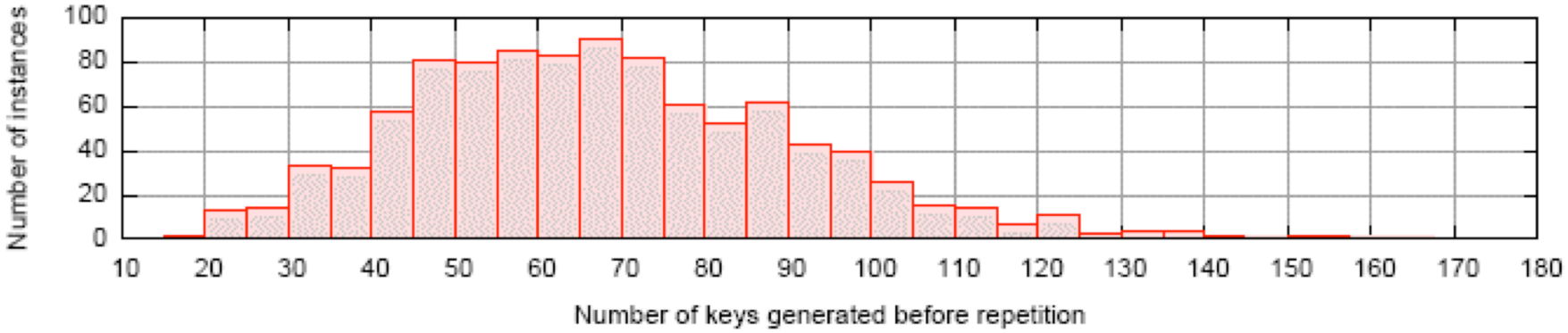
- Introduction to RFID
- Original low-cost RFID authentication protocol
- Implementation Results
- Repeated Keys Attack
- Nibble Attack
- Suggestions for future protocols

Implementation Results

- 128-bit key length
- 1,000 trials with 10,000 sessions/trial
- After an average of 68 keys, the session key repeats
  - Average: 68.7%, cycle period = 2, i.e. $k^{(i)}=k^{(i-2)}$
  - Minimum: 31.9%, cycle period = 1
  - Maximum: 0.1%, cycle period = 36



Key Repetition



- 1,000 trials with random $k^{(0)}$'s execute until a key repeats
- Average is 68 transactions before keys repeat
- When keys repeat, they repeat (cycle) every 2 keys on average

Outline

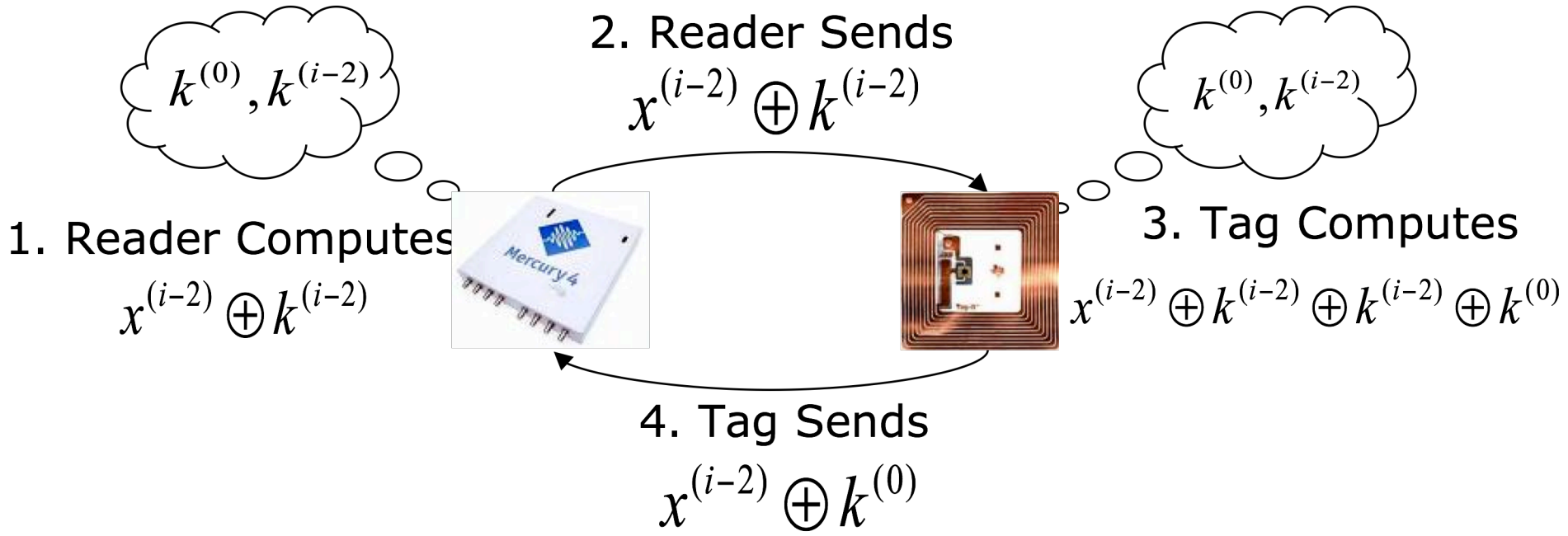
- Introduction to RFID
- Original low-cost RFID authentication protocol
- Implementation Results
- Repeated Keys Attack
- Nibble Attack
- Suggestions for future protocols

Repeated Keys Attack

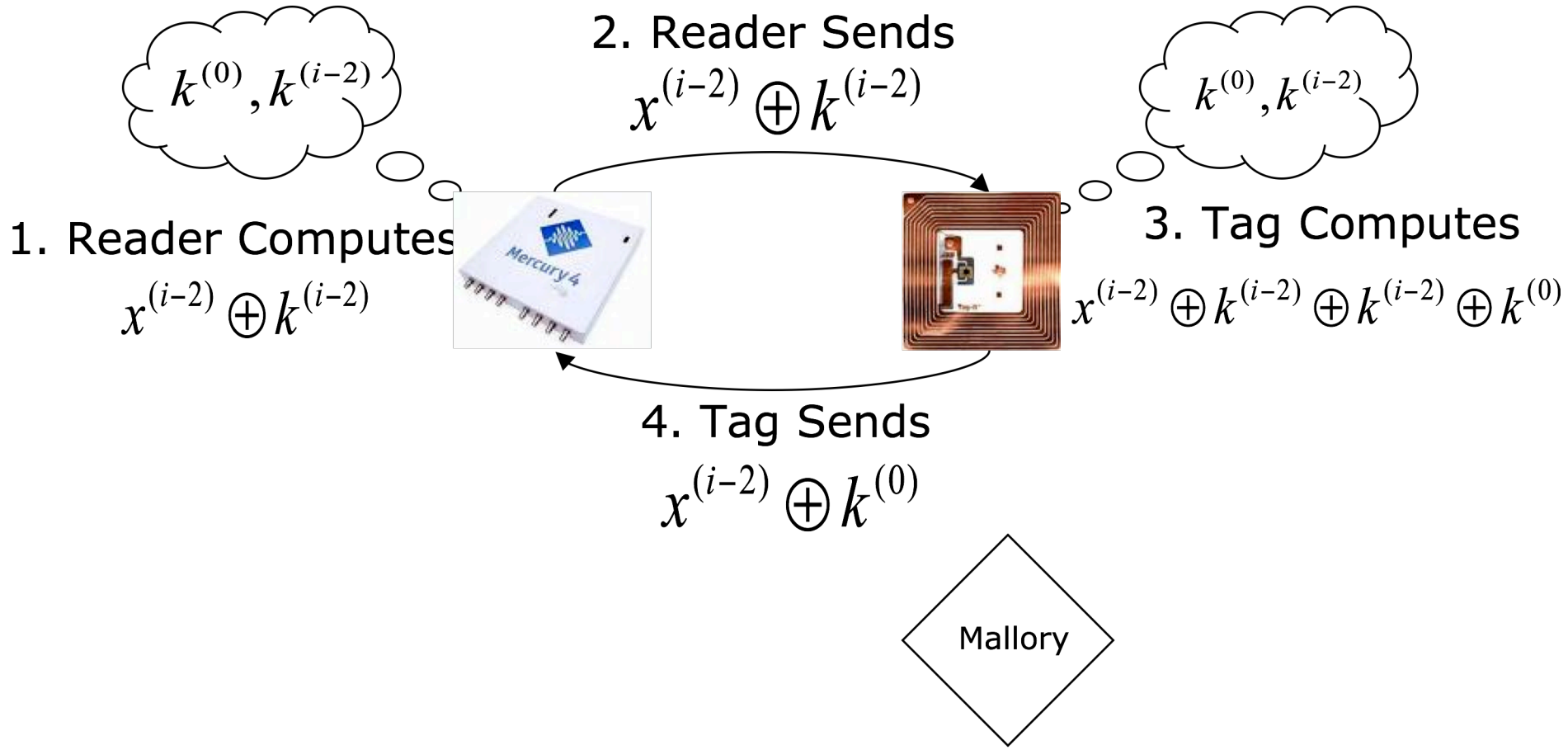
- Can impersonate a tag
- Information is leaked by each transaction
- A passive eavesdropper can impersonate the tag in as few as 3 transactions, on average

Repeated Keys Attack, Transaction *i*-2 $k^{(i)} = k^{(i-2)}$

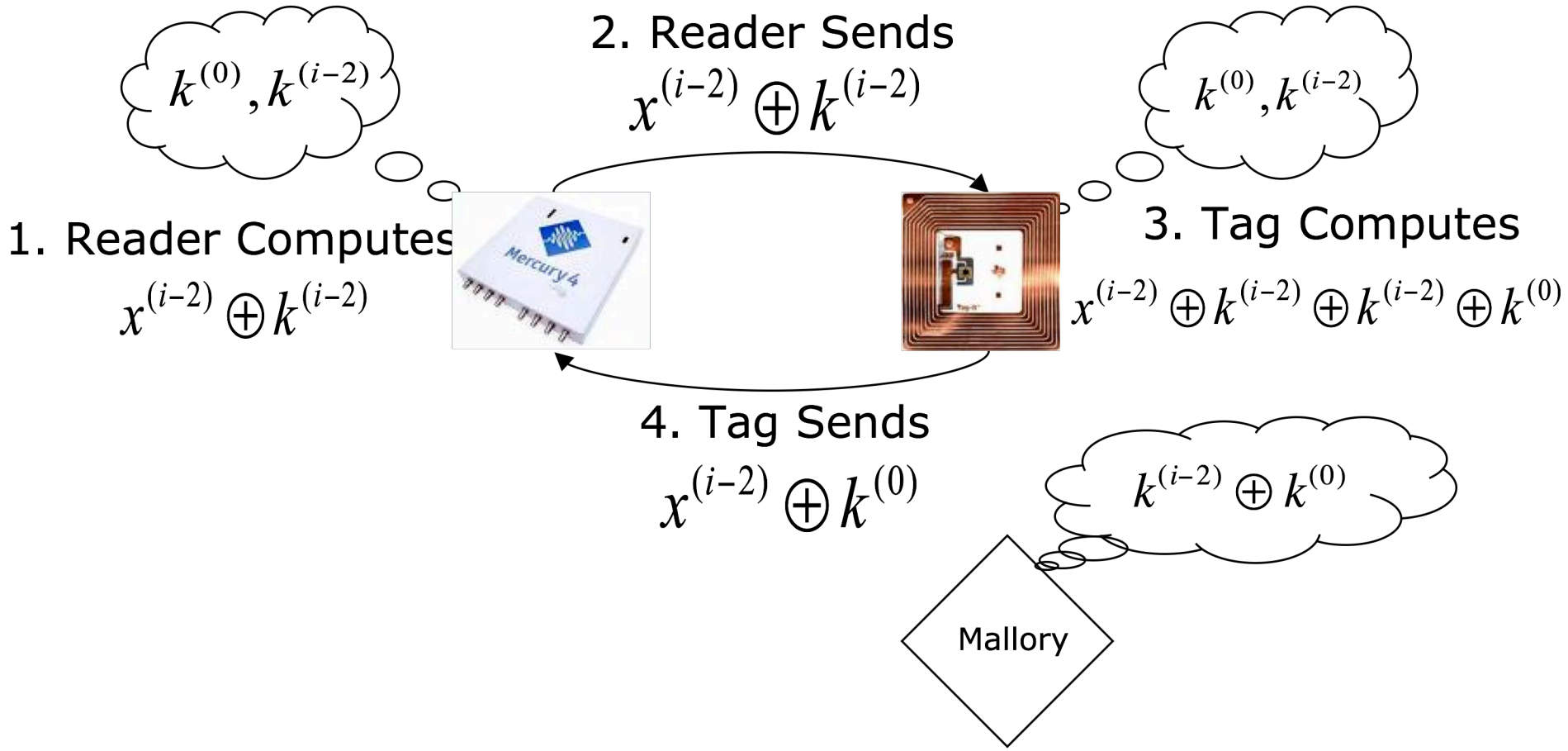
Repeated Keys Attack, Transaction *i-2* $k^{(i)} = k^{(i-2)}$



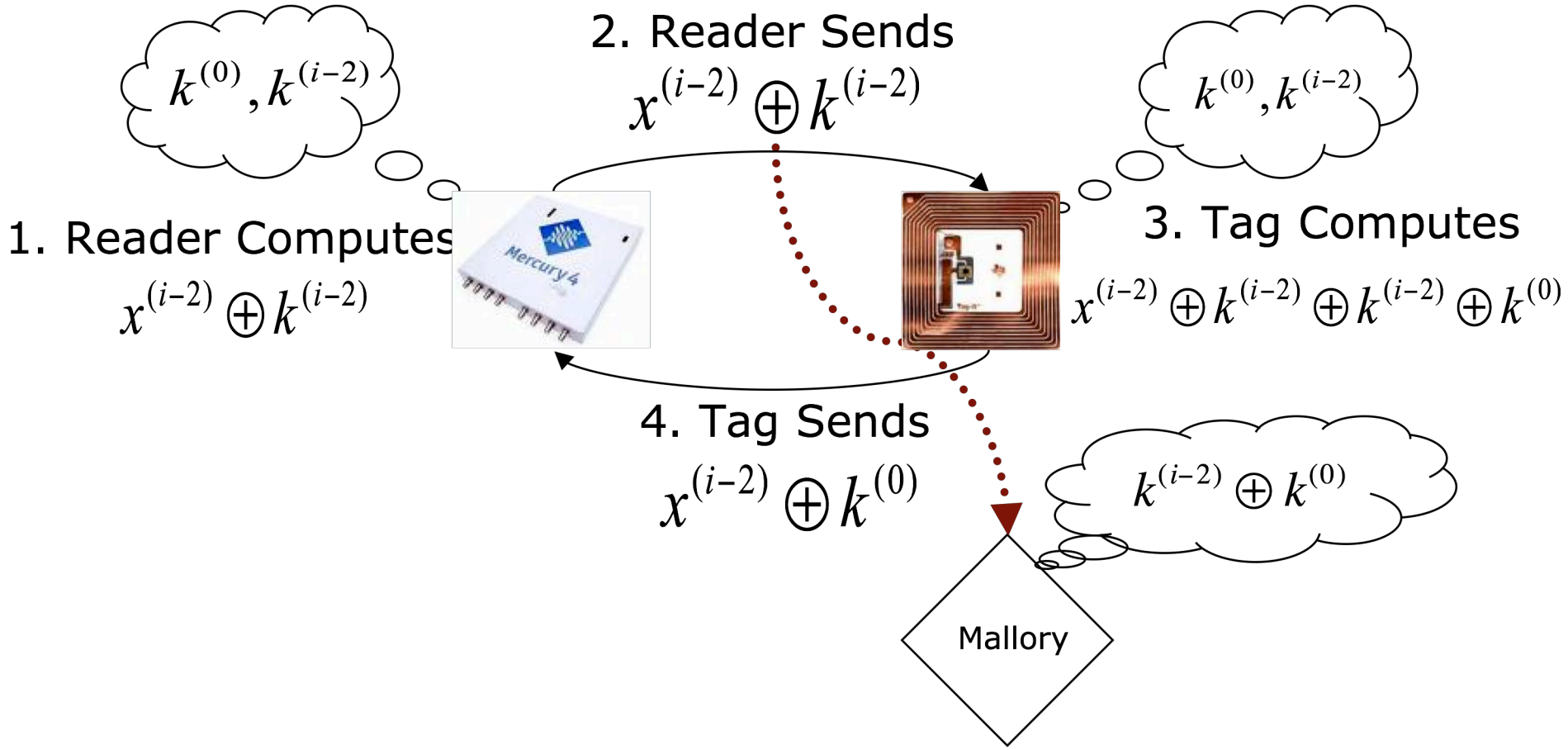
Repeated Keys Attack, Transaction *i*-2 $k^{(i)} = k^{(i-2)}$



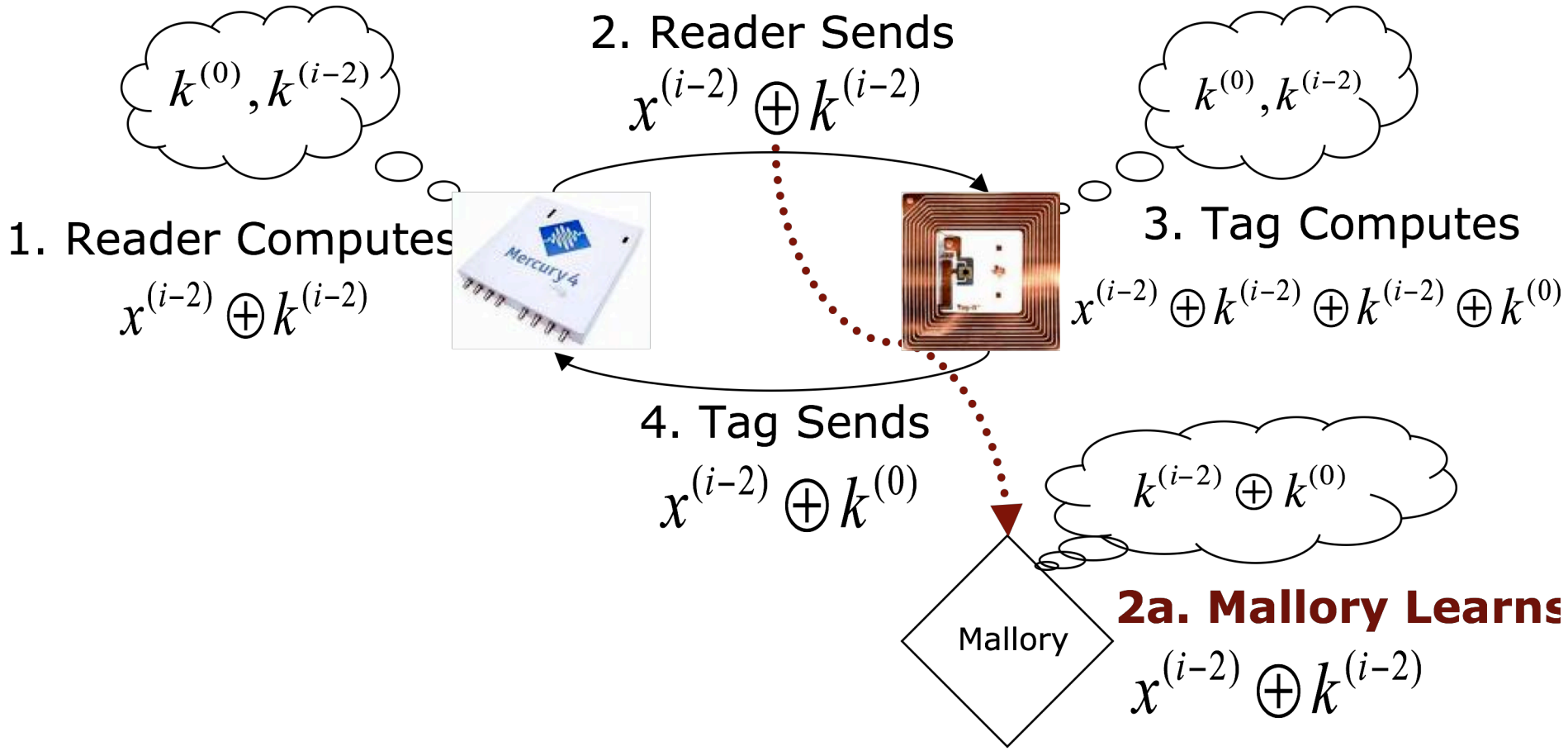
Repeated Keys Attack, Transaction *i*-2 $k^{(i)} = k^{(i-2)}$



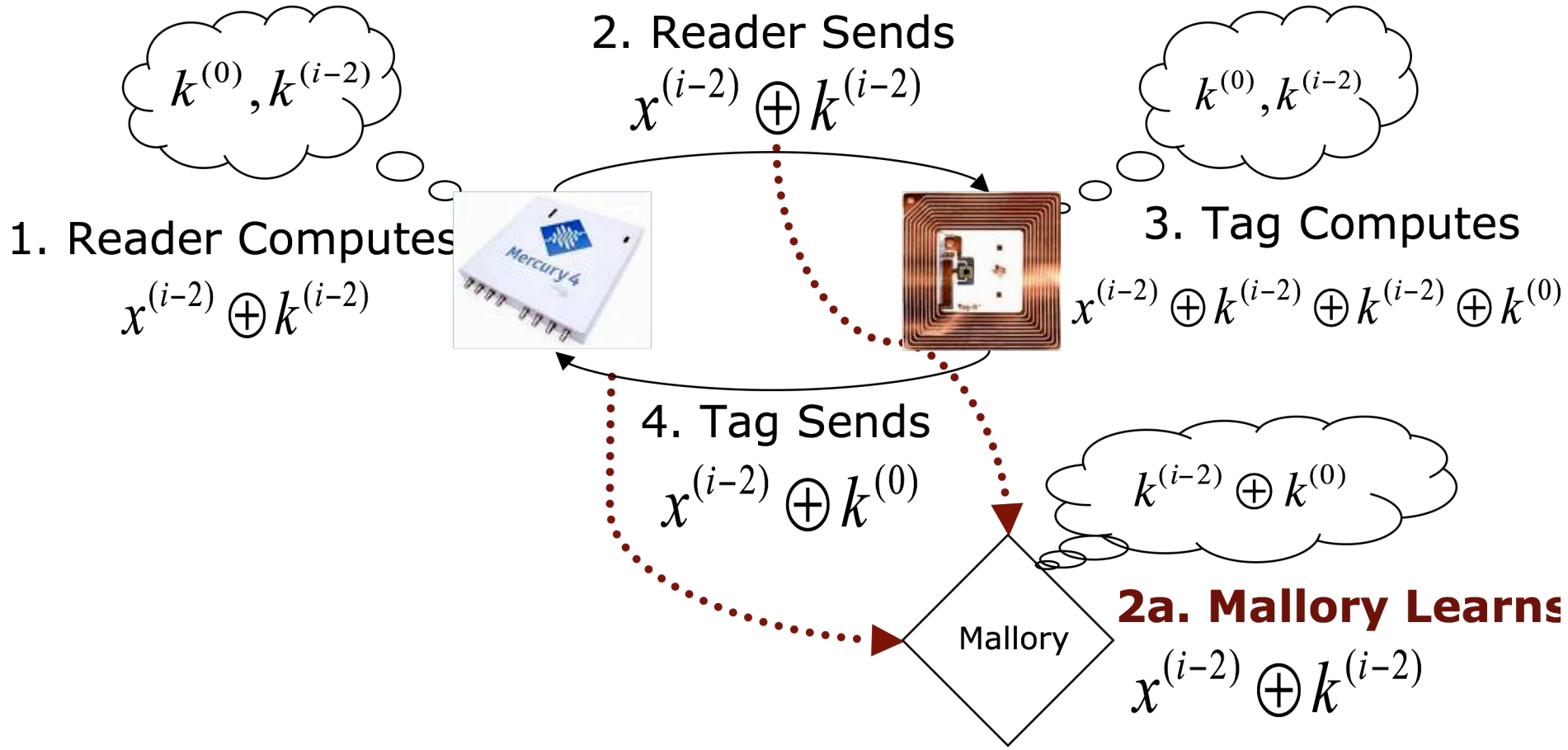
Repeated Keys Attack, Transaction *i*-2 $k^{(i)} = k^{(i-2)}$



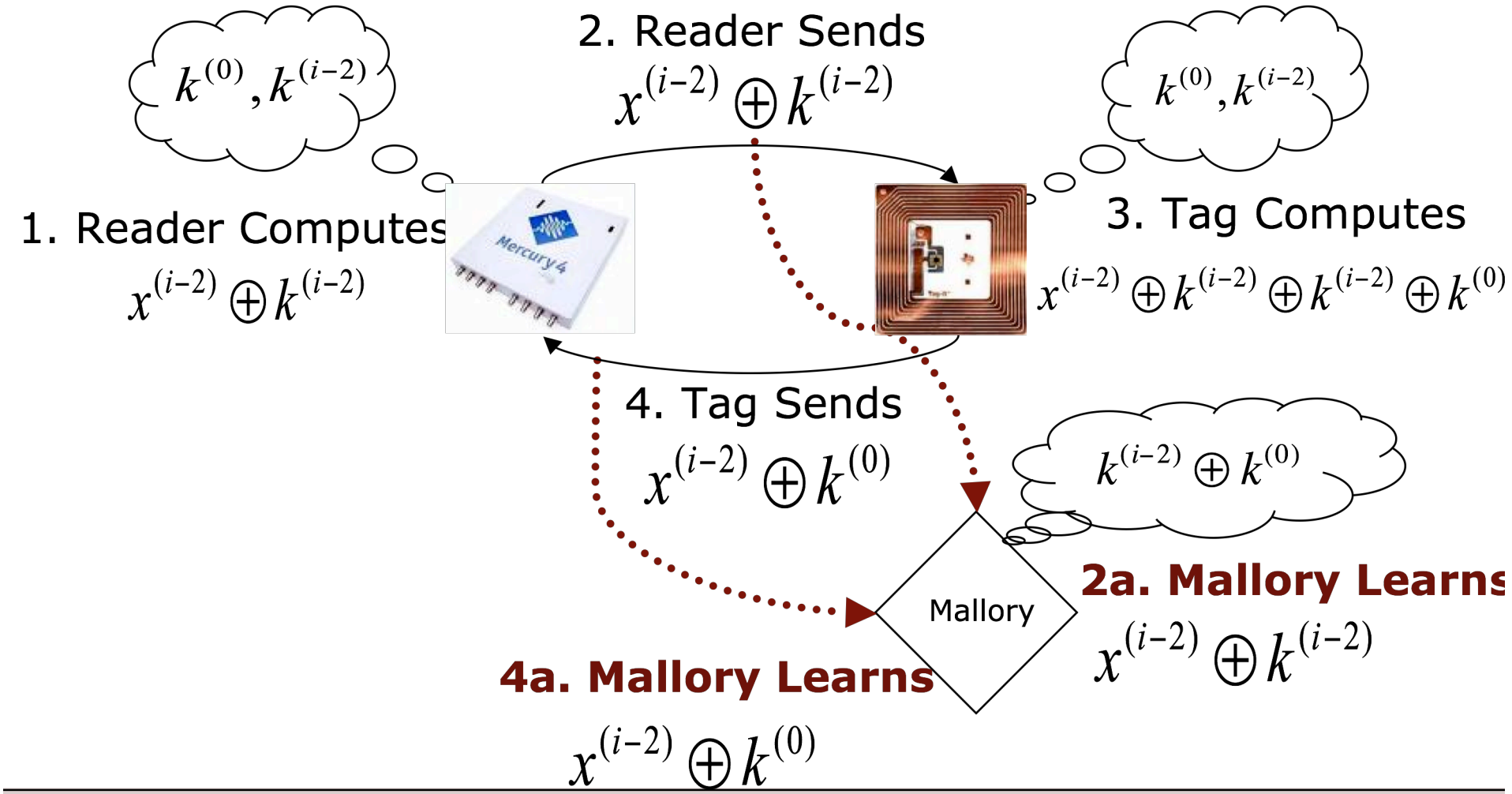
Repeated Keys Attack, Transaction *i-2* $k^{(i)} = k^{(i-2)}$



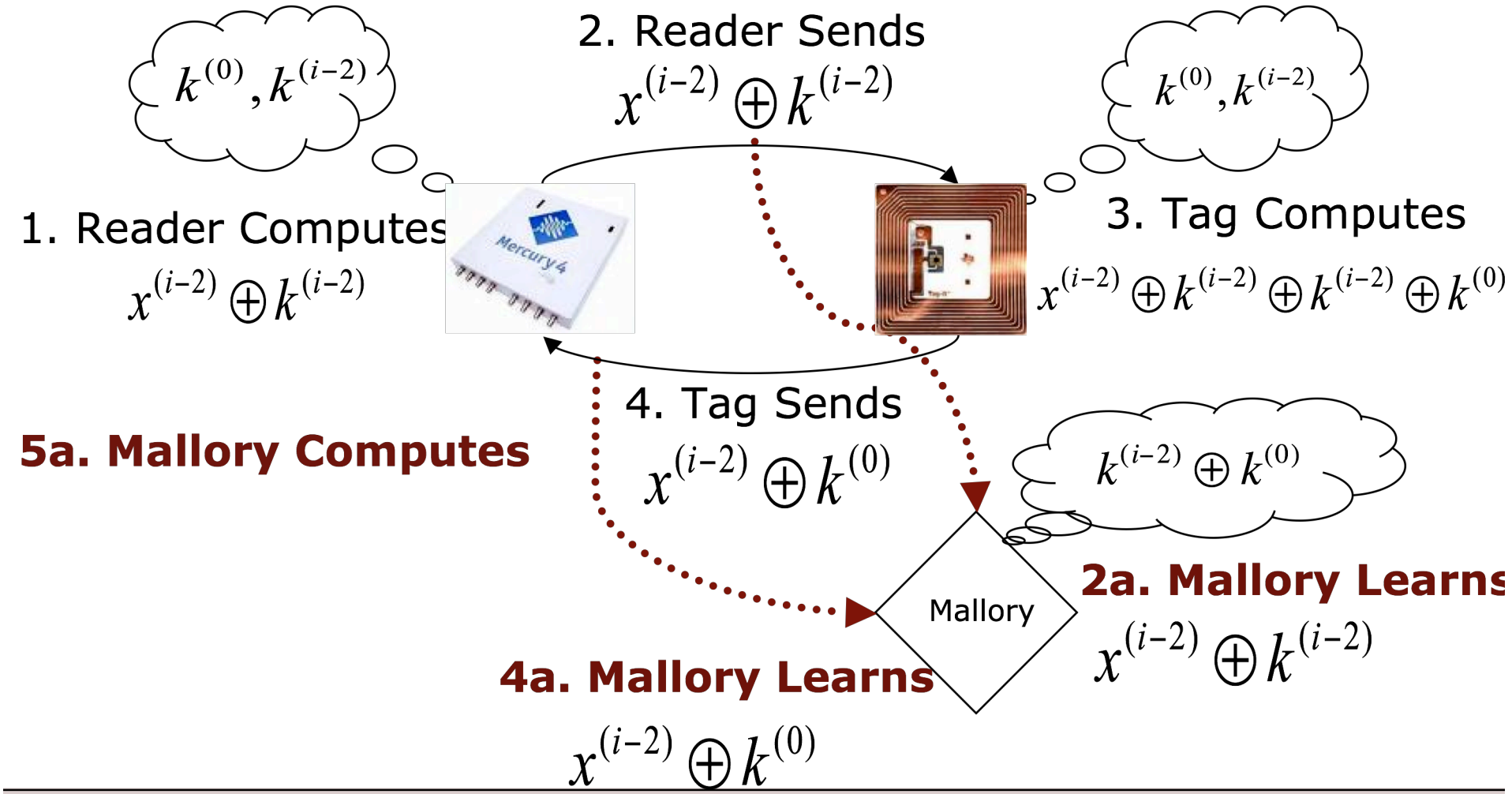
Repeated Keys Attack, Transaction *i-2* $k^{(i)} = k^{(i-2)}$



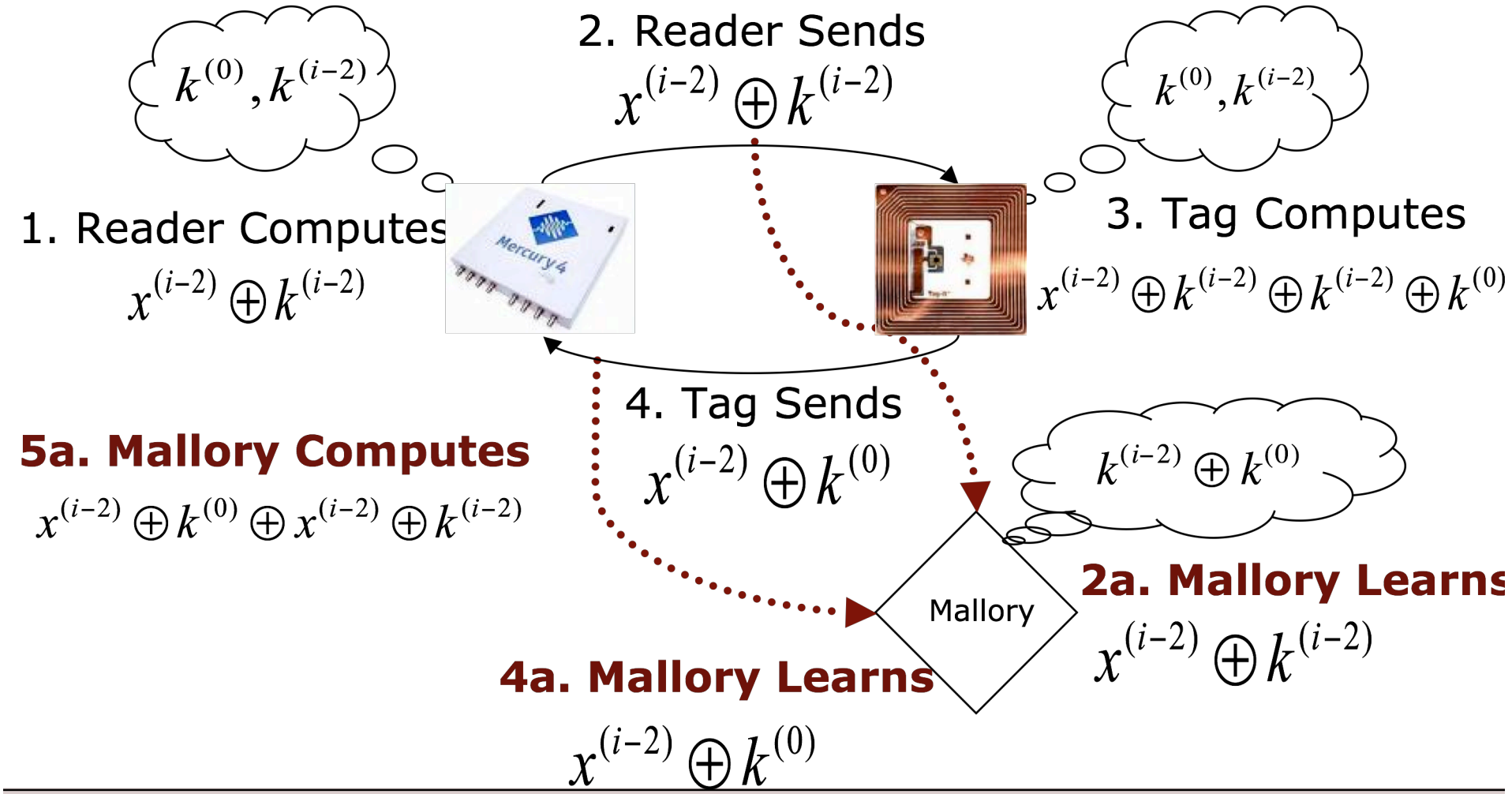
Repeated Keys Attack, Transaction *i-2* $k^{(i)} = k^{(i-2)}$



Repeated Keys Attack, Transaction *i-2* $k^{(i)} = k^{(i-2)}$



Repeated Keys Attack, Transaction *i-2* $k^{(i)} = k^{(i-2)}$



Repeated Keys Attack, Transaction *i-2* $k^{(i)} = k^{(i-2)}$

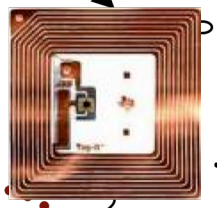
$k^{(0)}, k^{(i-2)}$

1. Reader Computes

$x^{(i-2)} \oplus k^{(i-2)}$

2. Reader Sends

$x^{(i-2)} \oplus k^{(i-2)}$



$k^{(0)}, k^{(i-2)}$

3. Tag Computes

$x^{(i-2)} \oplus k^{(i-2)} \oplus k^{(i-2)} \oplus k^{(0)}$

4. Tag Sends

$x^{(i-2)} \oplus k^{(0)}$

5a. Mallory Computes

$x^{(i-2)} \oplus k^{(0)} \oplus x^{(i-2)} \oplus k^{(i-2)}$

$k^{(i-2)} \oplus k^{(0)}$

$k^{(i-2)} \oplus k^{(0)}$

Mallory

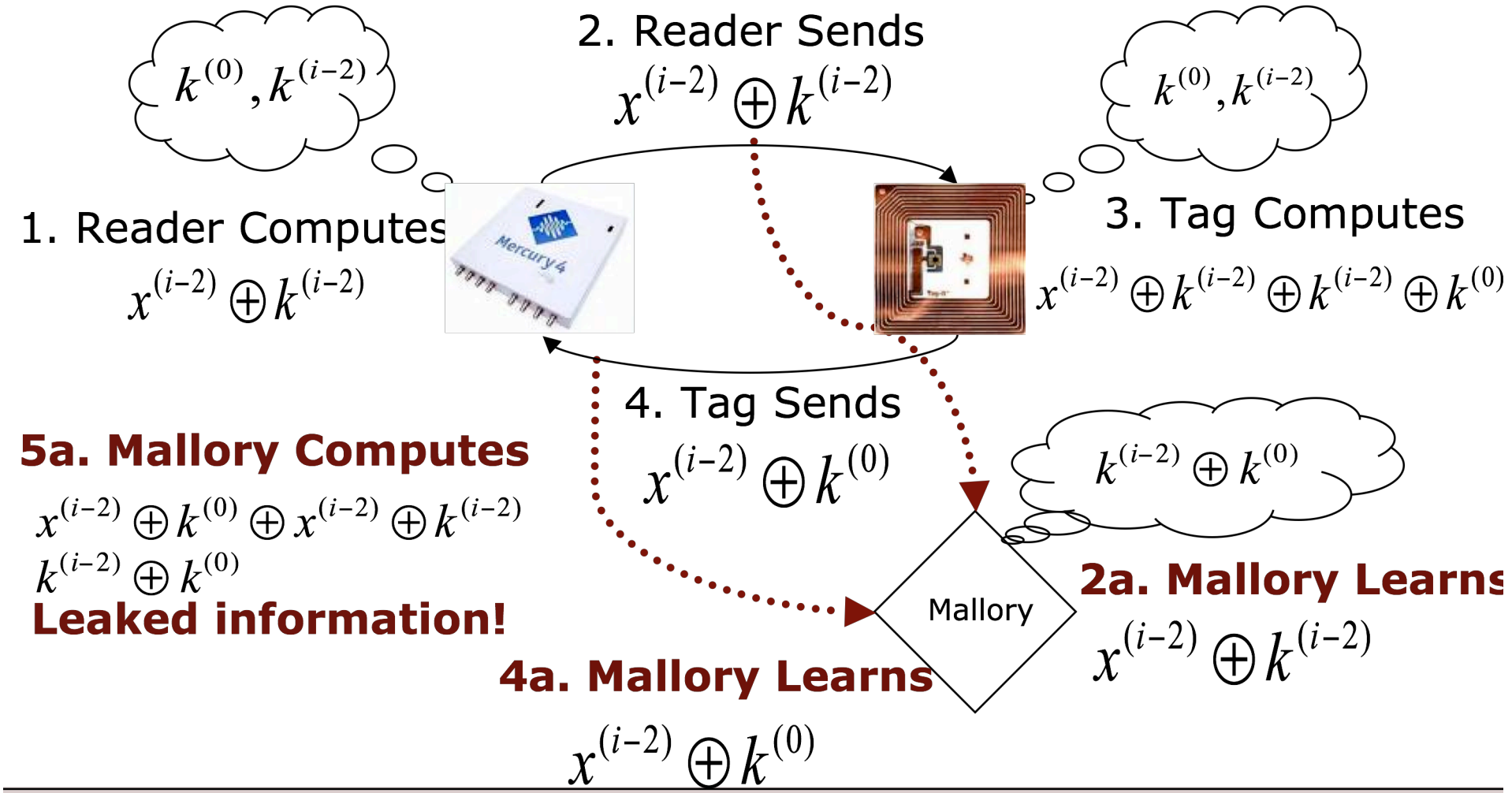
2a. Mallory Learns

$x^{(i-2)} \oplus k^{(i-2)}$

4a. Mallory Learns

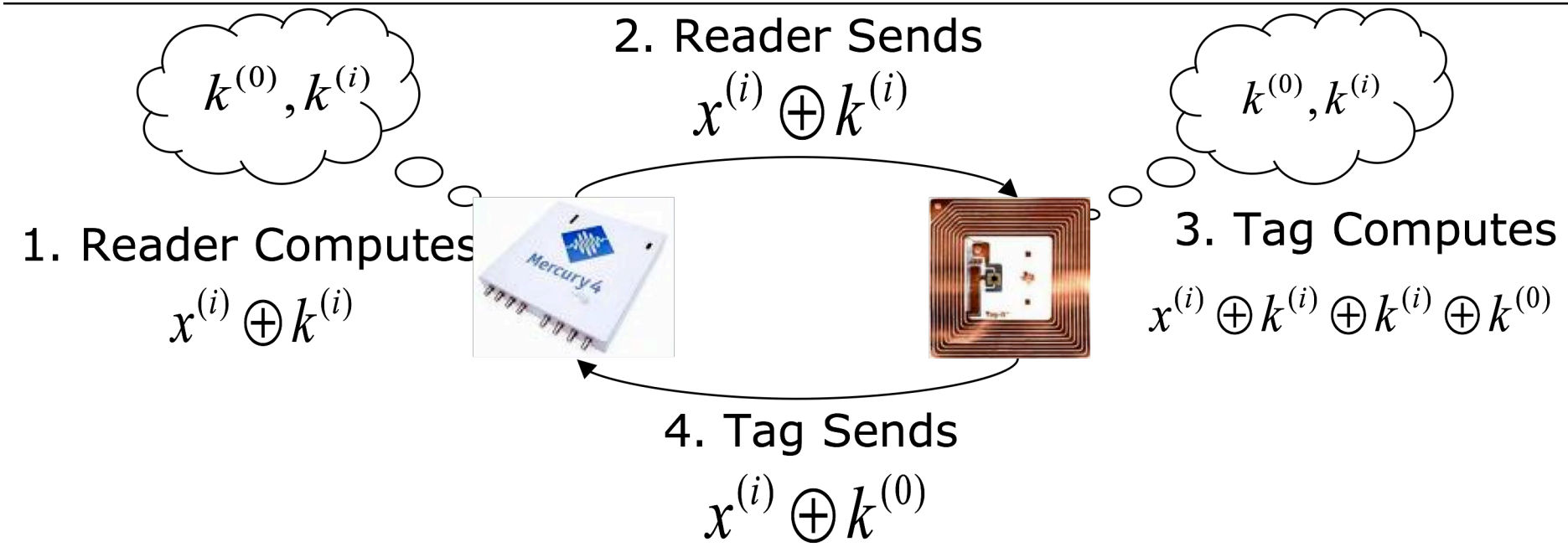
$x^{(i-2)} \oplus k^{(0)}$

Repeated Keys Attack, Transaction *i-2* $k^{(i)} = k^{(i-2)}$

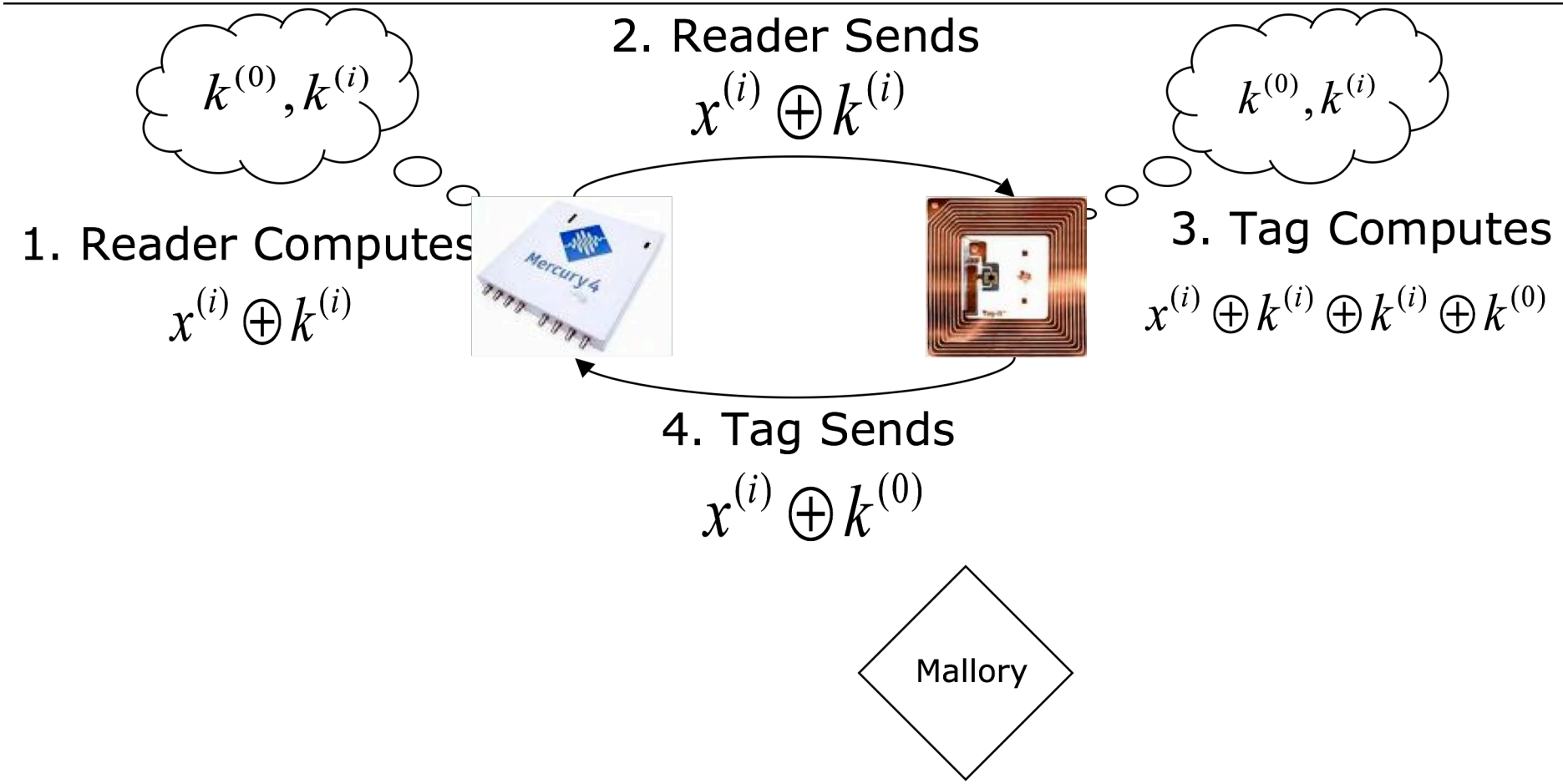


Repeated Keys Attack, Transaction i $k^{(i)} = k^{(i-2)}$

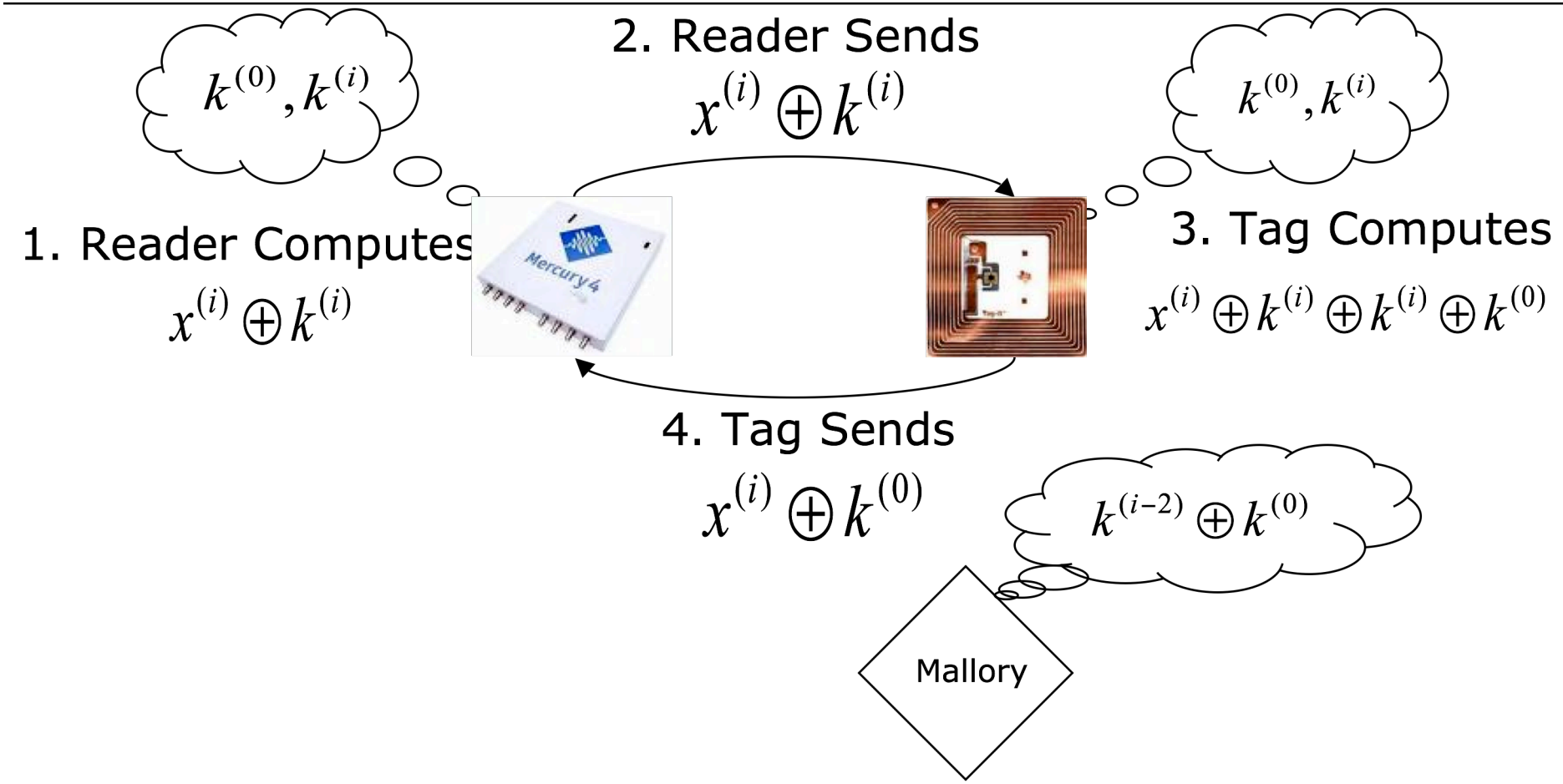
Repeated Keys Attack, Transaction i $k^{(i)} = k^{(i-2)}$



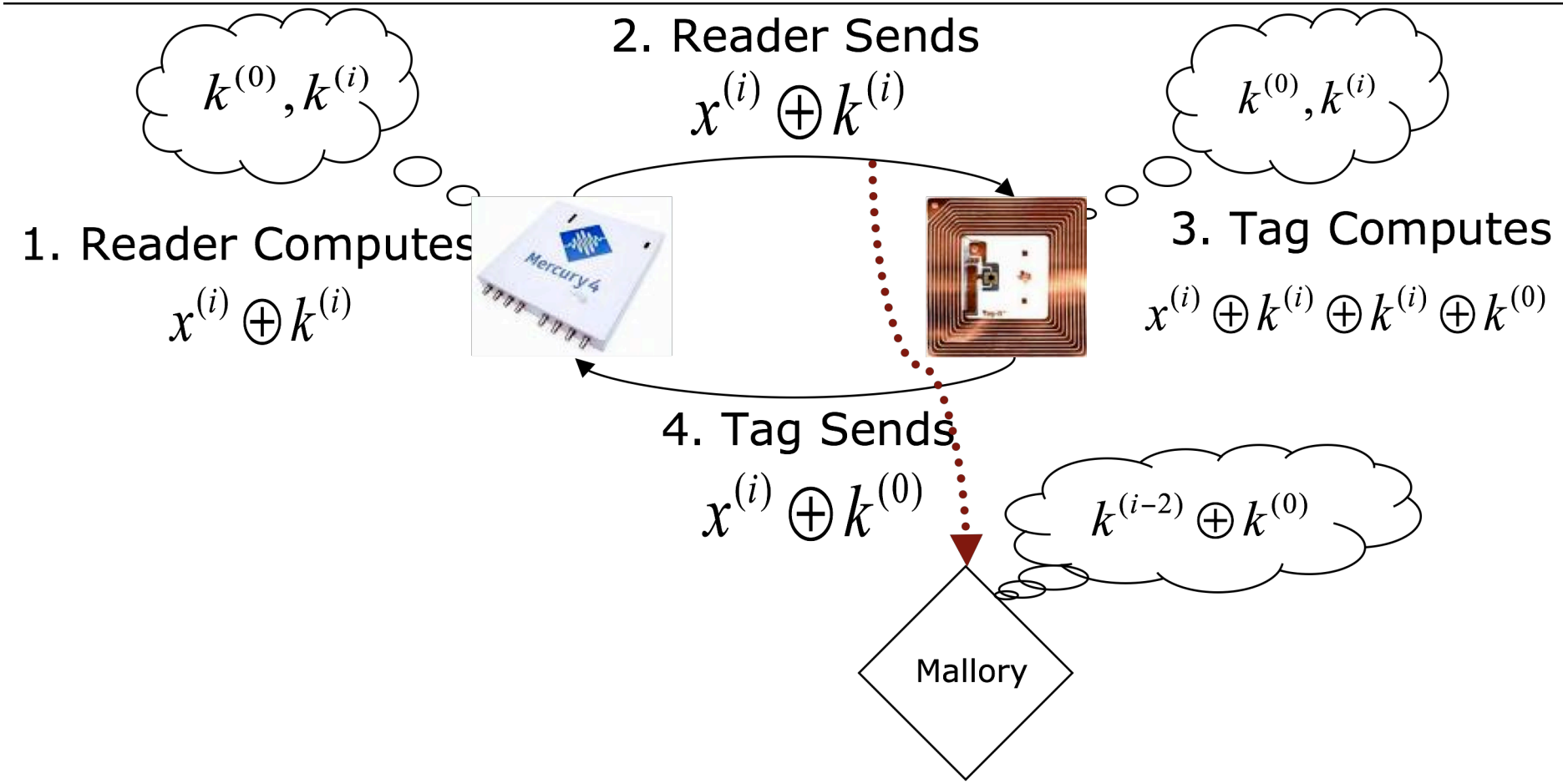
Repeated Keys Attack, Transaction i $k^{(i)} = k^{(i-2)}$



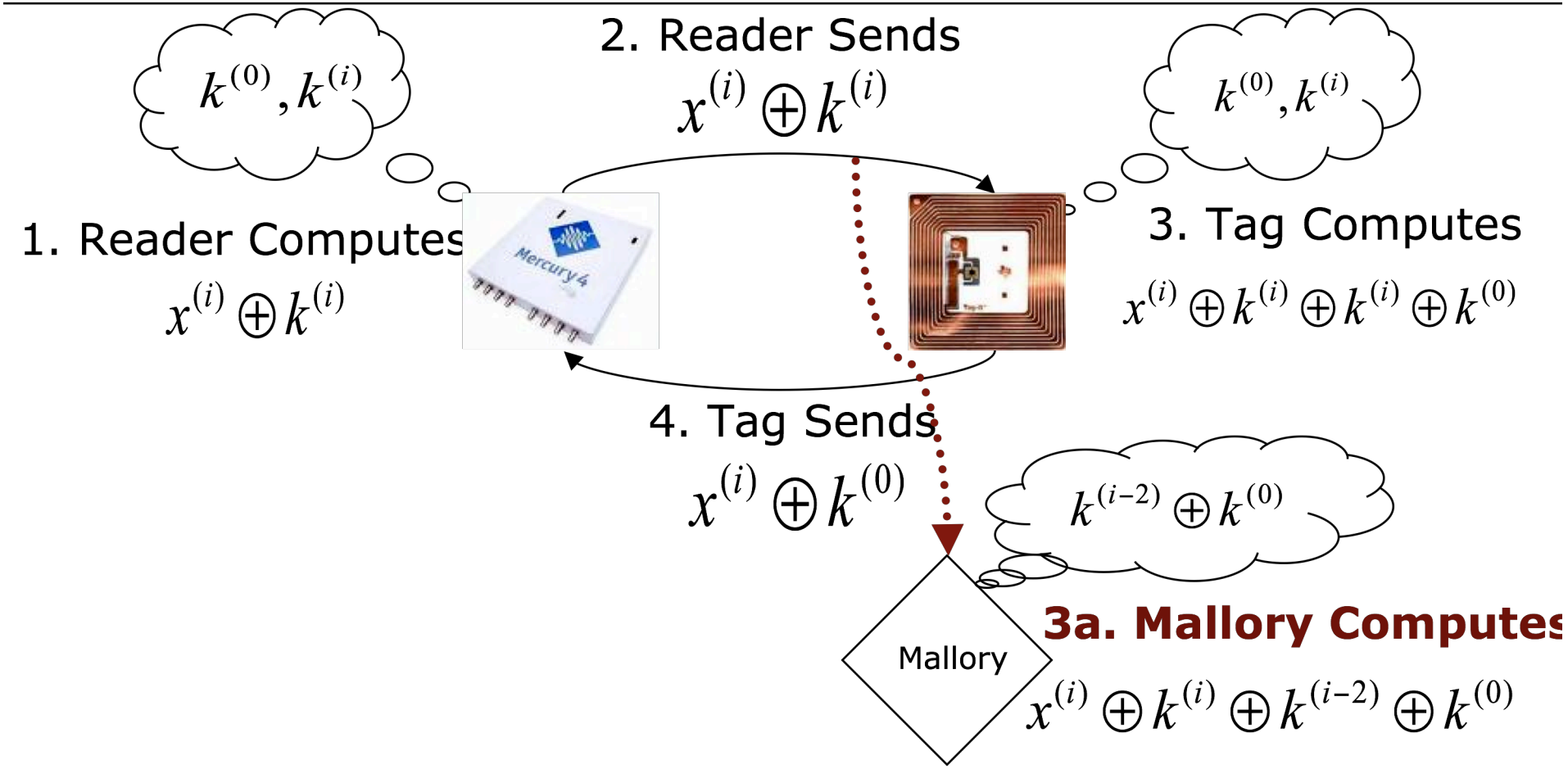
Repeated Keys Attack, Transaction i $k^{(i)} = k^{(i-2)}$



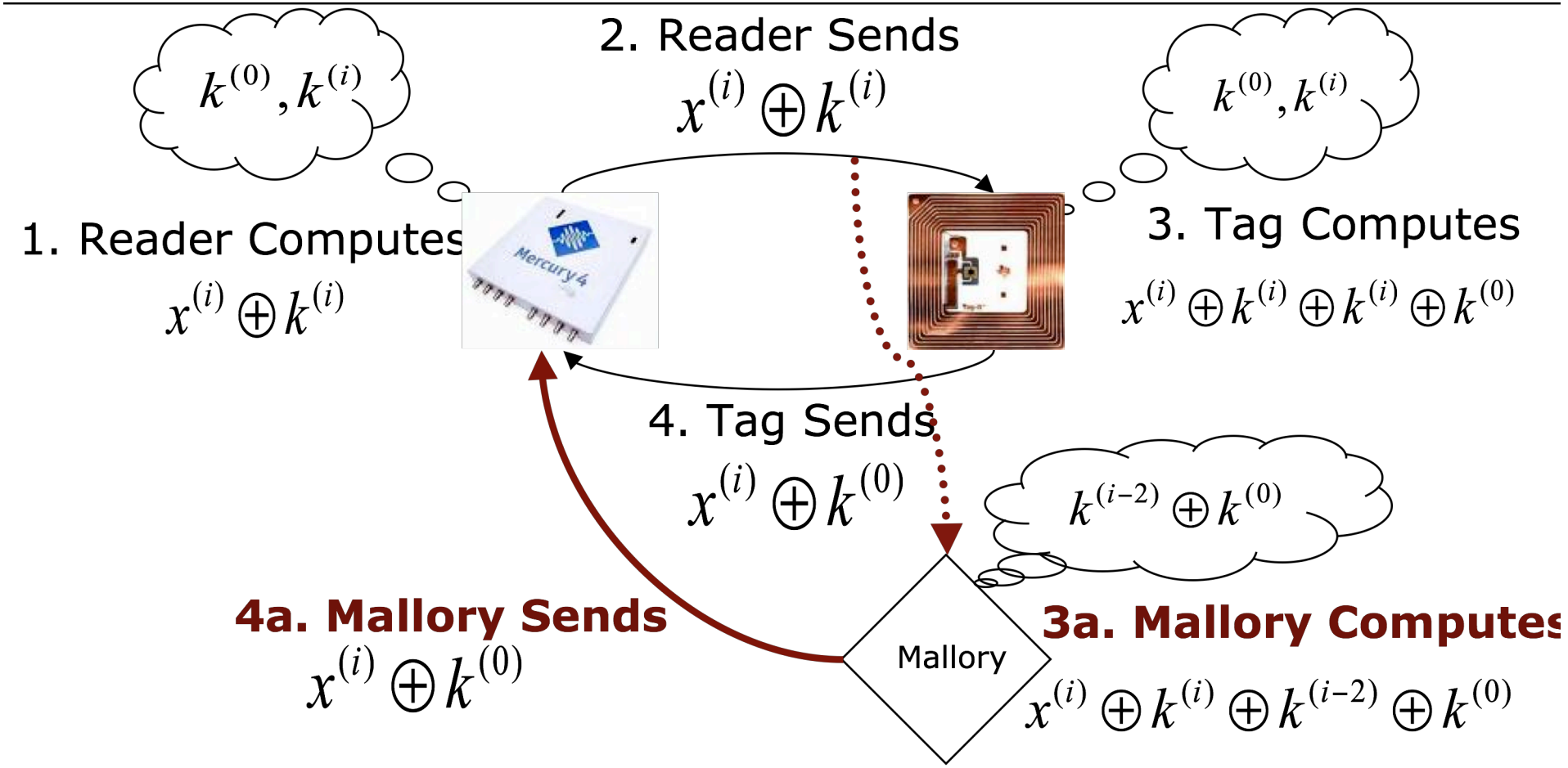
Repeated Keys Attack, Transaction i $k^{(i)} = k^{(i-2)}$



Repeated Keys Attack, Transaction i $k^{(i)} = k^{(i-2)}$



Repeated Keys Attack, Transaction i $k^{(i)} = k^{(i-2)}$



Repeated Keys Attack

- Mallory knows information leaked from an earlier transaction
- XOR's this information with current challenge to get valid response
- Does *not* know the session key or shared secret
- Does *not* brute force

Implications of Repeated Keys Attack

- A passive eavesdropper can impersonate the tag after an average of:
  - 70 transactions if listening from start
  - 3 transactions if listening after 68th transaction
- Vajda and Buttyán gave a theoretical maximum of $16! \times 2 = 4.18455798 \times 10^{13}$ transactions

Outline

- Introduction to RFID
- Original low-cost RFID authentication protocol
- Implementation Results
- Repeated Keys Attack
- Nibble Attack
- Suggestions for future protocols

Nibble Attack

- Exploits properties of the session key function
- Consider 128-bit key length example
- Attacker can learn shared secret after observing an expected 1,092 transactions

Session Key $k^{(i)}$

$k^{(i)} =$ | 1010 | 0100 | 1000 | 0110 | ... | 0001 | 1100 | 0111 | 0010 |

$k_{0,L}^{(i)}$ $k_{0,R}^{(i)}$ $k_{1,L}^{(i)}$ $k_{1,R}^{(i)}$ $k_{14,L}^{(i)}$ $k_{14,R}^{(i)}$ $k_{15,L}^{(i)}$ $k_{15,R}^{(i)}$

$k_L^{(i)} = \left\{k_{0,L}^{(i)}, k_{1,L}^{(i)}, \ldots, k_{15,L}^{(i)}\right\} =$ | 1010 | 1000 | ... | 0001 | 0111 |

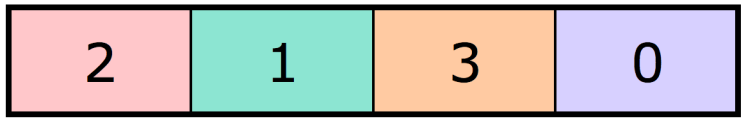
$k_R^{(i)} = \left\{k_{0,R}^{(i)}, k_{1,R}^{(i)}, \ldots, k_{15,R}^{(i)}\right\} =$ | 0100 | 0110 | ... | 1100 | 0010 |

Session Key Function $k^{(i+1)} = F(k^{(i)})$

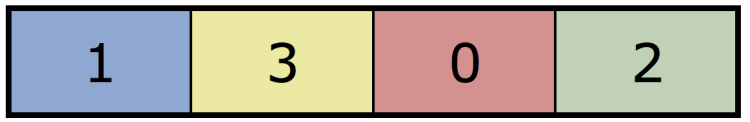
- $k^{(i+1)}$ is formed by moving nibbles of the left and right vectors of $k^{(i)}$
- $k_L^{(i+1)}$ is formed in a similar way using elements of $k_R^{(i)}$ to swap elements of $k_L^{(i)}$
- $k^{(i+1)}$ is formed by interleaving $k_L^{(i+1)}$ and $k_R^{(i+1)}$

Session Key Example, Original Protocol

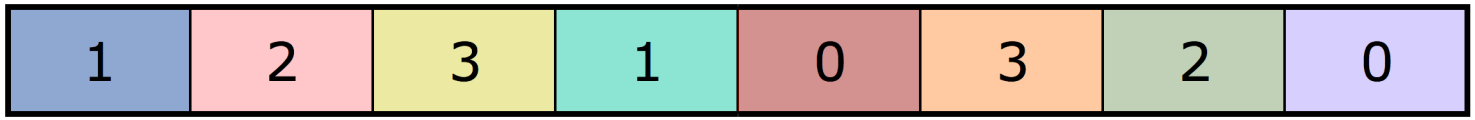
$$k_R^{(i)} = \left\{ k_{0,R}^{(i)}, k_{1,R}^{(i)}, k_{2,R}^{(i)}, k_{3,R}^{(i)} \right\} =$$



$$k_L^{(i)} = \left\{ k_{0,L}^{(i)}, k_{1,L}^{(i)}, k_{2,L}^{(i)}, k_{3,L}^{(i)} \right\} =$$



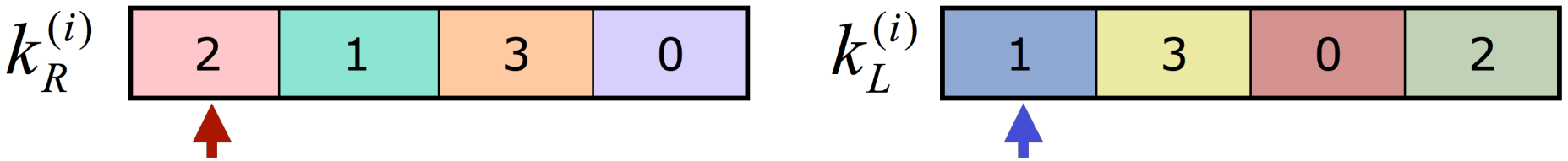
$$k^{(i)} =$$



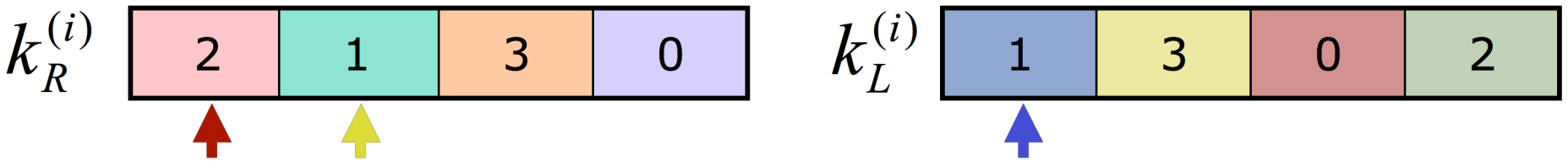
$k_{0,L}^{(i)}$ $k_{0,R}^{(i)}$ $k_{1,L}^{(i)}$ $k_{1,R}^{(i)}$ $k_{2,L}^{(i)}$ $k_{2,R}^{(i)}$ $k_{3,L}^{(i)}$ $k_{3,R}^{(i)}$

Session Key Example, Original Protocol

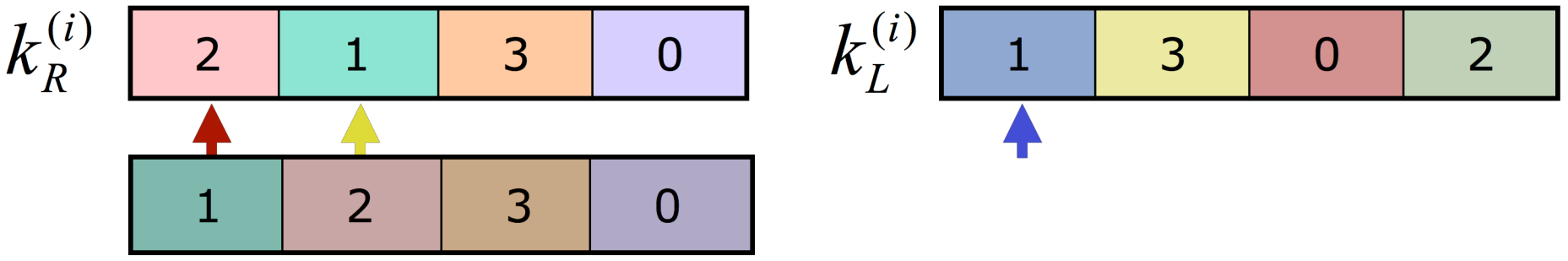
Session Key Example, Original Protocol



Session Key Example, Original Protocol



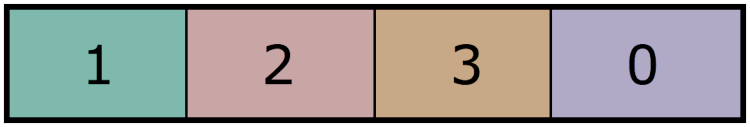
Session Key Example, Original Protocol



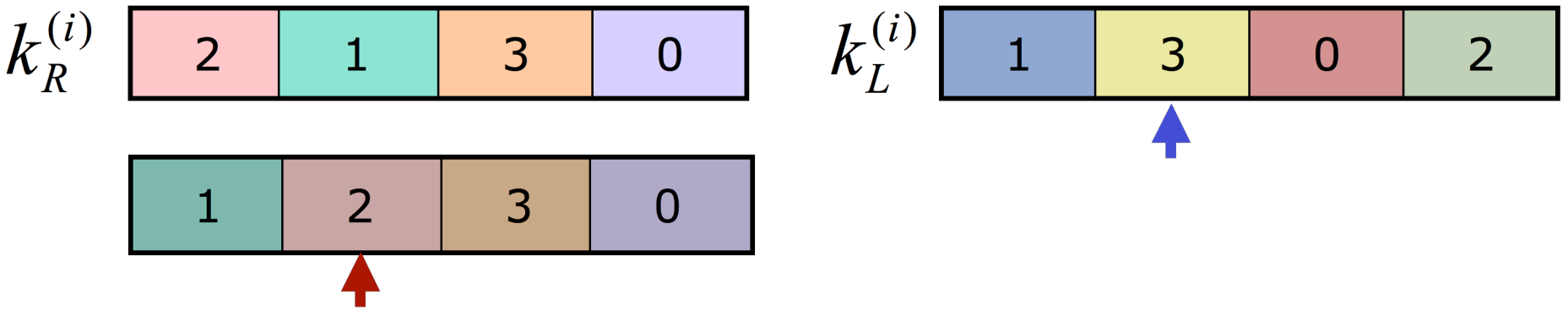
Session Key Example, Original Protocol

$k_R^{(i)}$

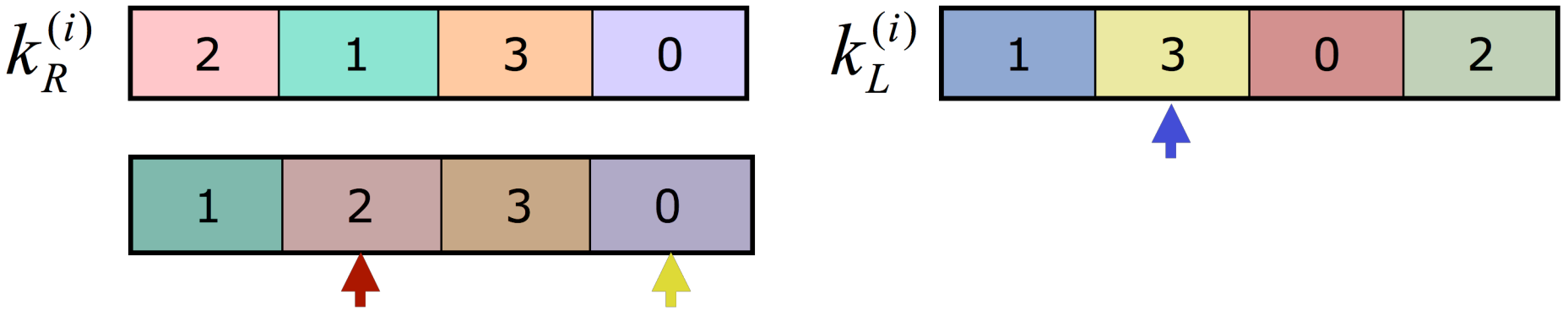
$k_L^{(i)}$



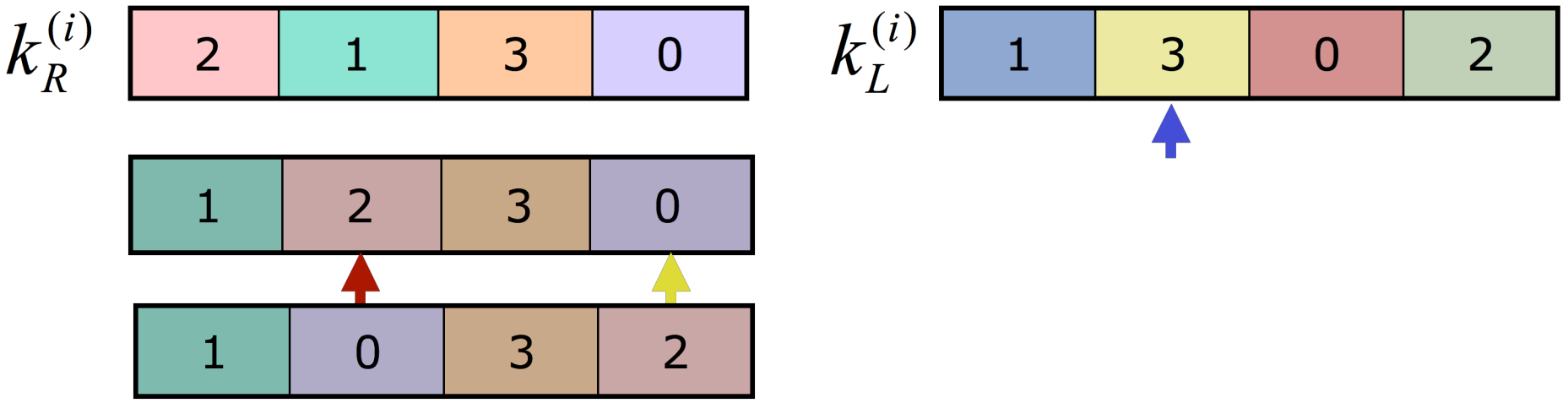
Session Key Example, Original Protocol



Session Key Example, Original Protocol



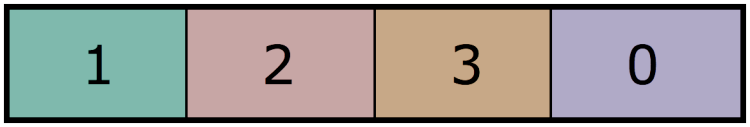
Session Key Example, Original Protocol



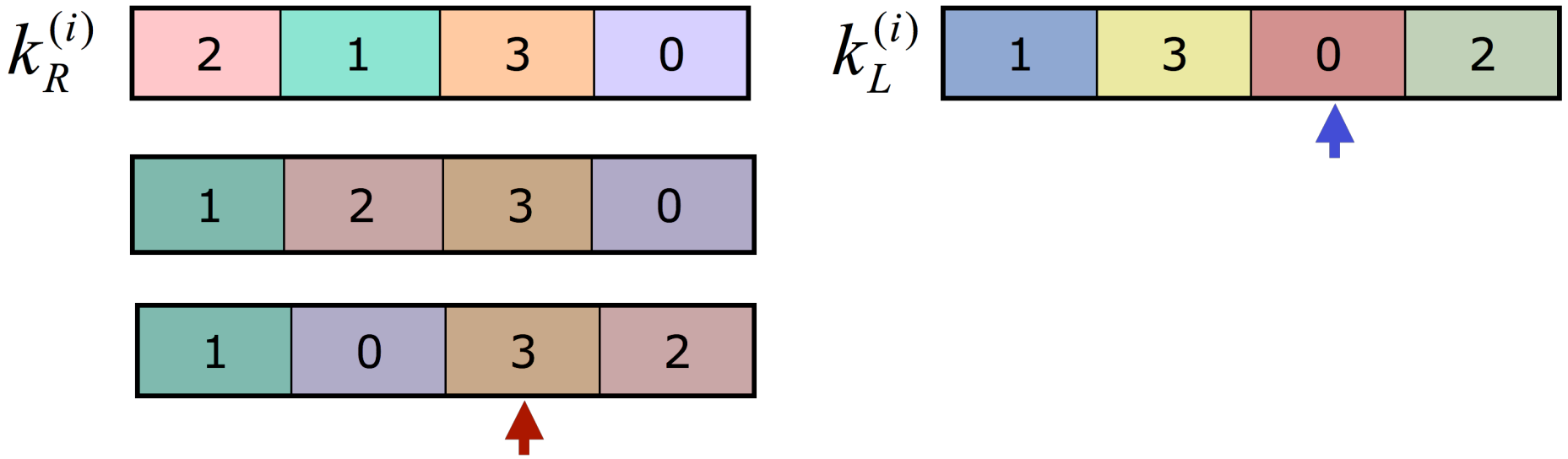
Session Key Example, Original Protocol

$k_R^{(i)}$

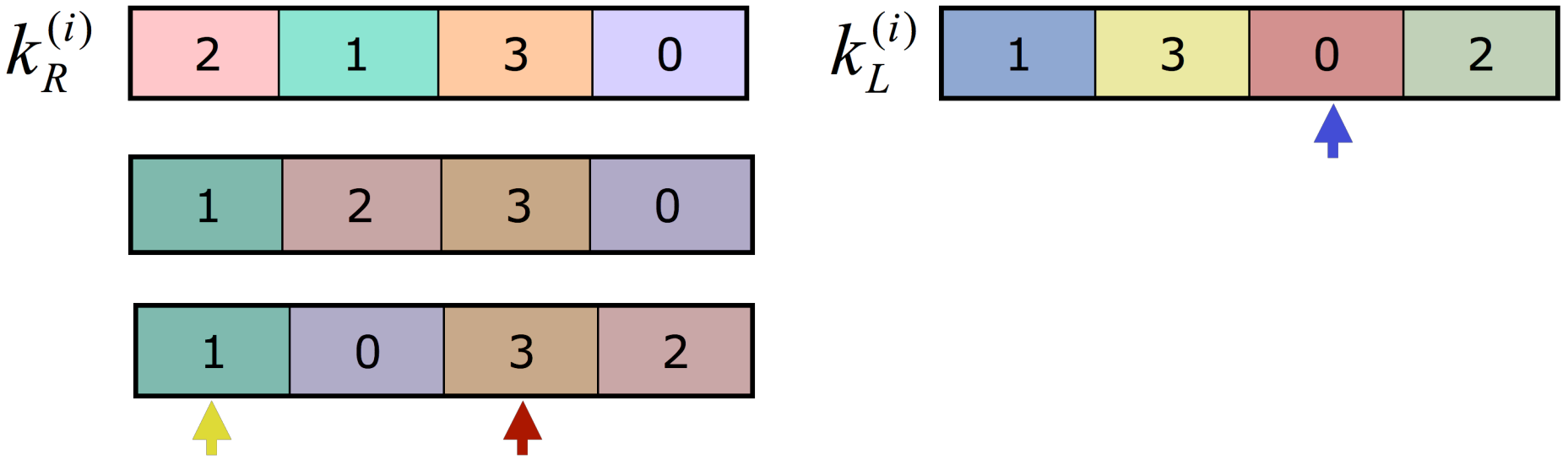
$k_L^{(i)}$



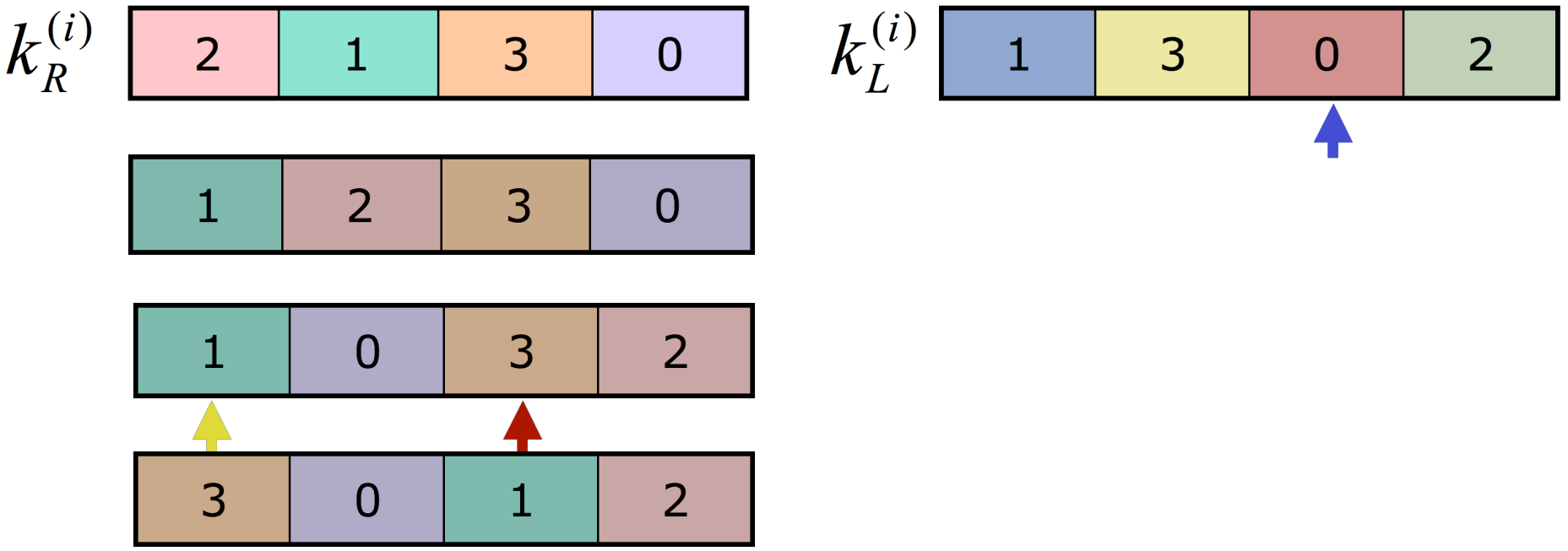
Session Key Example, Original Protocol



Session Key Example, Original Protocol



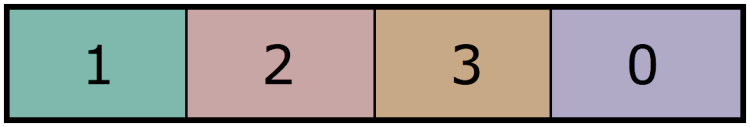
Session Key Example, Original Protocol



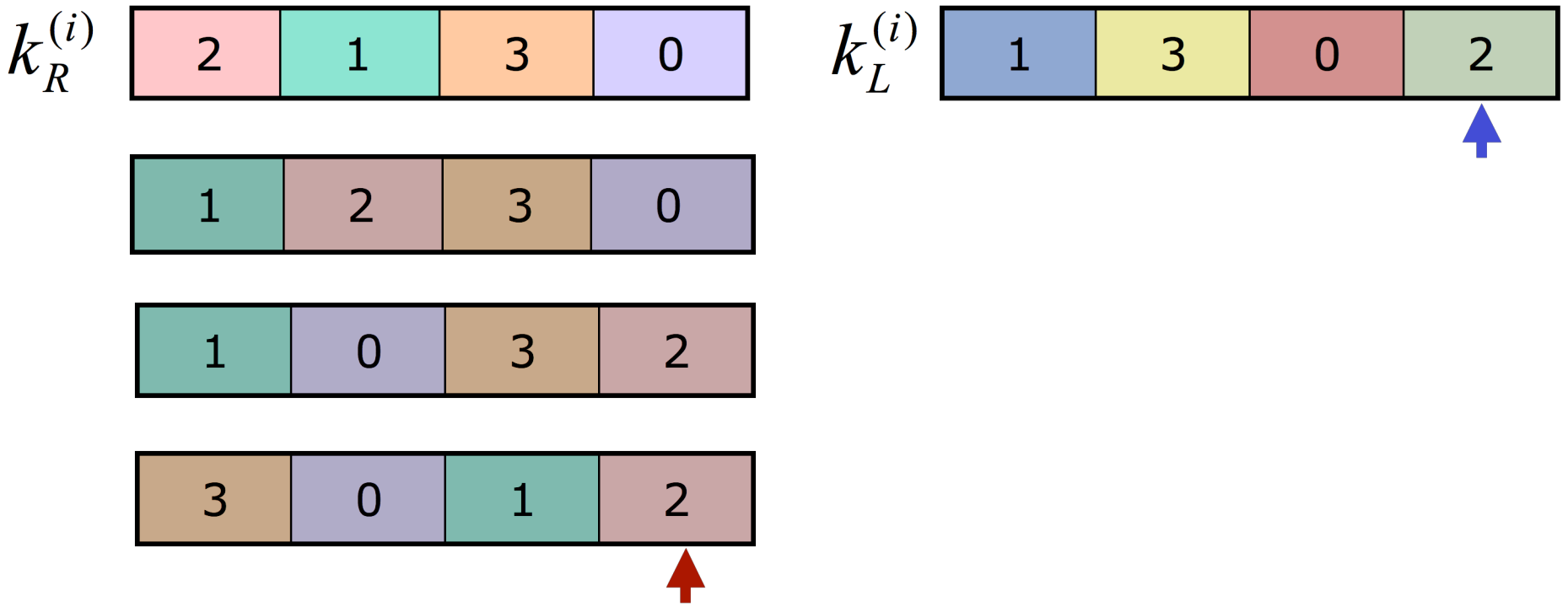
Session Key Example, Original Protocol

$k_R^{(i)}$

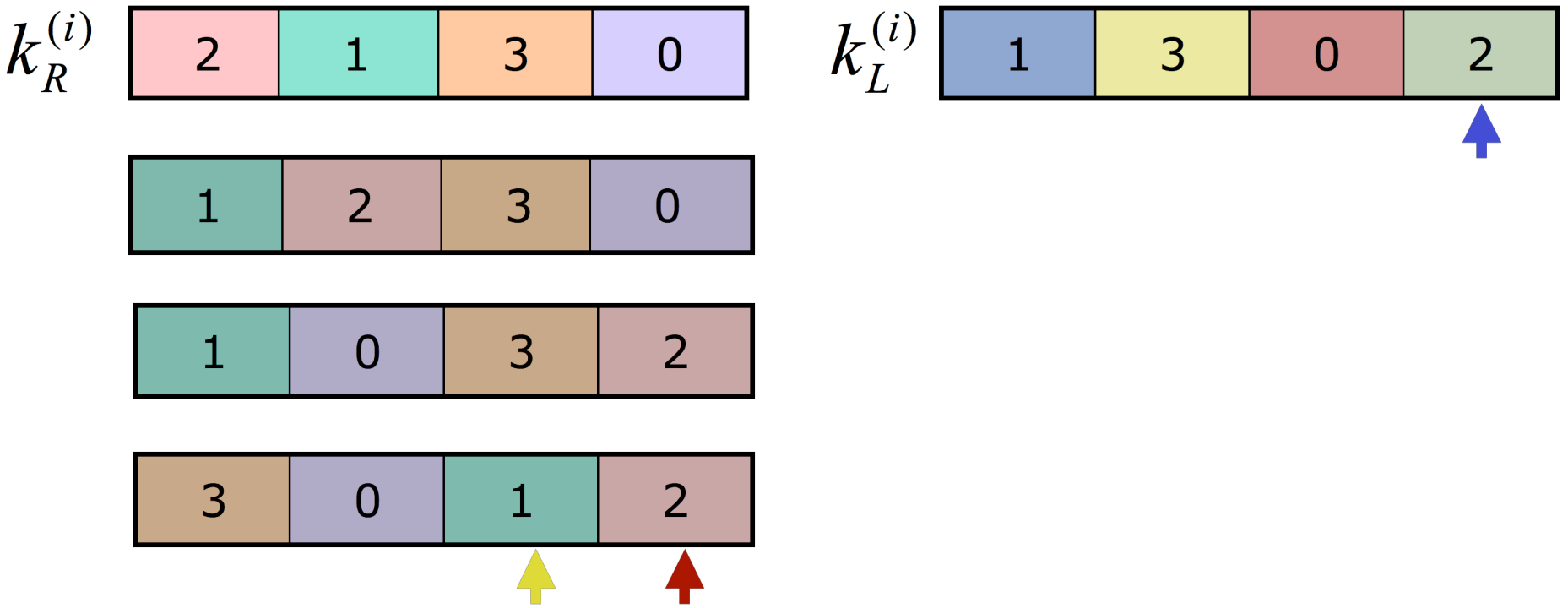
$k_L^{(i)}$



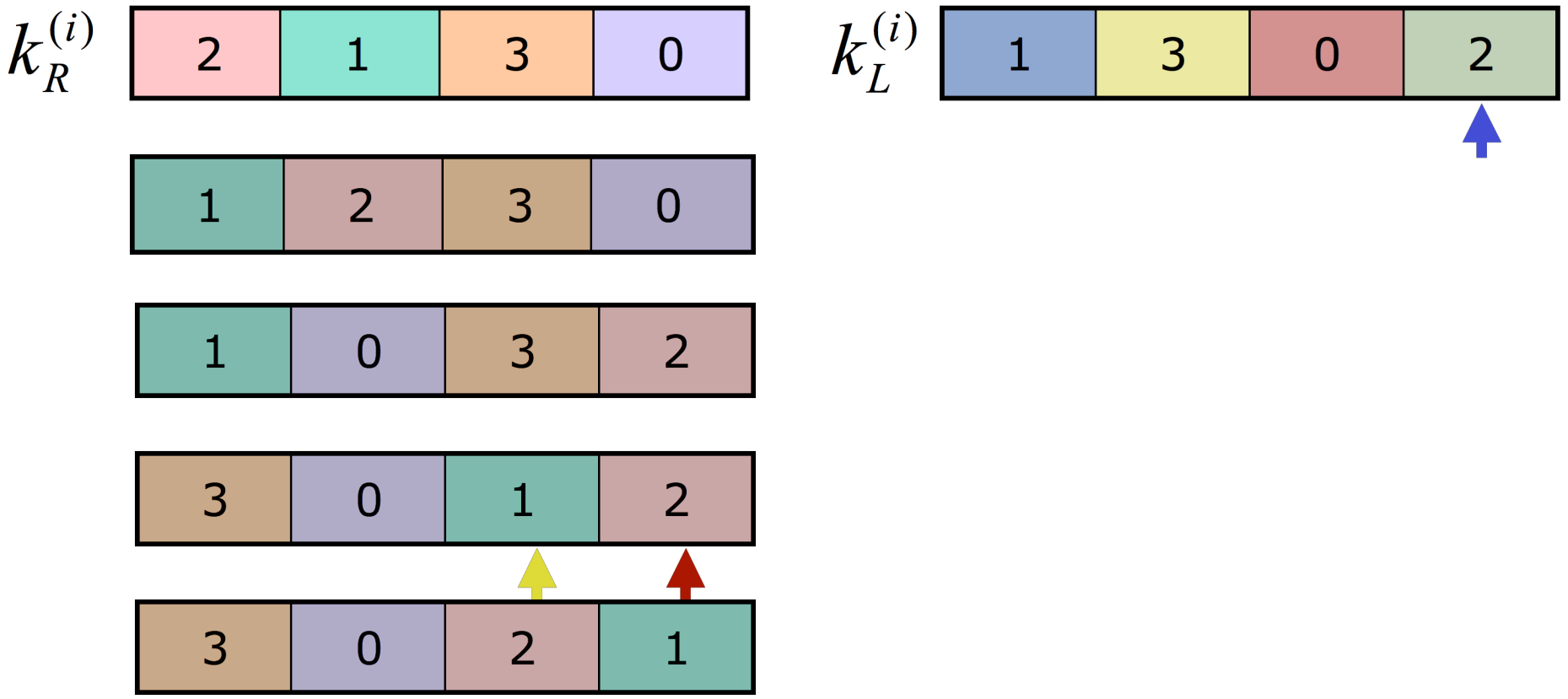
Session Key Example, Original Protocol



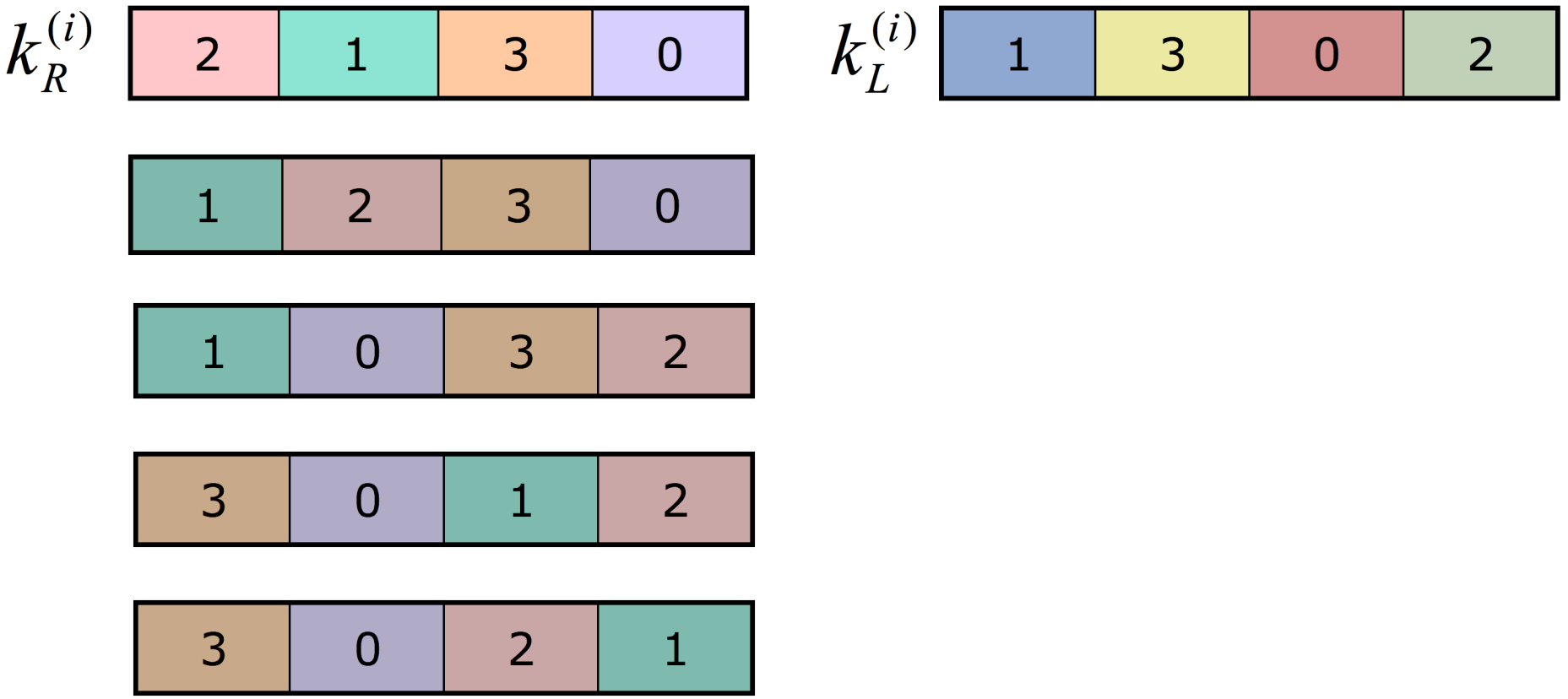
Session Key Example, Original Protocol



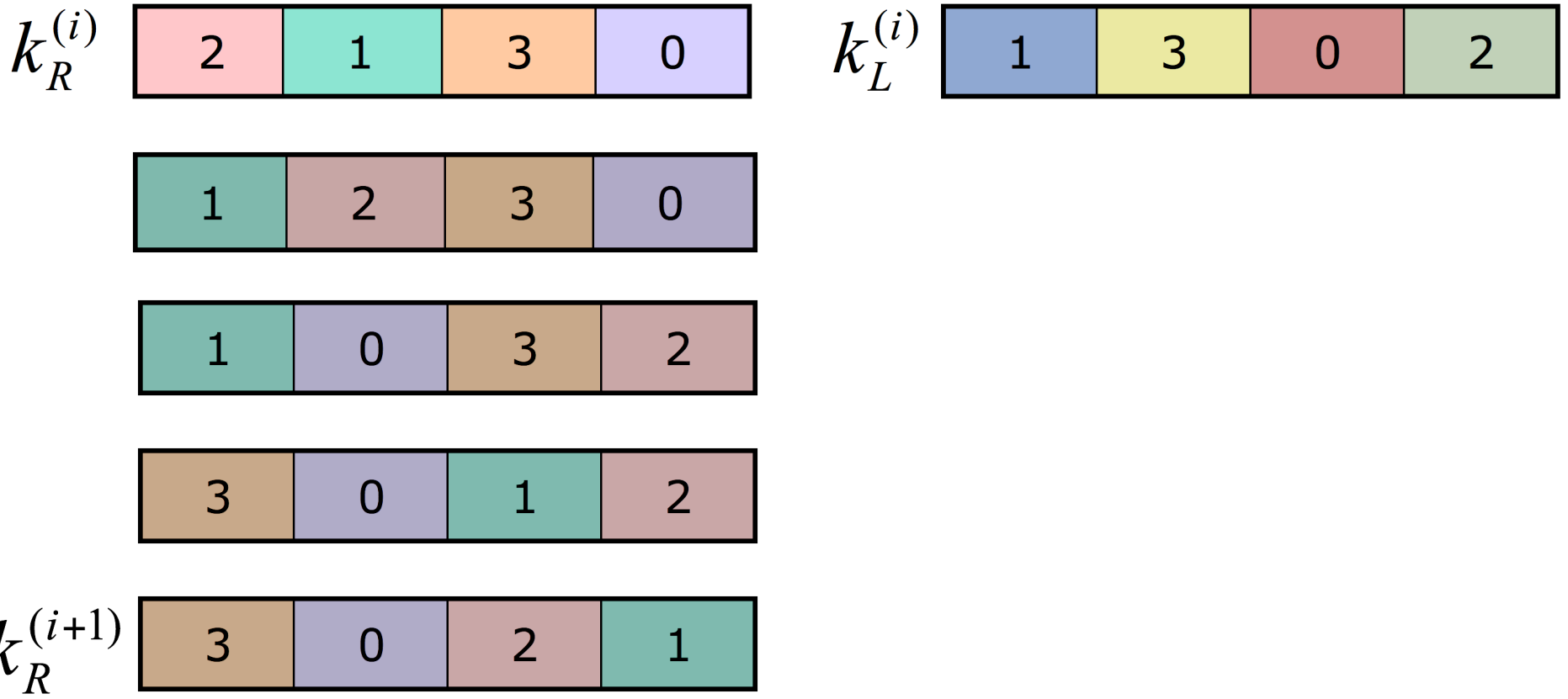
Session Key Example, Original Protocol



Session Key Example, Original Protocol



Session Key Example, Original Protocol

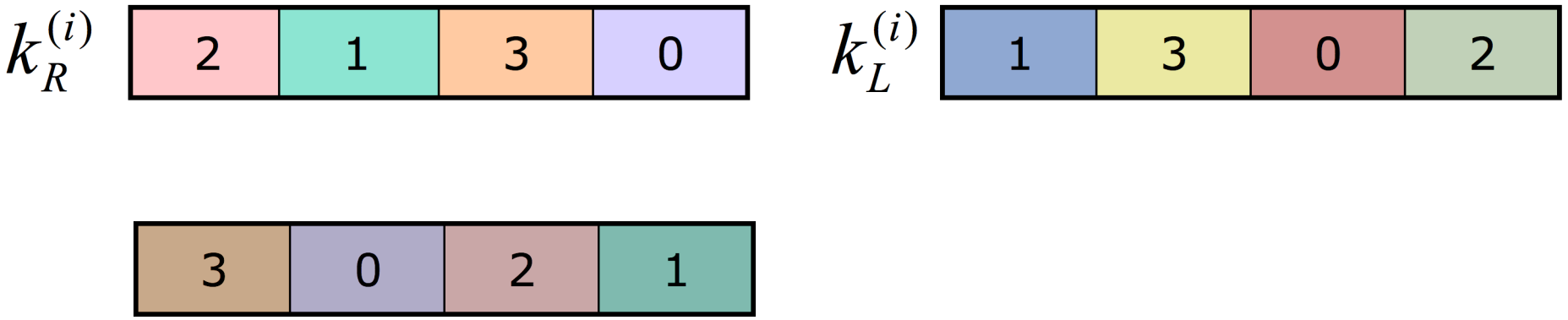


Session Key Example, Original Protocol

- Observe that sometimes a nibble "does not move"

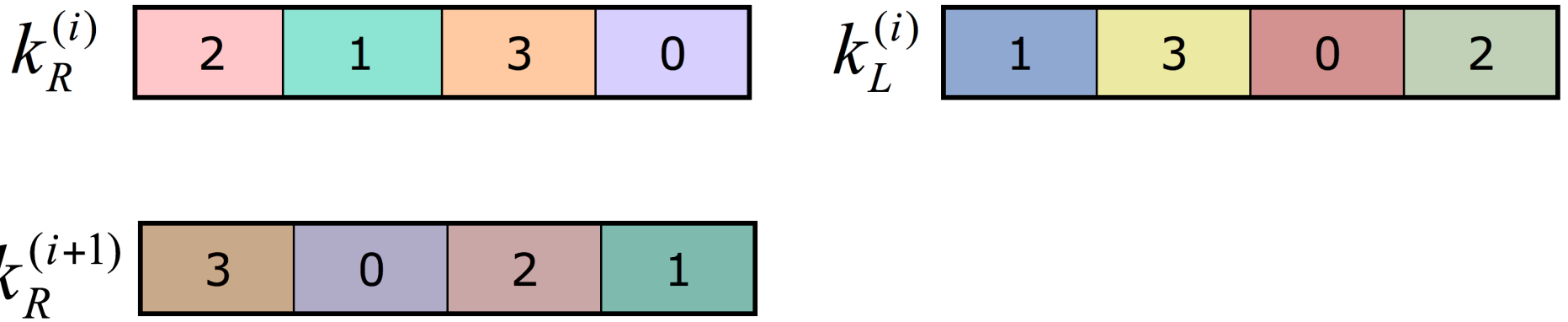
Session Key Example, Original Protocol

- Observe that sometimes a nibble "does not move"



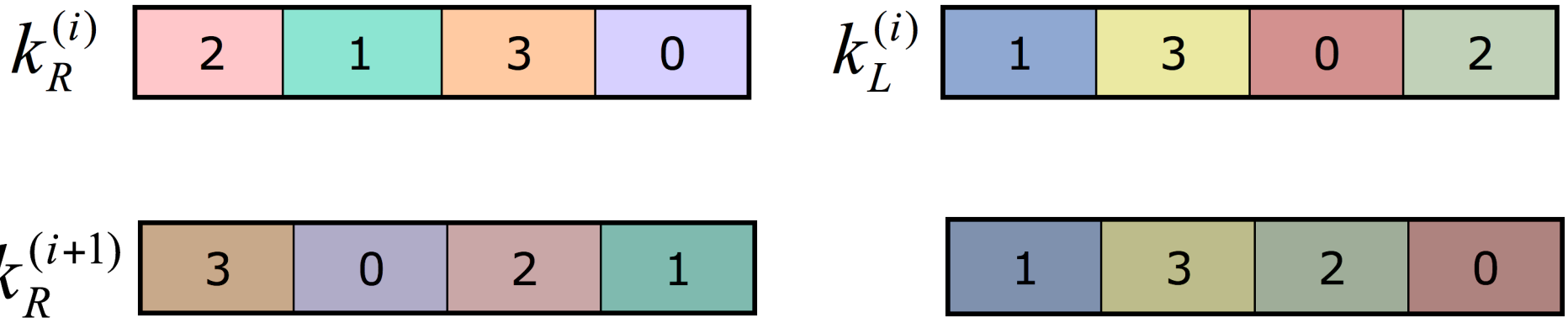
Session Key Example, Original Protocol

- Observe that sometimes a nibble "does not move"



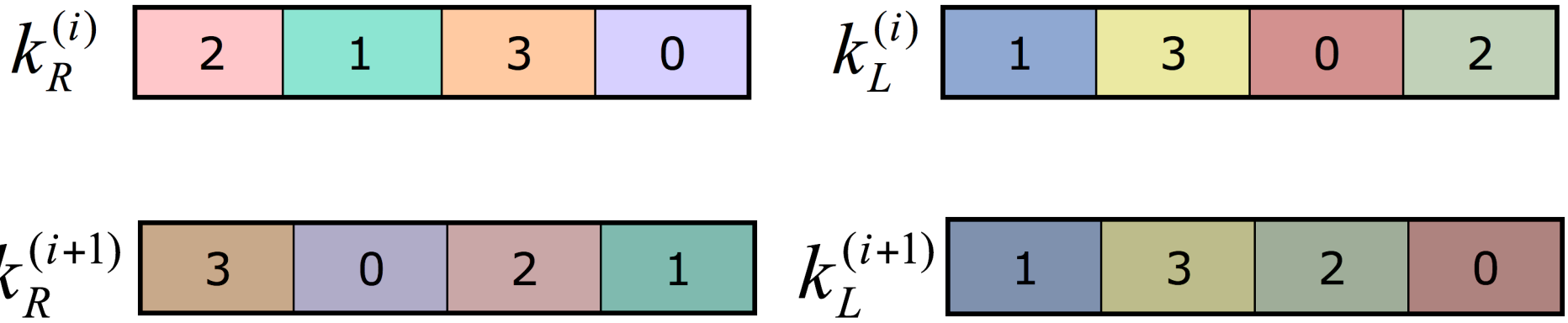
Session Key Example, Original Protocol

- Observe that sometimes a nibble "does not move"



Session Key Example, Original Protocol

- Observe that sometimes a nibble "does not move"



Nibble Attack

- Exploits deficiencies of session key function
  - Determine which nibbles "don't move"
  - Construct table of observed values
  - Can determine one nibble of shared secret
- Can determine full shared secret with high probability after expected 1,092 transactions
- Active or passive attack

Outline

- Introduction to RFID
- Original low-cost RFID authentication protocol
- Implementation Results
- Repeated Keys Attack
- Nibble Attack
- Suggestions for future protocols

Conclusions & Suggestions for future protocols

- Reader *cannot prevent or detect* the attacks
  - Attack 1: impersonate a tag after an average of 70 transactions
  - Attack 2: learn shared secret after an average of 1,092 transactions
- Adversary *never* needs to brute force the key
- Suggestions
  - Importance of implementation
  - Statistical measurements

Questions?

defend@cs.umass.edu
http://www.cs.umass.edu/~defend
http://www.rfid-cusp.org/

Session Key Function $k^{(i+1)} = F(k^{(i)})$

- $k^{(i+1)}$ is formed by moving nibbles of the left and right vectors of $k^{(i)}$
- To form $k_R^{(i+1)}$:

$k_{0,R}^{(i+1)}$ = swap 0^{th} and $k_{0,L}^{(i)}{}^{th}$ elements of $k_R^{(i)}$

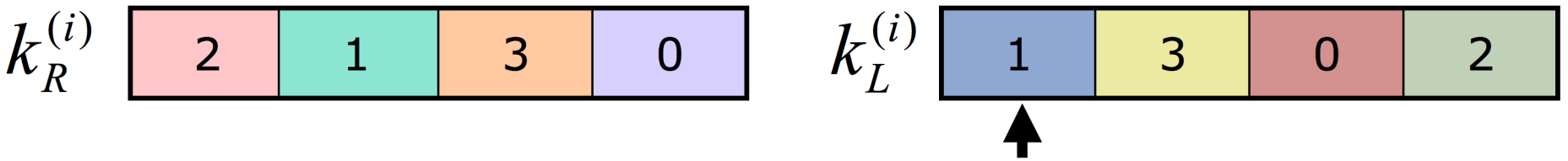
$k_{1,R}^{(i+1)}$ = swap 1^{st} and $k_{1,L}^{(i)}{}^{th}$ elements of $k_R^{(i)}$

$\vdots$

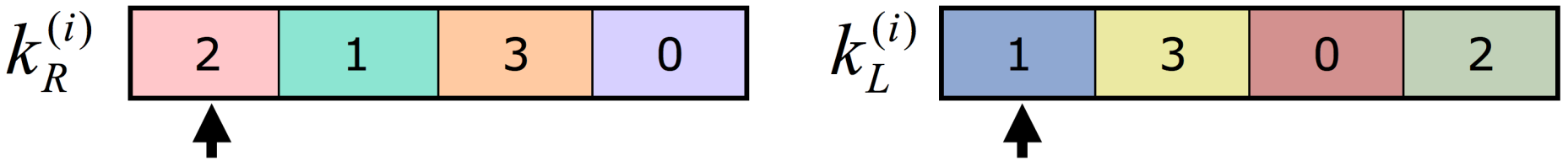
$k_{15,R}^{(i+1)}$ = swap 15^{th} and $k_{15,L}^{(i)}{}^{th}$ elements of $k_R^{(i)}$

Session Key Example

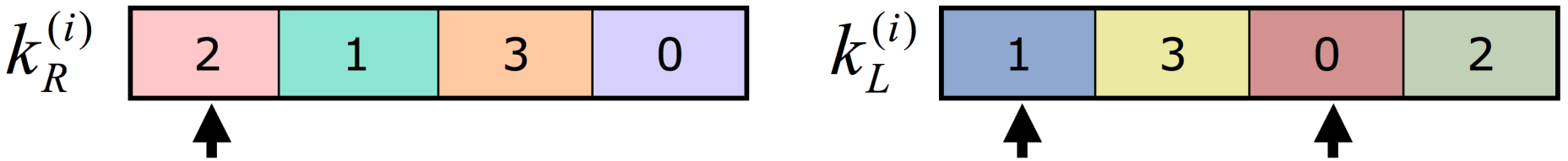
Session Key Example



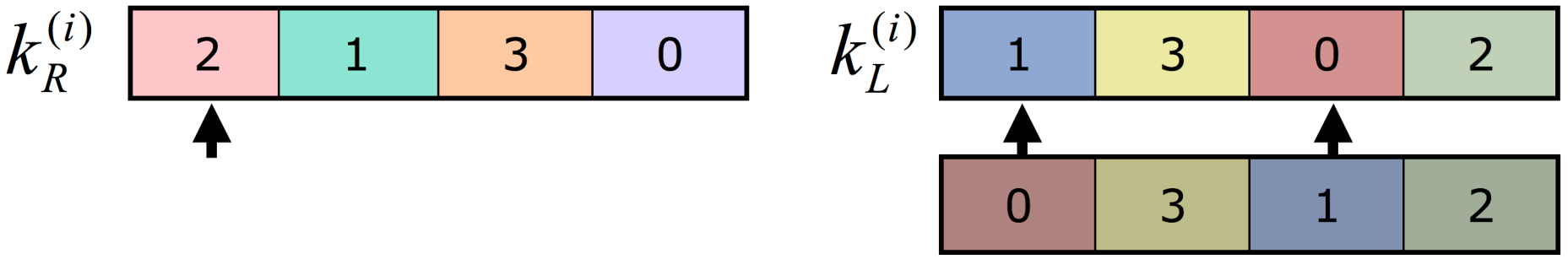
Session Key Example



Session Key Example



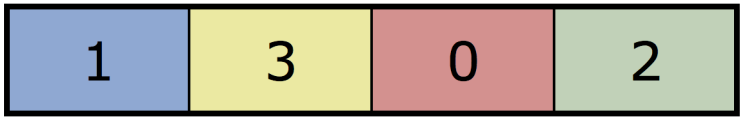
Session Key Example



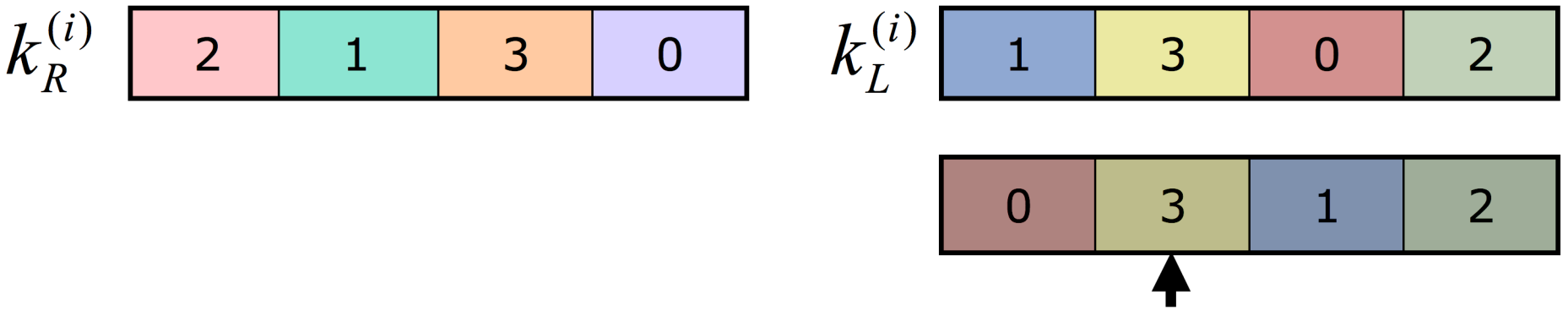
Session Key Example

$k_R^{(i)}$

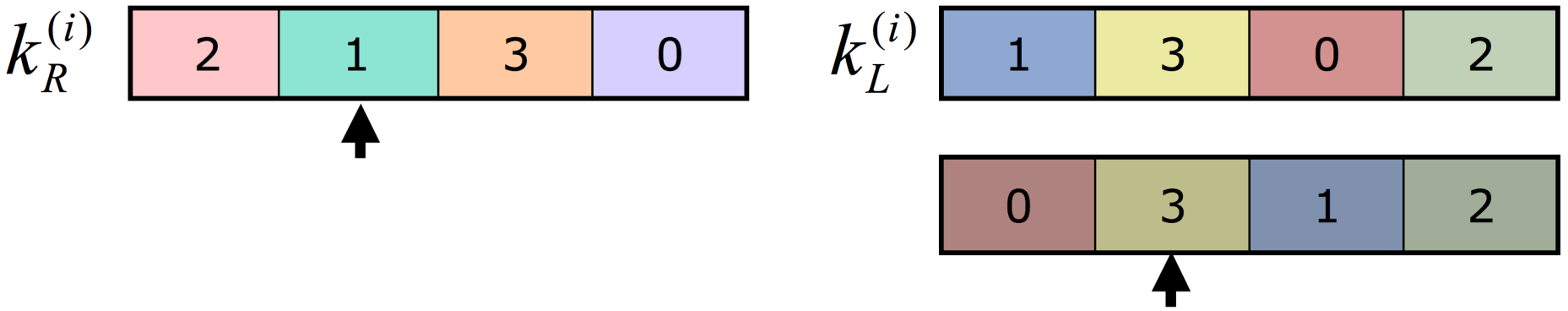
$k_L^{(i)}$



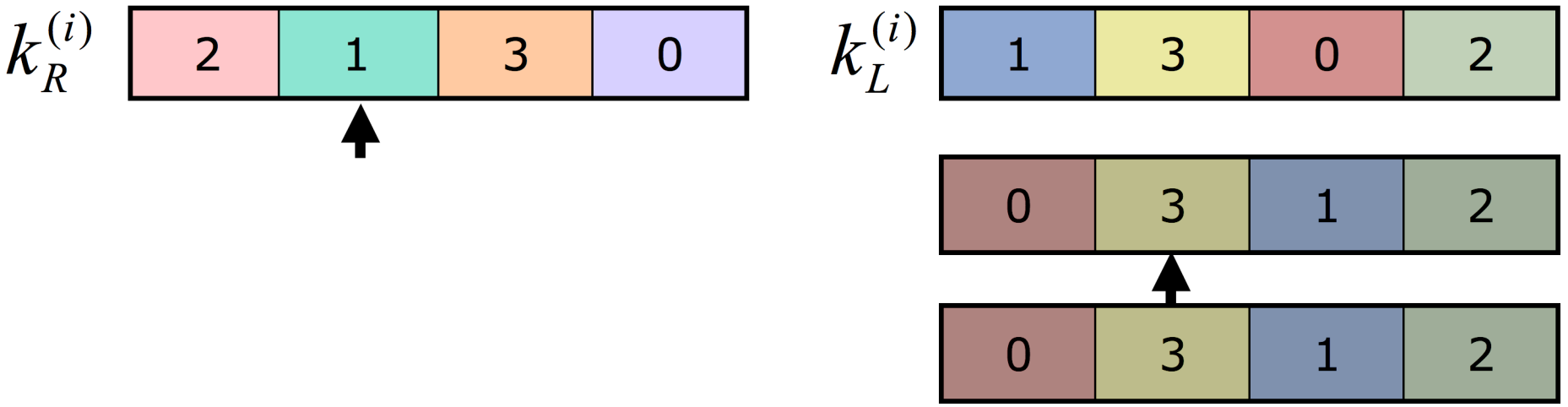
Session Key Example



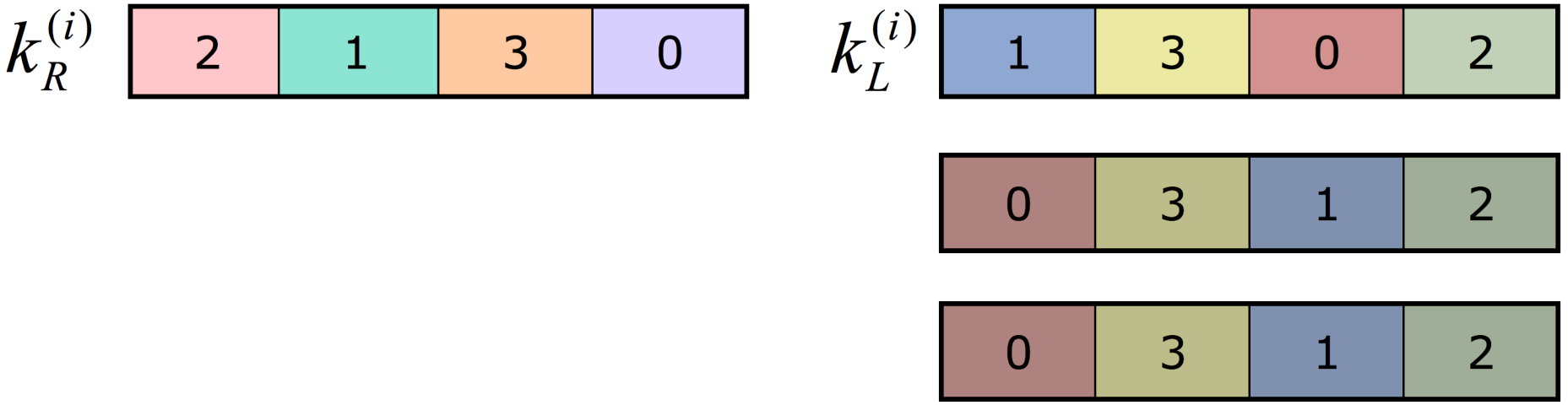
Session Key Example



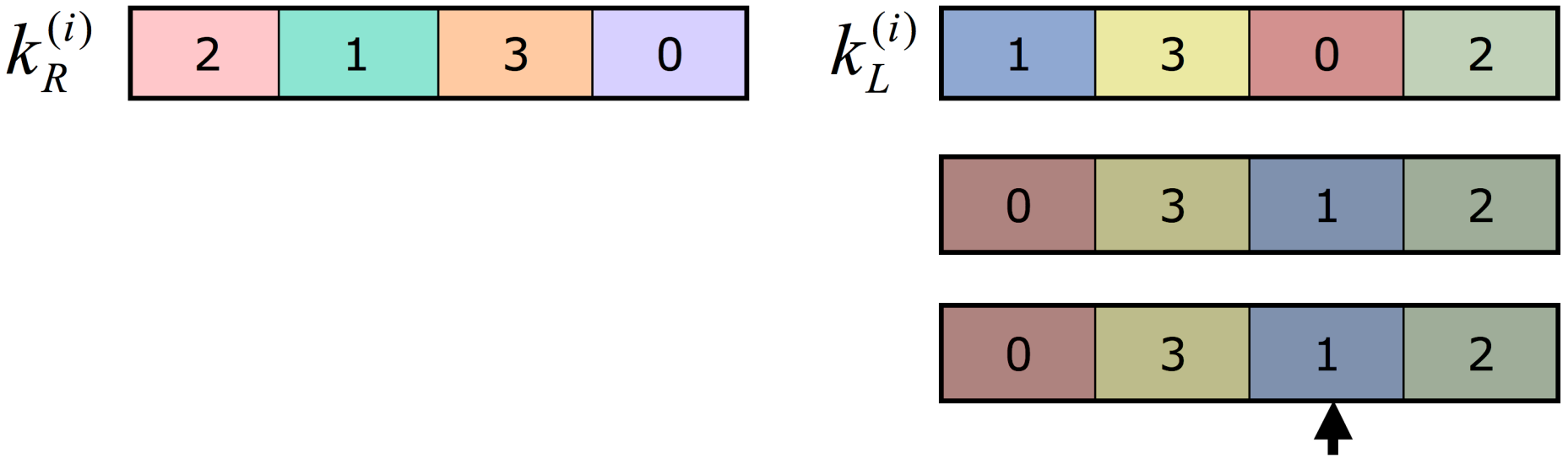
Session Key Example



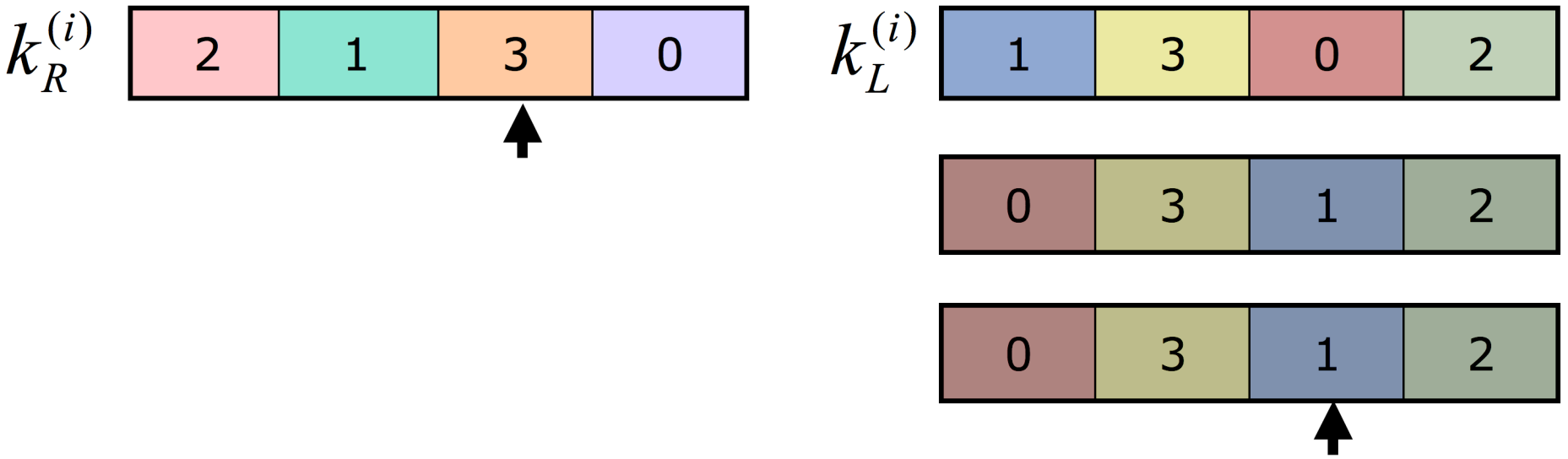
Session Key Example



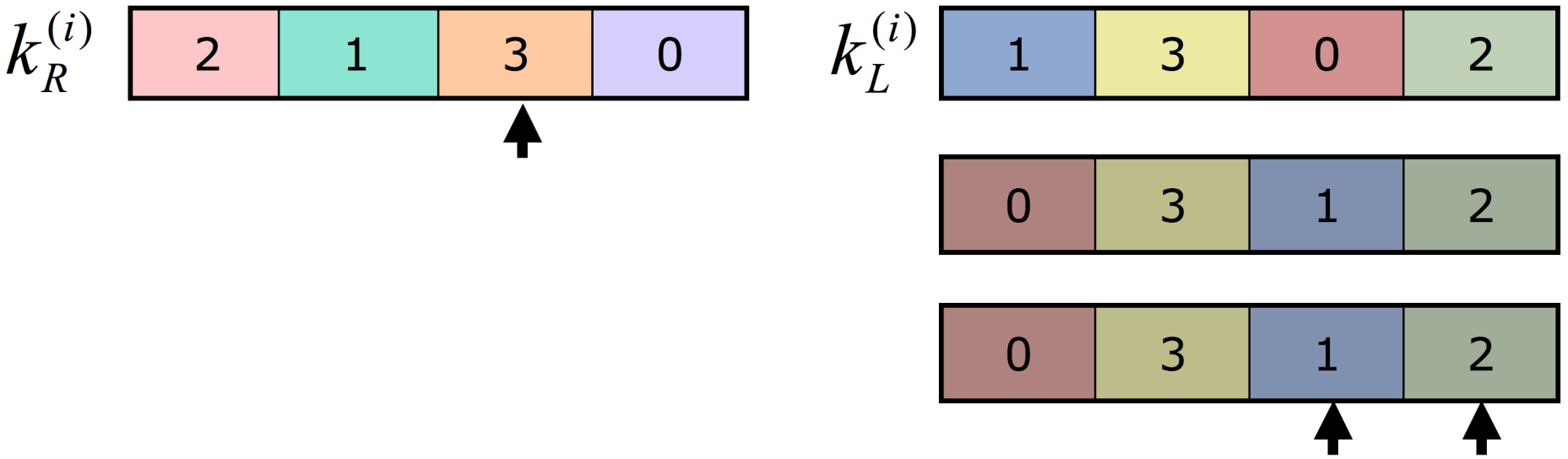
Session Key Example



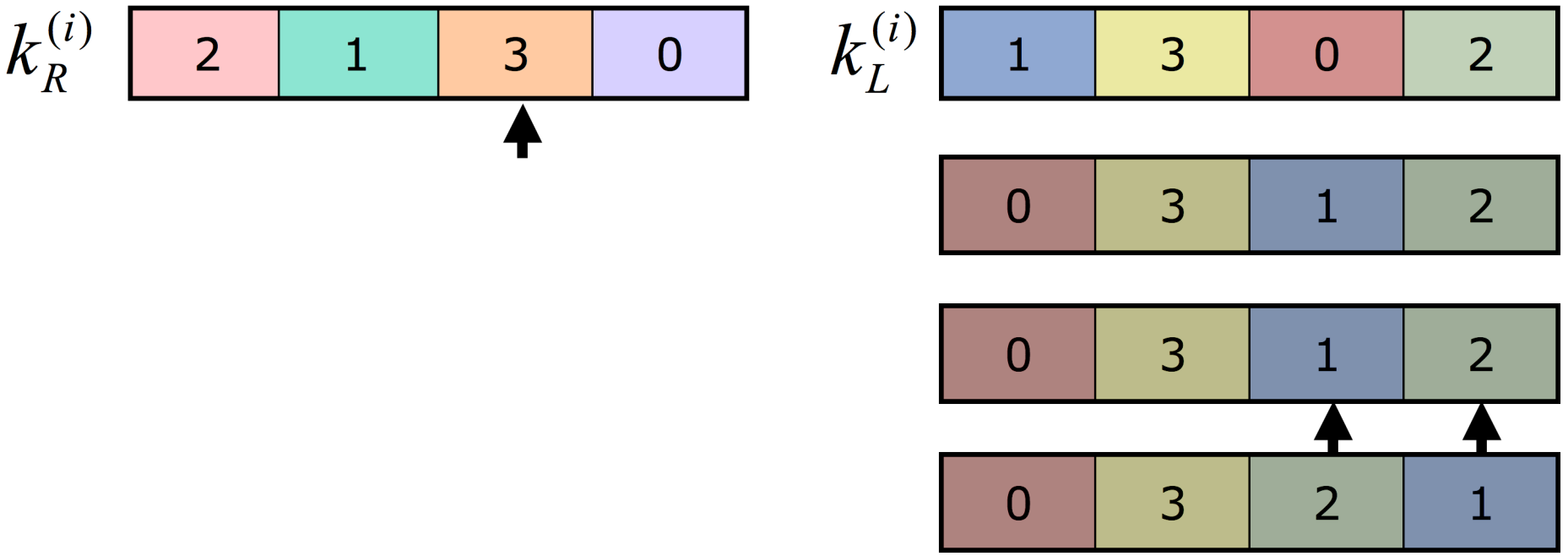
Session Key Example



Session Key Example



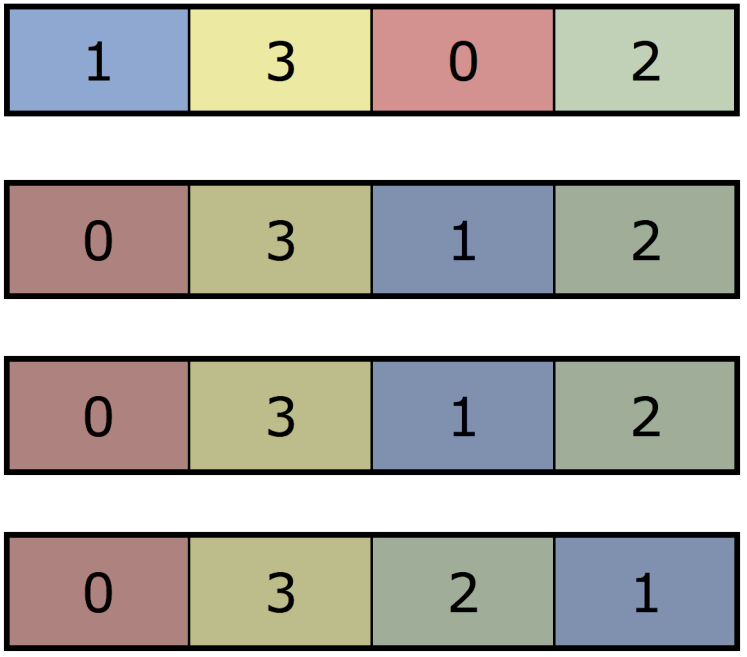
Session Key Example



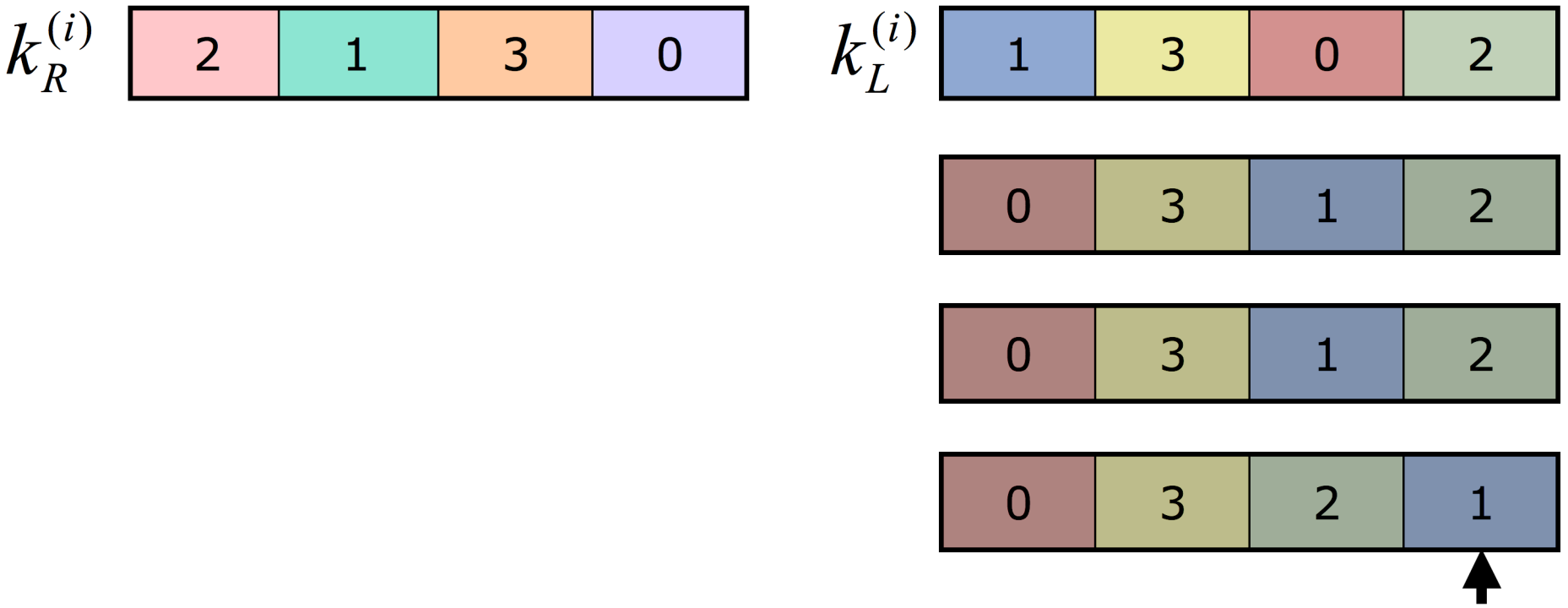
Session Key Example

$k_R^{(i)}$

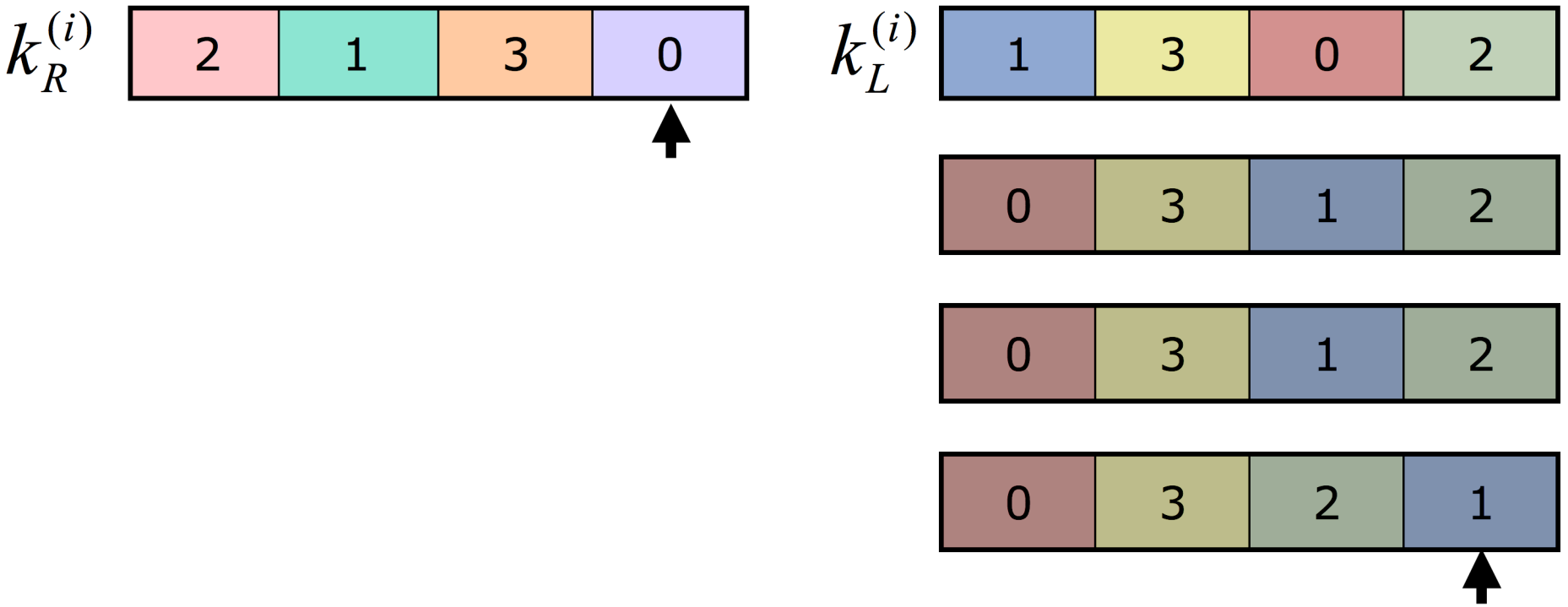
$k_L^{(i)}$



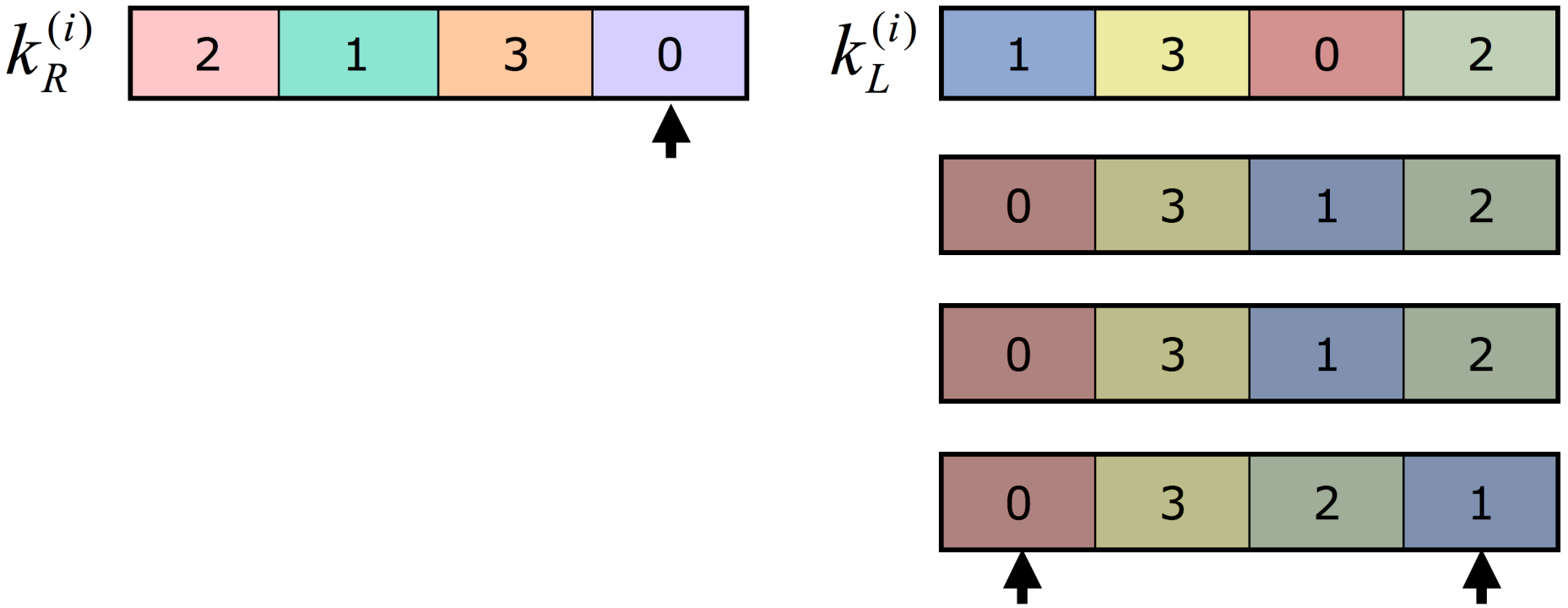
Session Key Example



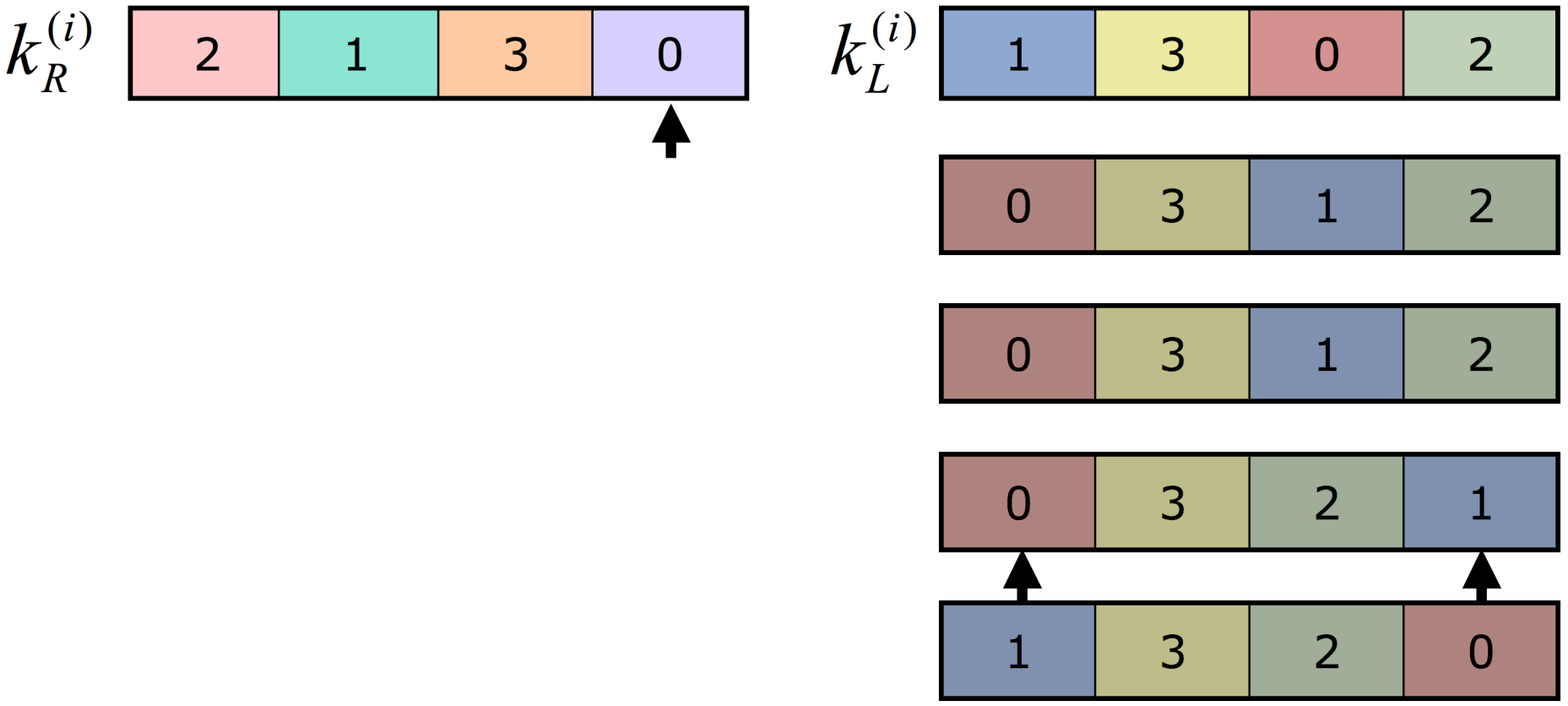
Session Key Example



Session Key Example



Session Key Example



Session Key Example