Verification of Security Protocols

References

- J. C. Mitchell, V. Shmatikov, and U. Stern. Finite-state analysis of SSL 3.0, *USENIX*, 1998.
- F. J.Thayer Fábrega, J. C. Herzog, J. D. Guttman. Strand spaces: proving security protocols correct, *Journal of Computer Security*, 1999.
- D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries, *TCC*, LNCS2951, 2004.
- *Foundations of Cryptography, Volume I Basic Tools*, Goldreich, Cambridge, 2001
- *Foundations of Cryptography, Volume II Basic Applications*, Goldreich, Cambridge, 2004

Two Approaches to Protocol Verification

- Symbolic approach
  - Assumes perfect cryptography
  - Protocols are modeled as concurrent systems
  - Various formal methods can be applied
- Computational approach
  - Allows realistic (vulnerable) cryptography
  - Taken by cryptographers
  - Accepted as standard for guaranteeing security
- Efforts to imply the latter by the former

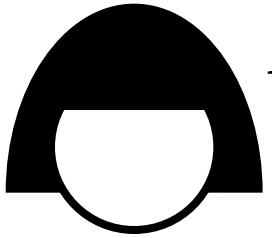
Symbolic Approach

- Assumes perfectly secure cryptography
  - Dolev-Yao model
    - Messages are represented as symbolic terms
- Formal methods are applied
  - Modeling
    - State transition systems
    - Process calculi (general or special-purpose)
    - Strand space
  - Automatic verification
    - Model checkers
    - Special-purpose verifiers

Example of Symbolic Approach

- The main part of the Needham-Schroeder authentication protocol by public cryptography (1978) in Alice-Bob notation

$$A \rightarrow B : \{A, N_A\}_{KB}$$
$$B \rightarrow A : \{N_A, N_B\}_{KA}$$
$$A \rightarrow B : \{N_B\}_{KB}$$



Alice

random number or
nonce (number once)
(secret known only to the creator)

$\{A, N_A\}_{KB}$

{Alice, Alice's secret}$_{\text{Bob's public key}}$

At this point, Alice's secret is known only to Alice and Bob

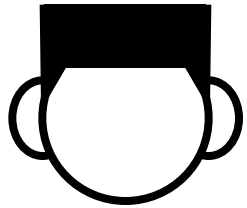
$\{N_A, N_B\}_{KA}$

{Alice's secret, Bob's secret}$_{\text{Alice's public key}}$

Receiving Alice's secret implies that Bob has received the first message

$\{N_B\}_{KB}$

Bob



$\{A, N_A\}_{KB}$

{Alice, Alice's secret}$_{\text{Bob's public key}}$

$\{N_A, N_B\}_{KA}$

{Alice's secret, Bob's secret}$_{\text{Alice's public key}}$

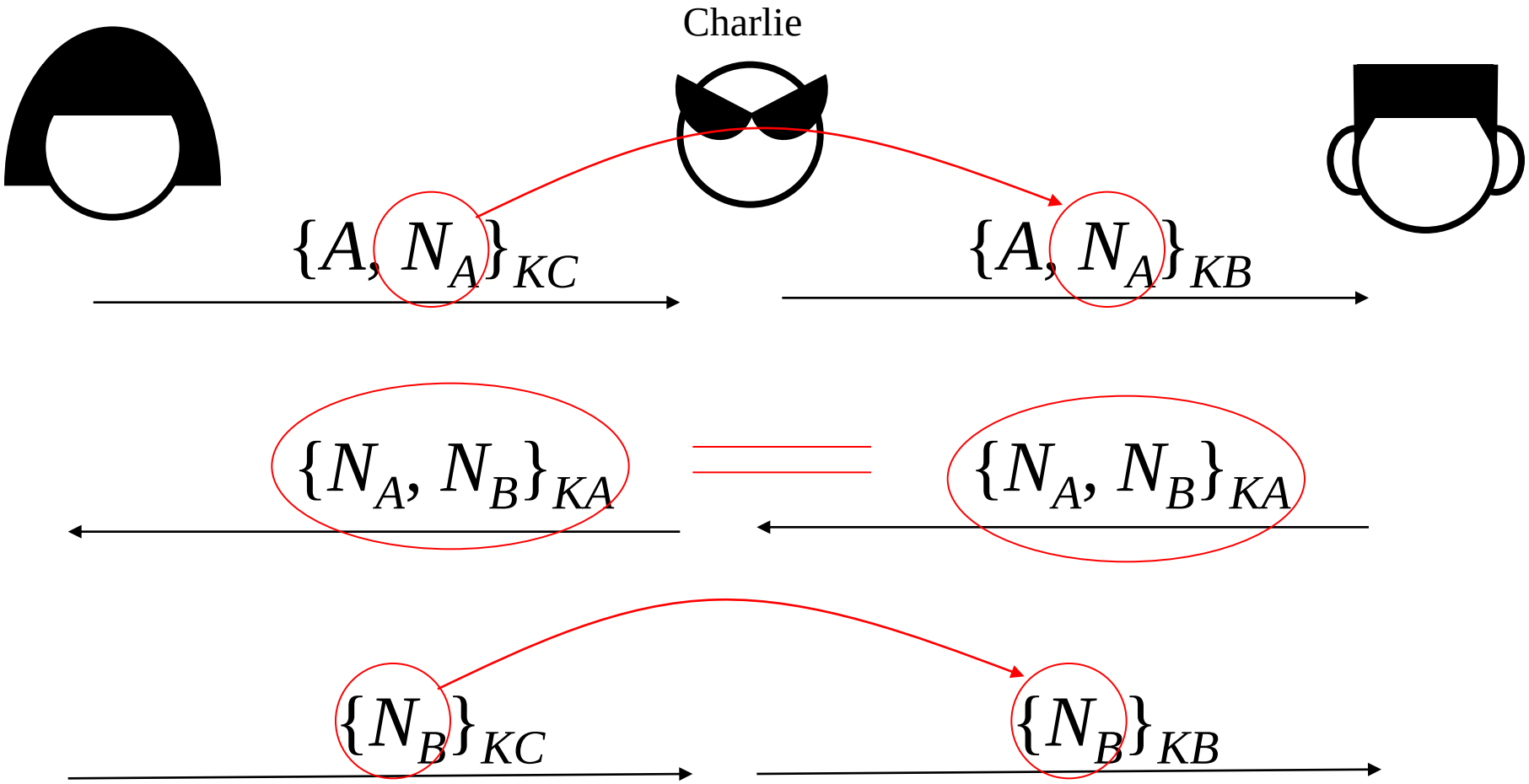
At this point, Bob's secret is known only to Alice and Bob

$\{N_B\}_{KB}$

Ditto

Man-in-the-Middle Attack

- Discovered by Lowe in 1995
  - Nearly 20 years after publication
  - Partly due to formal modeling of the protocol



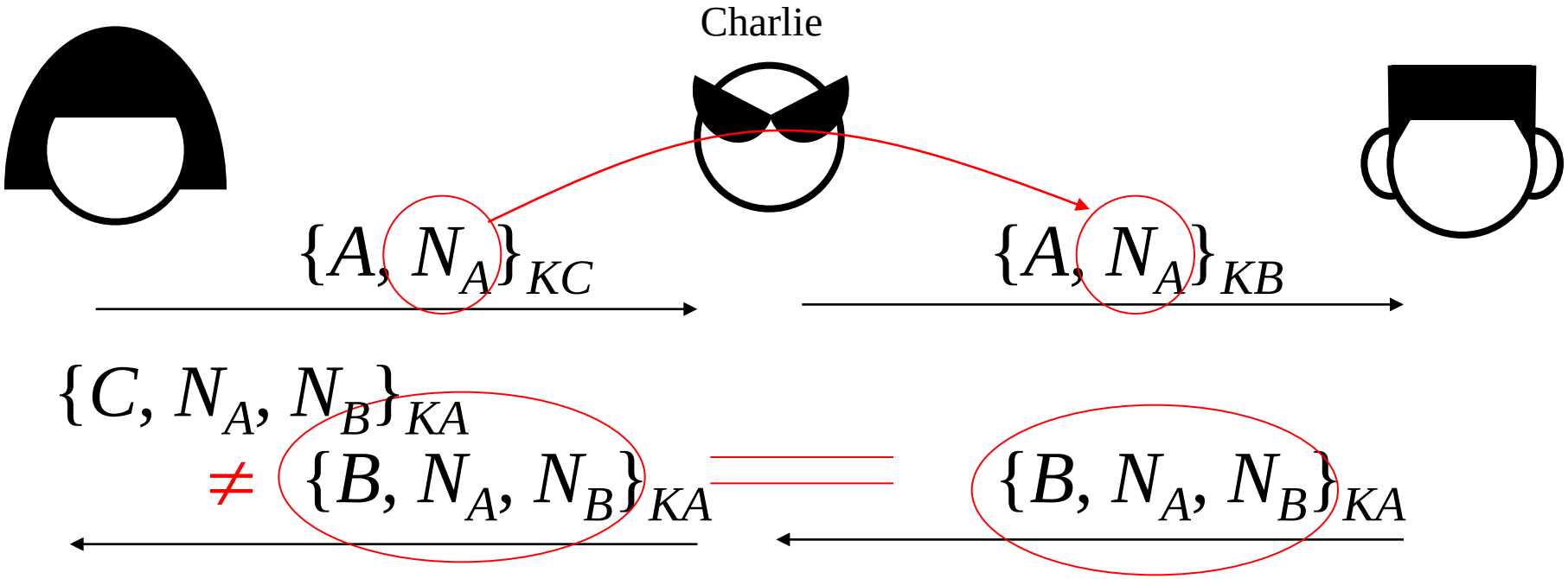
Corrected Protocol

- Needham-Schroeder-Lowe

$$A \rightarrow B : \{A, N_A\}_{KB}$$
$$B \rightarrow A : \{B, N_A, N_B\}_{KA}$$
$$A \rightarrow B : \{N_B\}_{KB}$$

Man-in-the-Middle Attack

- Discovered by Lowe in 1995
  - Nearly 20 years after publication
  - Partly due to formal modeling of the protocol



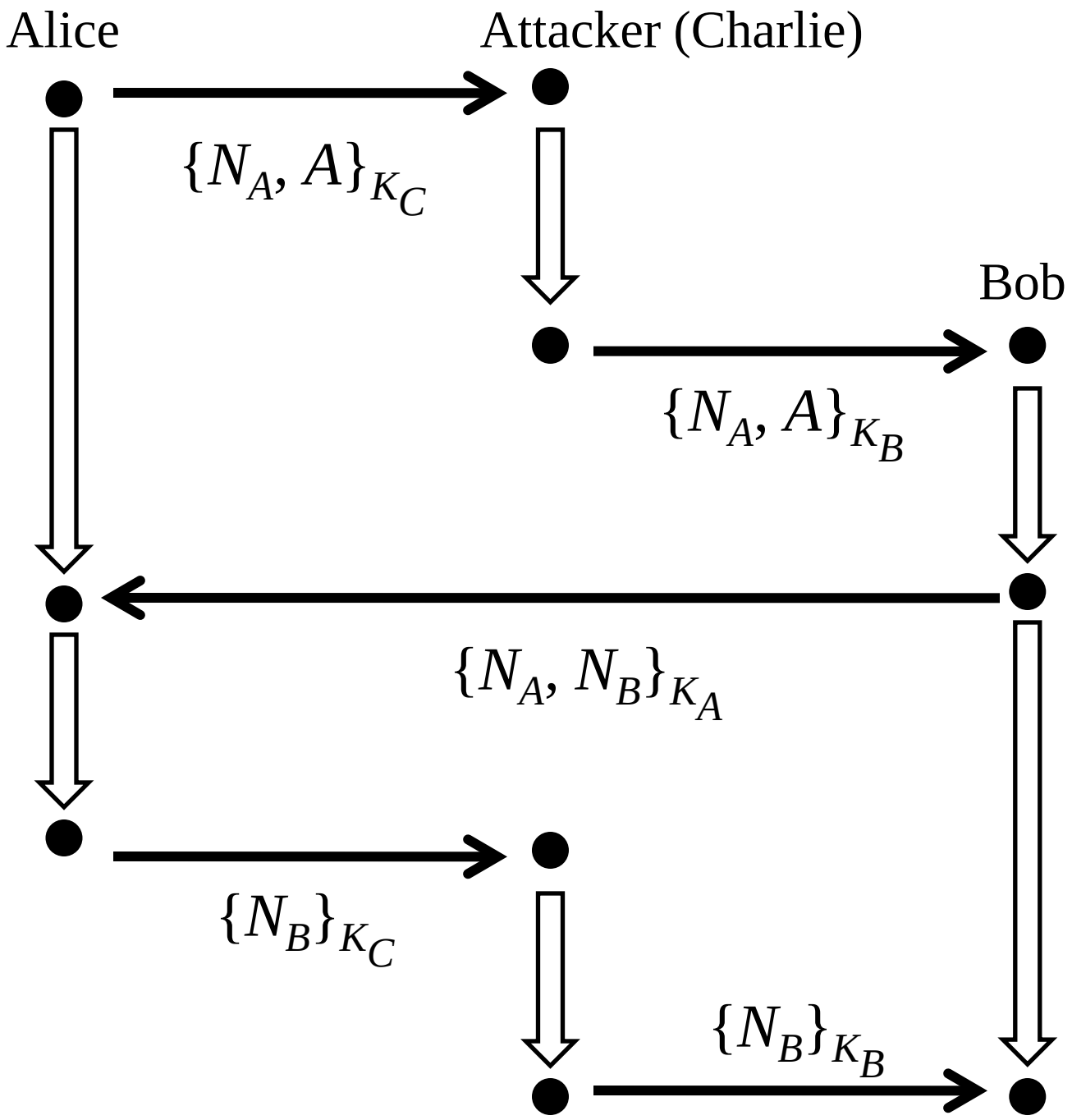
Model Checking

- Protocols are typical concurrent systems
- Model checking can be used to find bugs
  - But only applicable to finite instances

- Report for Part III

Strand Space Model

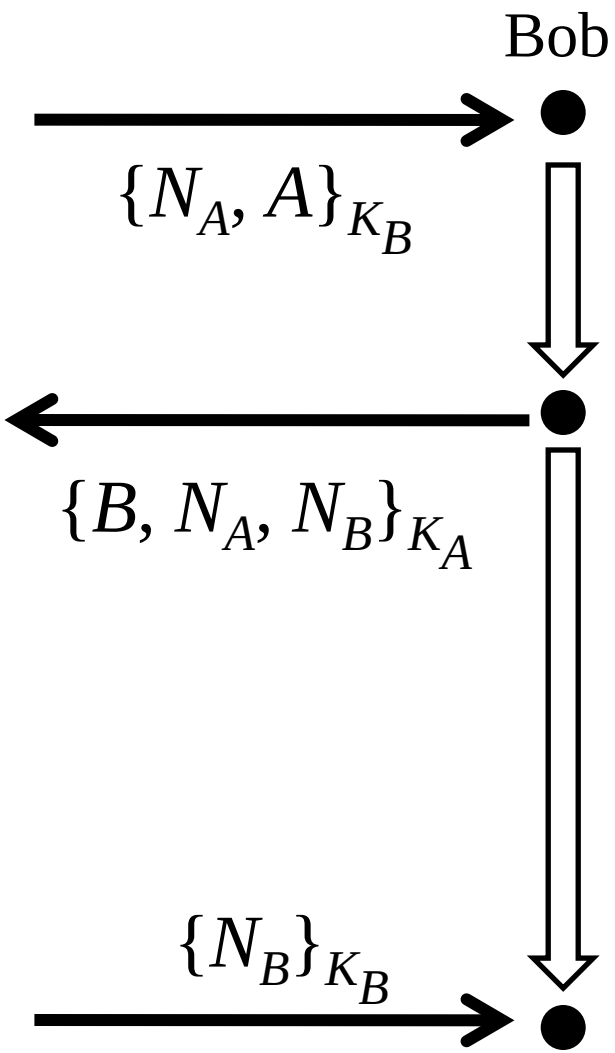
- Developed by Guttman, *et al.* (1998)
  - Strand --- execution trace by one principal
  - Bundle --- set of strands closed by causal relationship
- Enumeration of bundles by backward reasoning
- Attacker's strands cover all possible attacks
- Allows any number of principals (sessions)

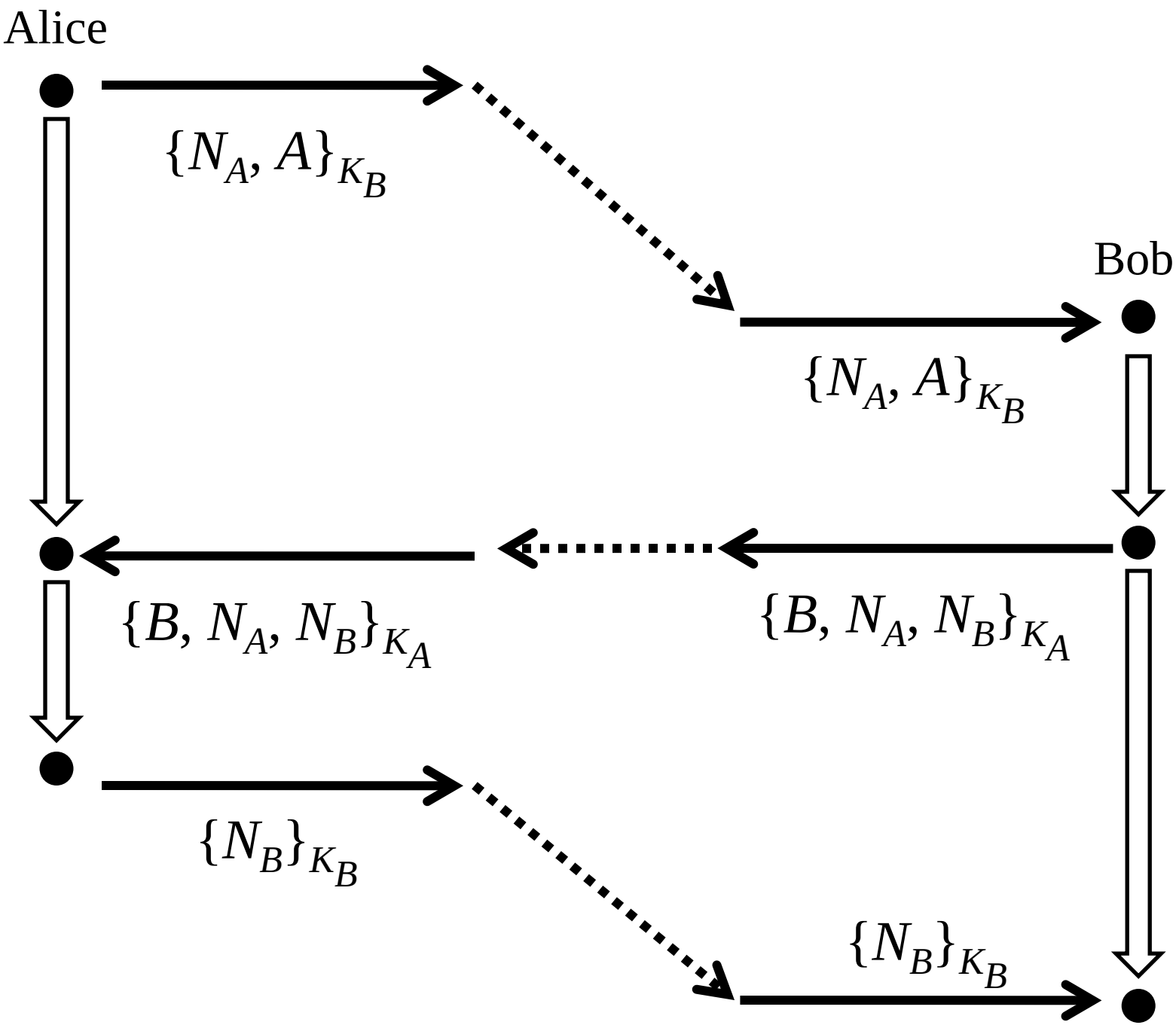

Alice
Attacker (Charlie)
Bob
$\{N_A, A\}_{K_C}$
$\{N_A, A\}_{K_B}$
$\{N_A, N_B\}_{K_A}$
$\{N_B\}_{K_C}$
$\{N_B\}_{K_B}$

Verification in Strand Space Model

- Verification of agreement

  Assuming a strand of a certain principal, enumerate all possible bundles that may contain the strand, and show that each such bundle contains a strand of the corresponding principal
- Authenticity
- Mutual authenticity

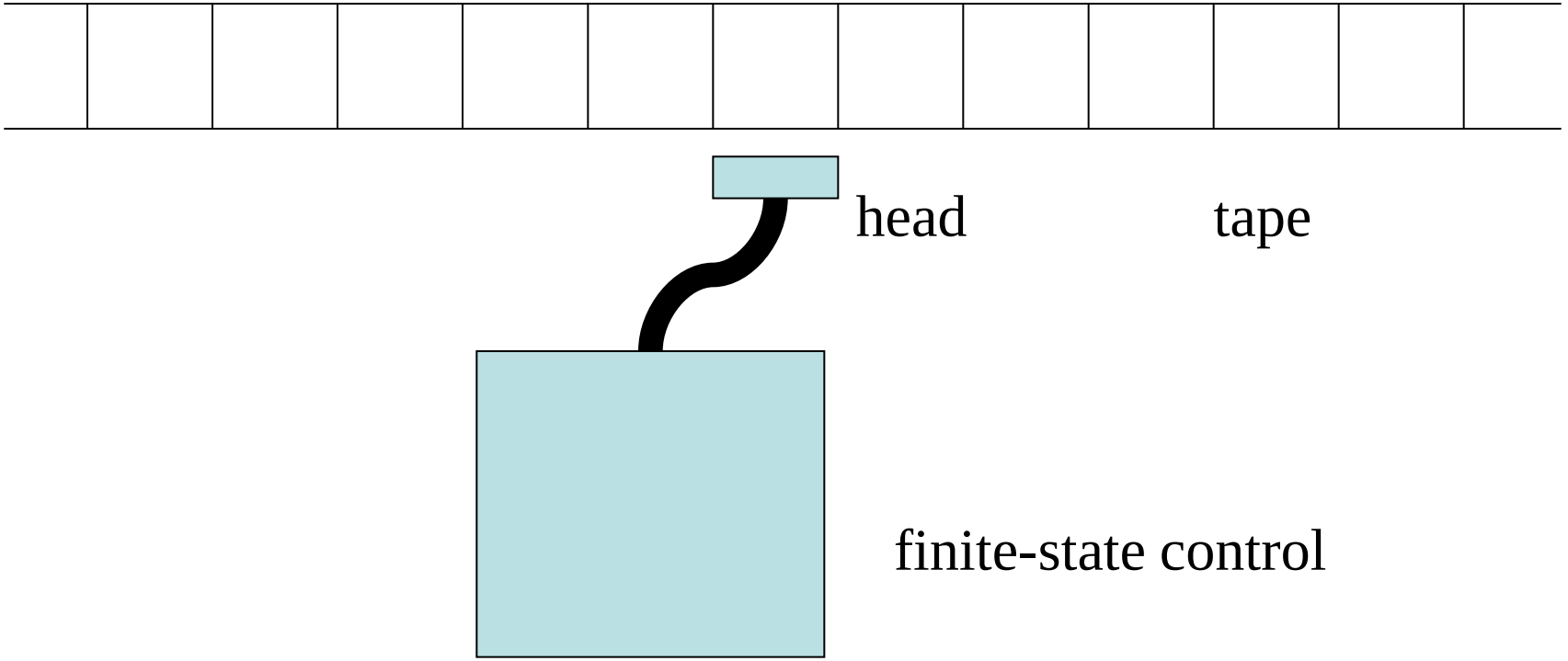

Bob
$\{N_A, A\}_{K_B}$
$\{B, N_A, N_B\}_{K_A}$
$\{N_B\}_{K_B}$


Alice
Bob
$\{N_A, A\}_{K_B}$
$\{N_A, A\}_{K_B}$
$\{B, N_A, N_B\}_{K_A}$
$\{B, N_A, N_B\}_{K_A}$
$\{N_B\}_{K_B}$
$\{N_B\}_{K_B}$

Computational Approach

- Assumes vulnerability of cryptography
  - IND-CPA
  - IND-CCA
- Based on probabilistic polynomial-time Turing machines
  - Modeling the attacker
  - Allows realistic attacks (e.g., guessing attack)
- Complicated and error-prone
  - Difficult to automate
  - Many mistakes in proofs have been reported

Turing Machine



State transition of a Turing machine:

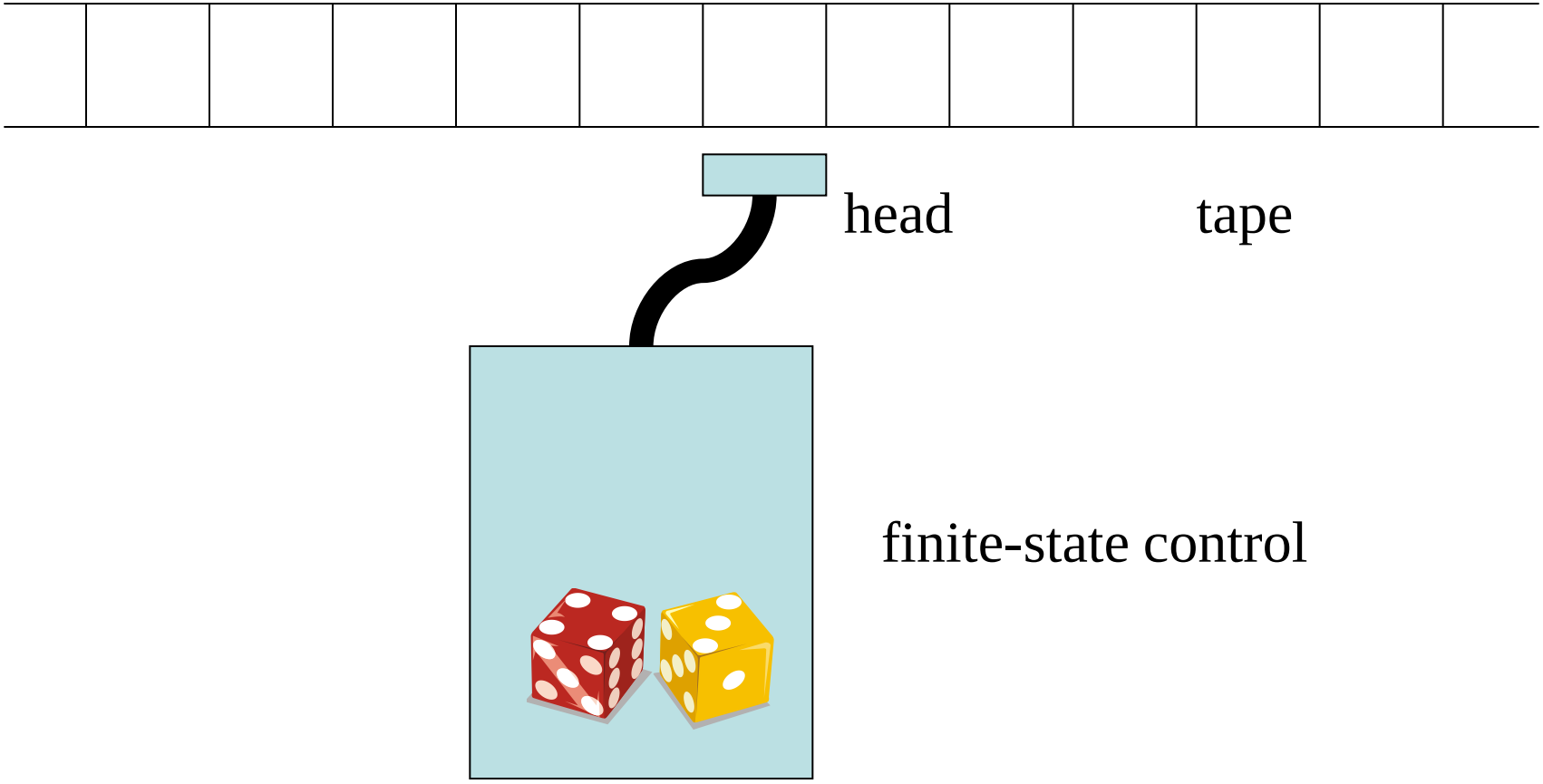
(current state, character read by the head) →

(next state, character written by the head, direction of the head)

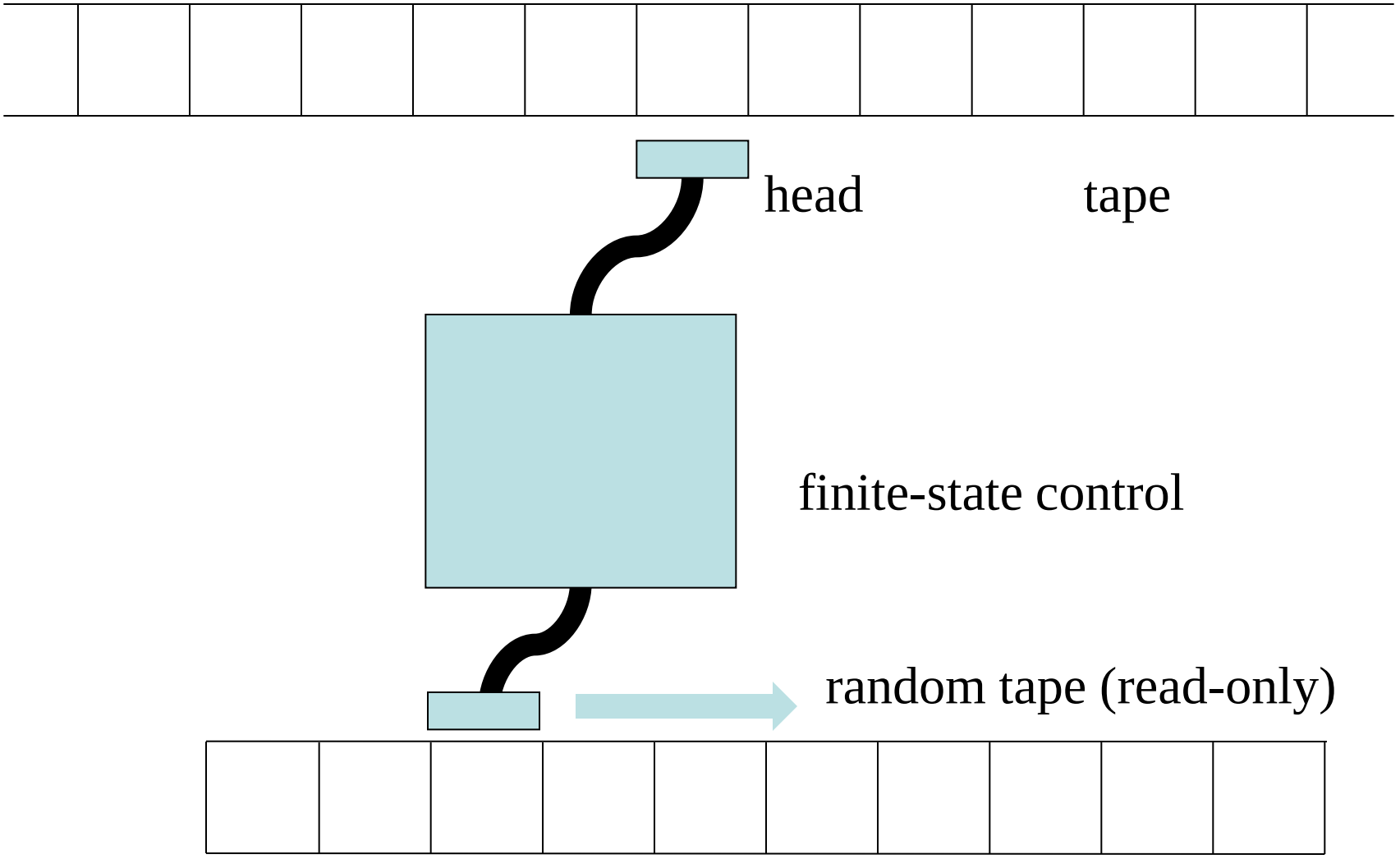
Deterministic vs. Nondeterministic vs. Probabilistic

- Deterministic
  - For each tuple (current state, character read by the head), transition (next state, character written by the head, direction of the head) is unique
- Nondeterministic
  - Transition is not always unique
- Probabilistic
  - Among possible choices, transition is determined probabilistically (by dice)

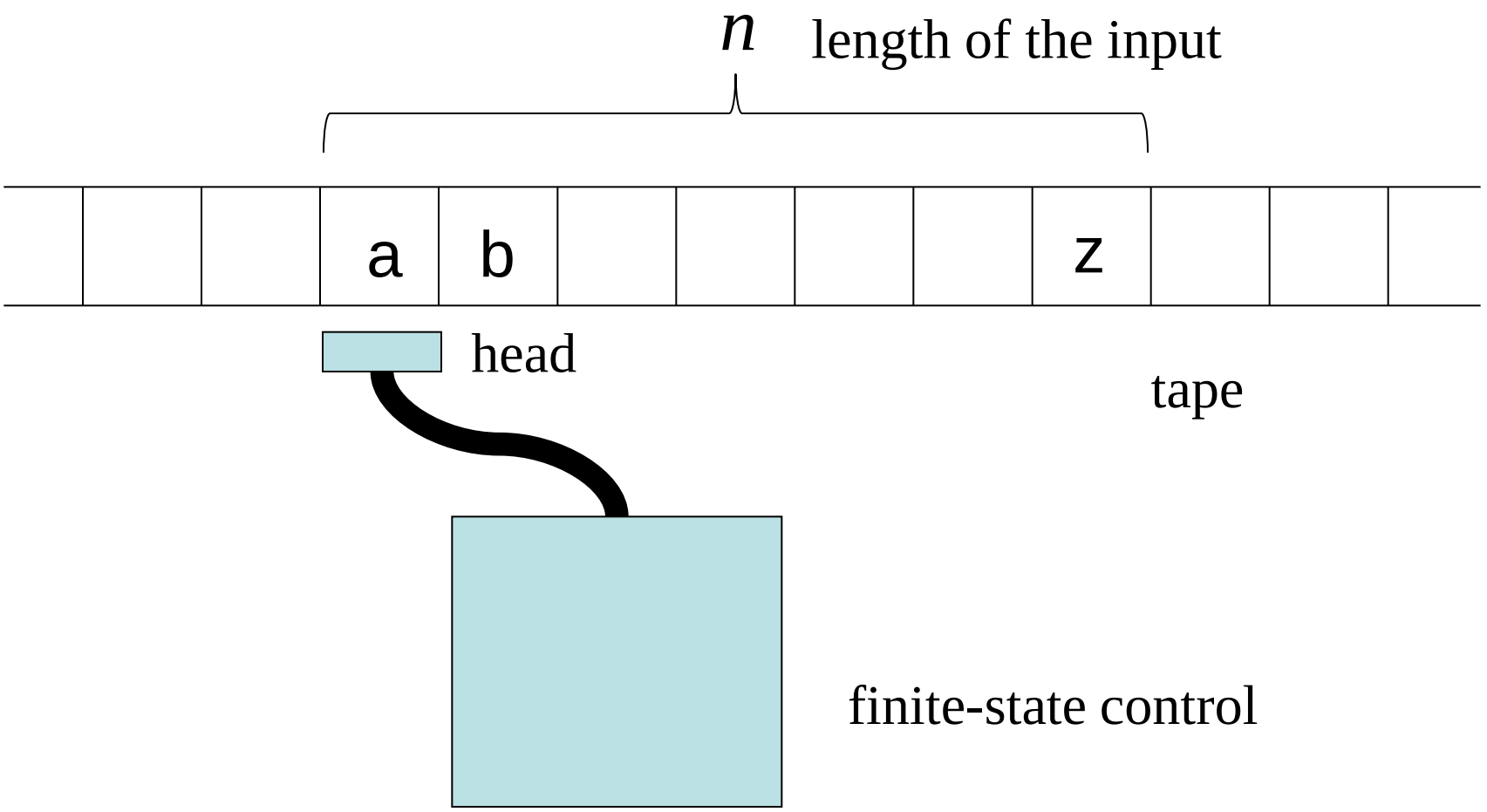
Probabilistic Turing Machine



Probabilistic Turing Machine

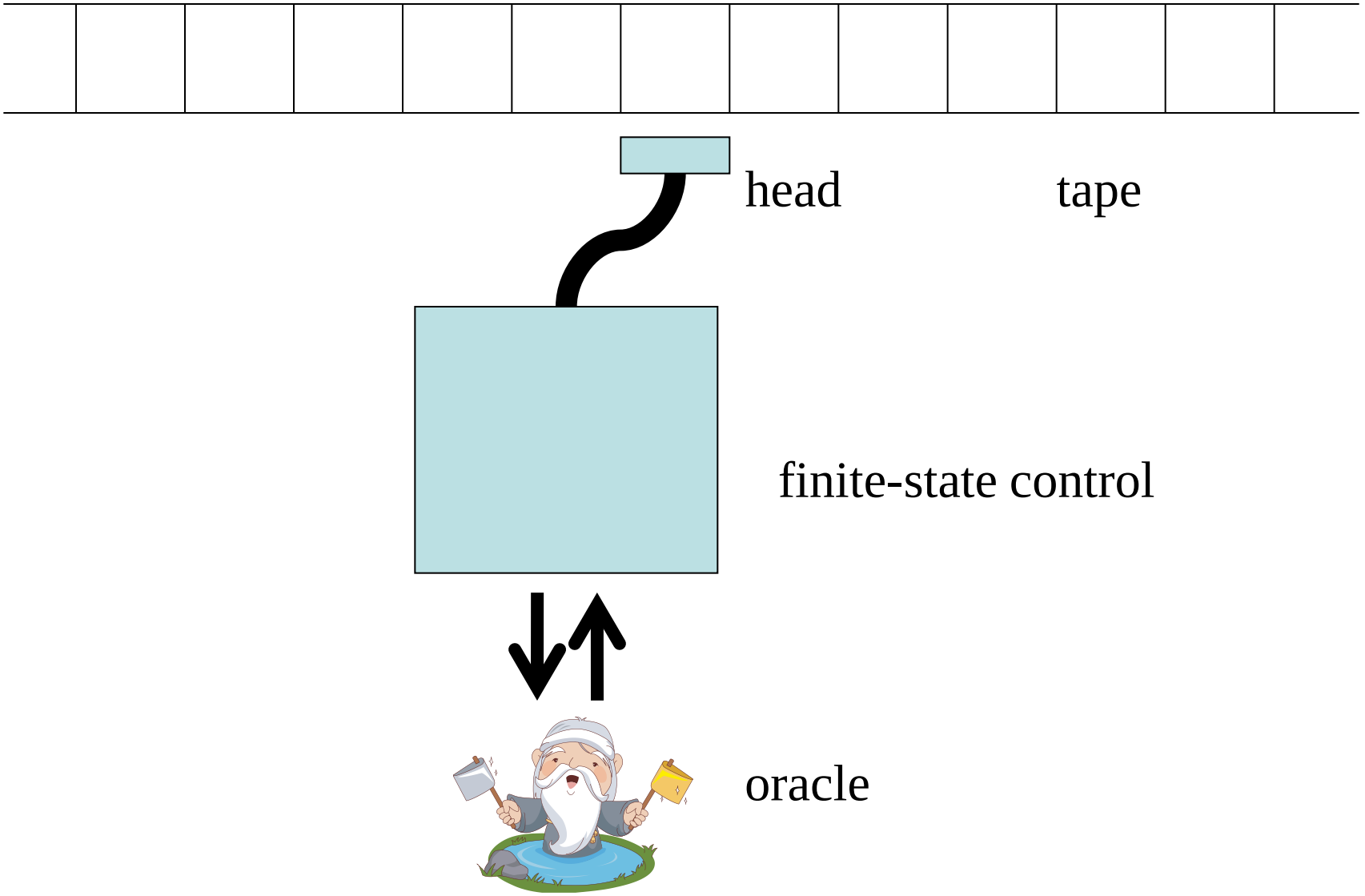


Polynomial-time Turing Machine



Stops in $p(n)$ steps --- $p(n)$ is a polynomal of n

Turing Machine with an Oracle



IND-CPA Game

Attacker

Dealer

Generate public key pk and secret key sk

pk

Generate two messages m_0 and m_1

m_0, m_1

Randomly choose 0 or 1 as b and encrypt m_b by pk as ψ

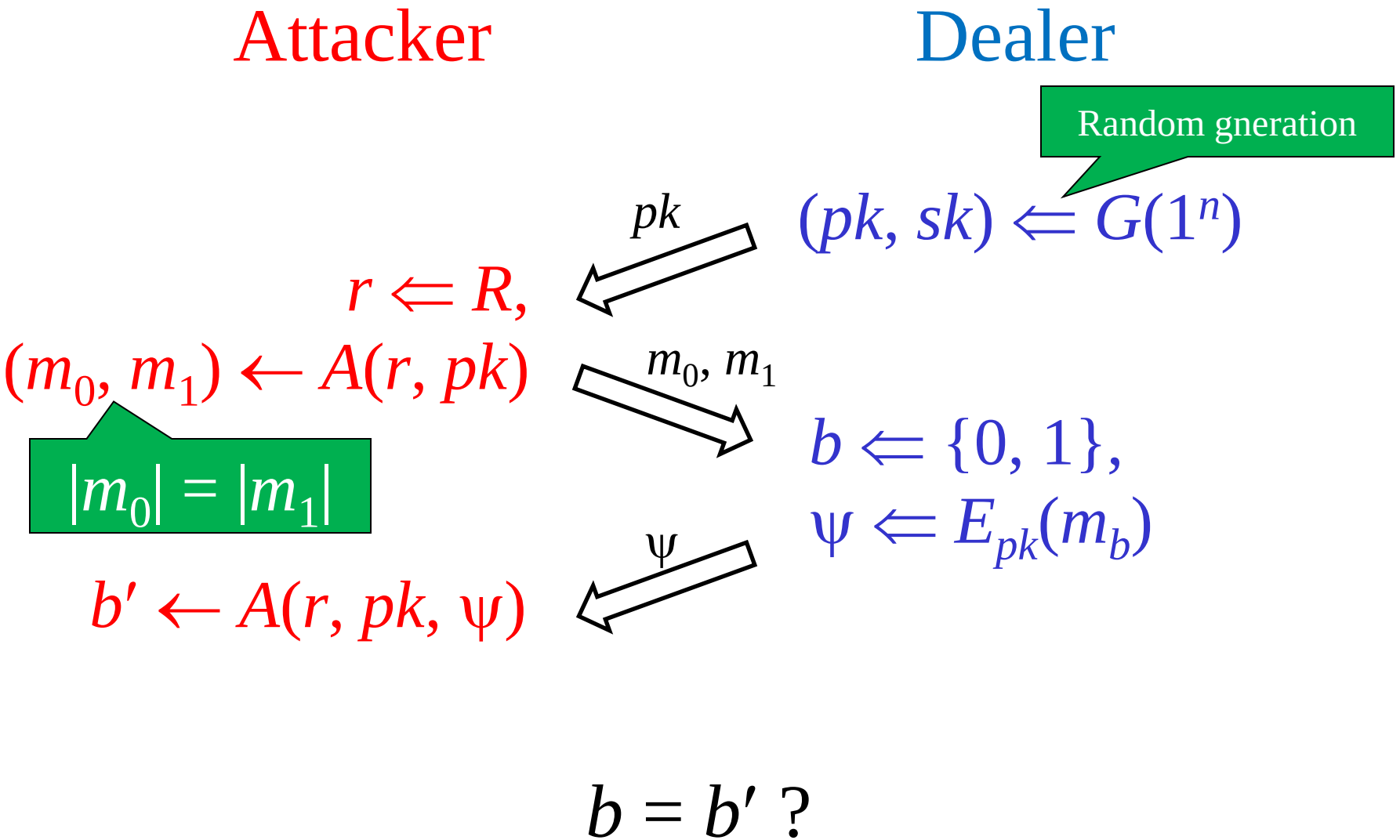
$|m_0| = |m_1|$

ψ

Guess b as b'

$$b = b' \ ?$$

IND-CPA Game


Attacker
Dealer
Random gneration
$(pk, sk) \Leftarrow G(1^n)$
pk
$r \Leftarrow R,$
$(m_0, m_1) \leftarrow A(r, pk)$
m_0, m_1
$|m_0| = |m_1|$
$b \Leftarrow \{0, 1\},$
$\psi \Leftarrow E_{pk}(m_b)$
ψ
$b' \leftarrow A(r, pk, \psi)$
$b = b'$?

IND-CPA Game

Attacker

$r \Leftarrow R,$

r is a random tape

$(m_0, m_1) \leftarrow A(r, pk)$

A is a probabilistic polynomial-time Turing machine, receiving pk as an input and r as a random tape

$|m_0| = |m_1|$

$b' \leftarrow A(r, pk, \psi)$

Although the same letter is used, A is another probabilistic polynomial-time Turing machine, producing b' as an output

IND-CPA Game

- Dealer: $(pk, sk) \Leftarrow G(1^n)$
- Attacker: $r \Leftarrow R,\ (m_0, m_1) \leftarrow A(r, pk)$ $|m_0| = |m_1|$
- Dealer: $b \Leftarrow \{0, 1\},\ \psi \Leftarrow E_{pk}(m_b)$
- Attacker: $b' \leftarrow A(r, pk, \psi)$

$\left| \Pr[b=b'] - 1/2 \right|$: negligible?

Negligible

- A function $\nu(n)$ is *negligible* in n **iff** for any polynomial $p(n)$, there exists N such that for any n, if $n > N$ then $\nu(n) < 1/p(n)$
  - For all sufficiently large n, $\nu(n) < 1/p(n)$
- n is called *security parameter* (e.g., length of keys)

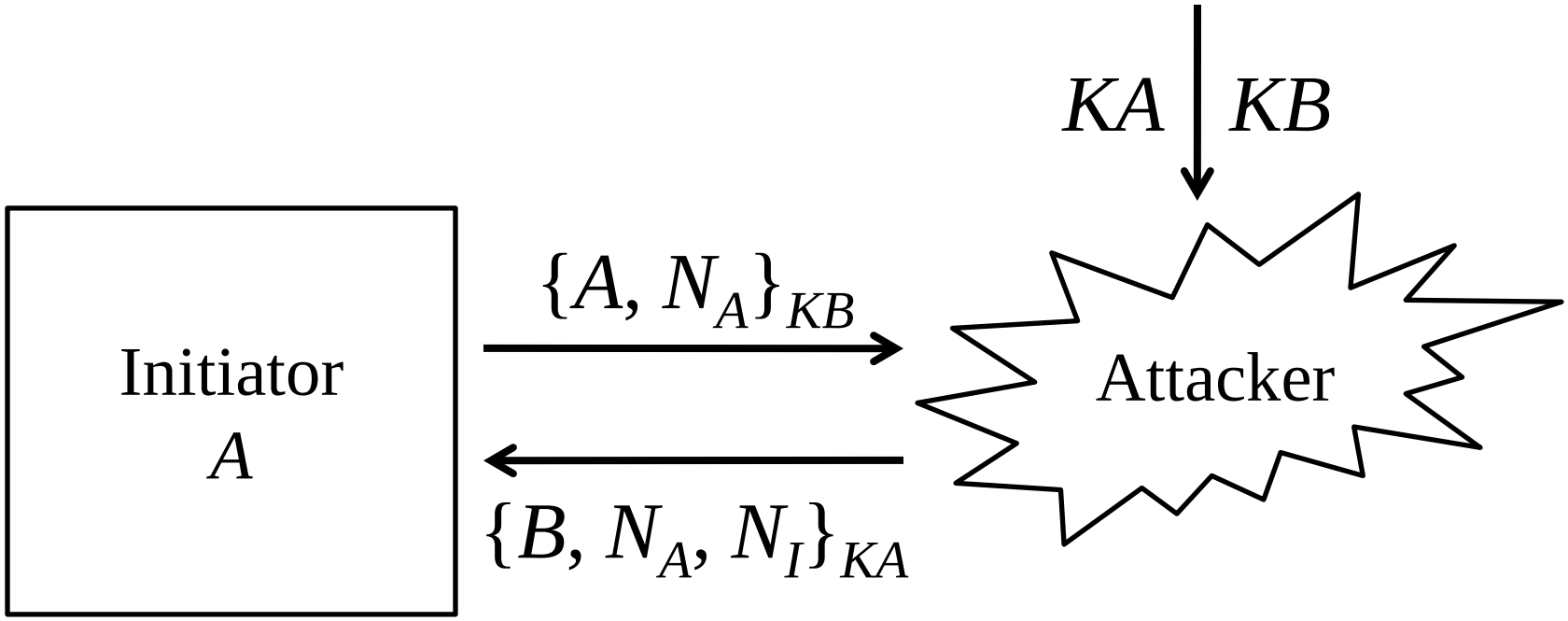
Corrected Protocol

- Needham-Schroeder-Lowe

$$A \rightarrow B : \{A, N_A\}_{KB}$$
$$B \rightarrow A : \{B, N_A, N_B\}_{KA}$$
$$A \rightarrow B : \{N_B\}_{KB}$$

- Is it possible to impersonate B?

The Case Only with One Initiator



There is no responder

IND-CPA Game

Attacker

Dealer

$(KB, SB) \Leftarrow G(1^n)$

KB

$(KA, SA) \Leftarrow G(1^n)$

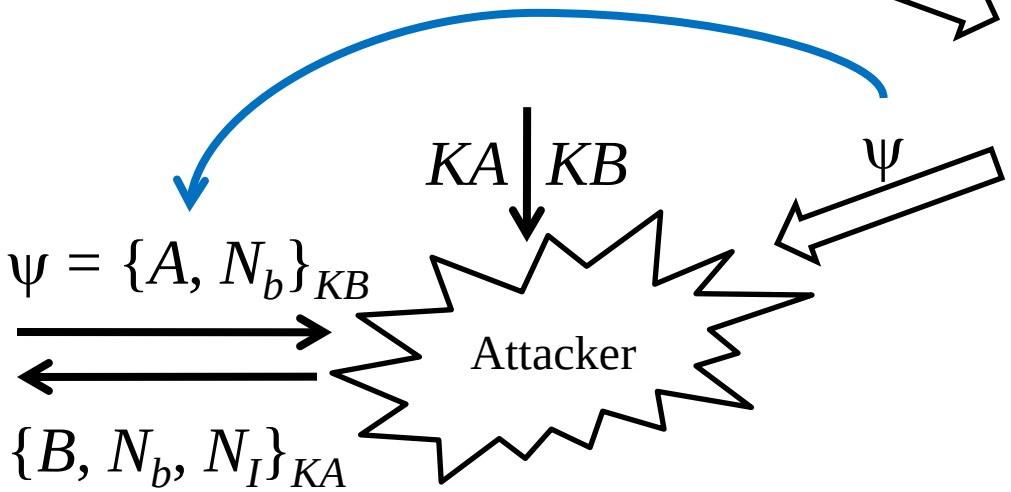
$m_0 \leftarrow (A, N_0)$

$m_1 \leftarrow (A, N_1)$

m_0, m_1

$b \Leftarrow \{0, 1\},$

$\psi \Leftarrow E_{KB}(m_b)$



$(B, N_b, N_I) \leftarrow D_{SA}(\{B, N_b, N_I\}_{KA})$

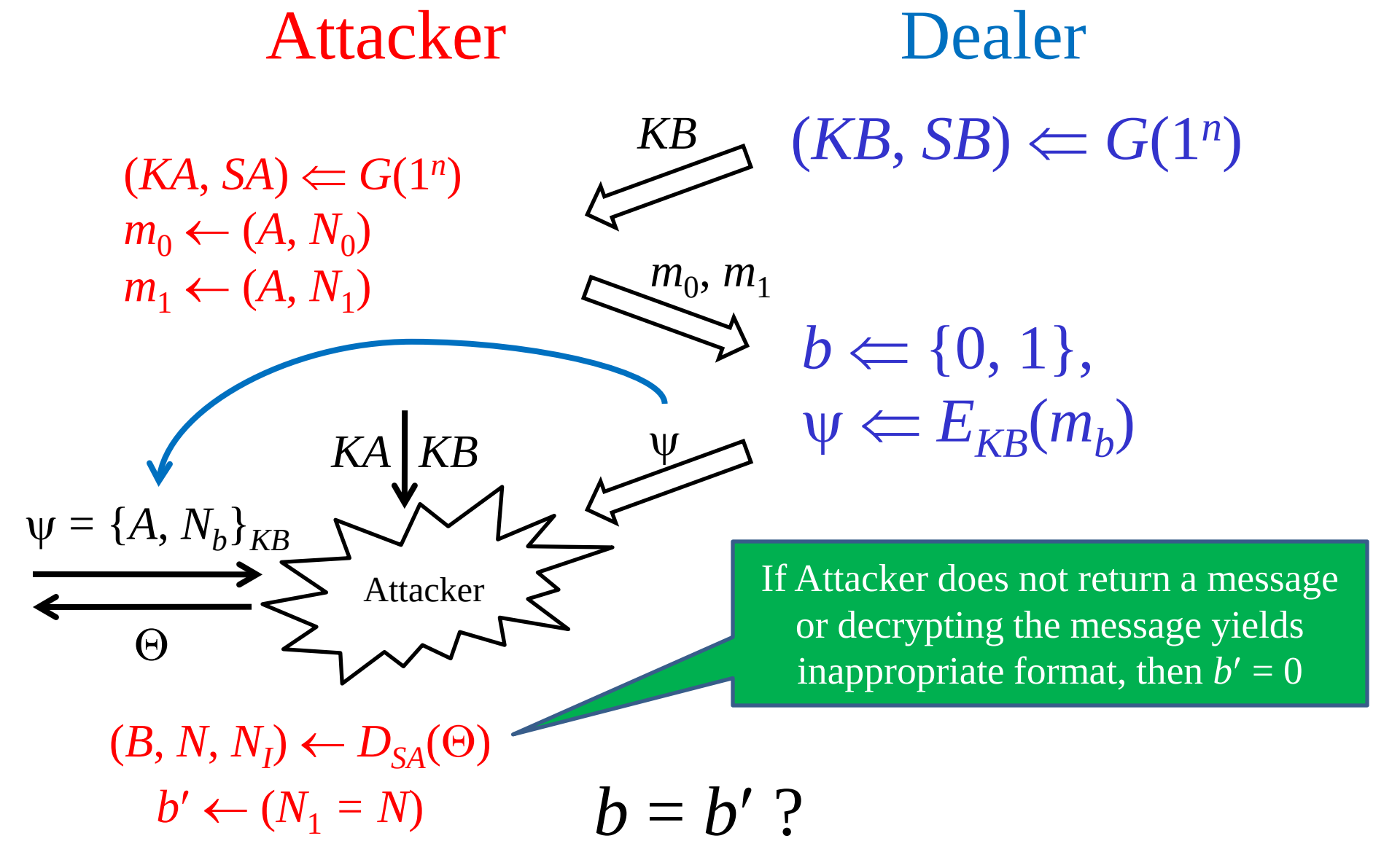
$b' \leftarrow (N_1 = N_b)$

$b = b'$?

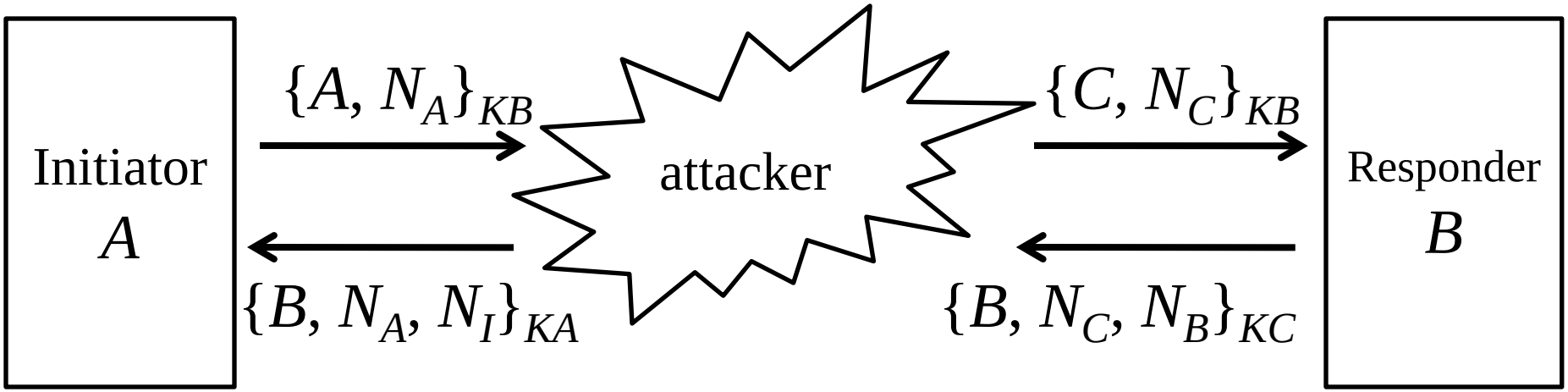
Discussion

- Since N_0 and N_1 are both random, they are symmetric
- If the probability that Attacker can impersonate a responder is p, then the probability that N_b is equal to N_0 is $p/2$ and the probability that N_b is equal to N_1 is also $p/2$; in both cases, $b = b'$ holds and this probability is p
- In other cases $N_1 = N_b$ does not hold, so $b' = 0$; since $b = 0$ and $b = 1$ have the same probability, so the probablity that $b = b'$ holds is $(1–p)/2$
- The probability that $b = b'$ holds is $1/2 + p/2$

IND-CPA Game


Attacker
Dealer
$(KB, SB) \Leftarrow G(1^n)$
KB
$(KA, SA) \Leftarrow G(1^n)$
$m_0 \leftarrow (A, N_0)$
$m_1 \leftarrow (A, N_1)$
m_0, m_1
$b \Leftarrow \{0, 1\},$
$\psi \Leftarrow E_{KB}(m_b)$
KA KB
ψ
$\psi = \{A, N_b\}_{KB}$
Attacker
Θ
If Attacker does not return a message or decrypting the message yields inappropriate format, then $b' = 0$
$(B, N, N_I) \leftarrow D_{SA}(\Theta)$
$b' \leftarrow (N_1 = N)$
$b = b'$?

The Case Also with One Responder



Two cases:

1. Attacker forwards $\{A, N_A\}_{KB}$ to Responder as it is
2. Not so

The Case Also with One Responder

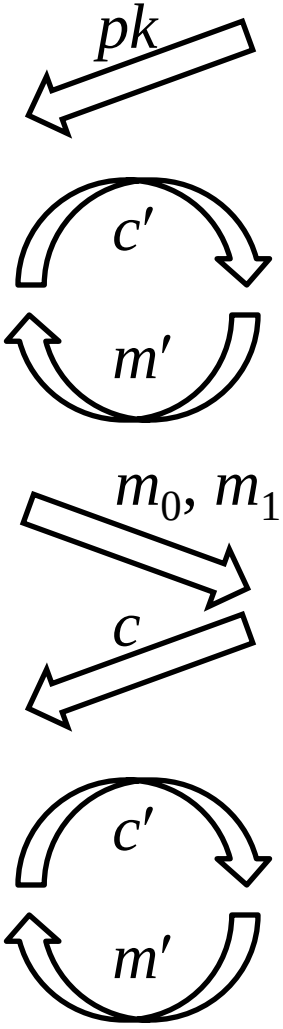
- If Dealer generates two key pairs, then both cases can be dealt with
- Here, we only examine the latter case in connection with the previous example with IND-CPA

IND-CCA2 Game

Indistinguishability under chosen ciphertext attack

Attacker

Dealer



Generate public key pk and secret key sk

Send the result m' of decrypting c'

Generate two messages m_0 and m_1

$|m_0| = |m_1|$

Randomly choose 0 or 1 as b and encrypt m_b by pk as c

Send message c' other than c

Send the result m' of decrypting c'

Guess b as b'

$b = b'$?

IND-CCA2 Game

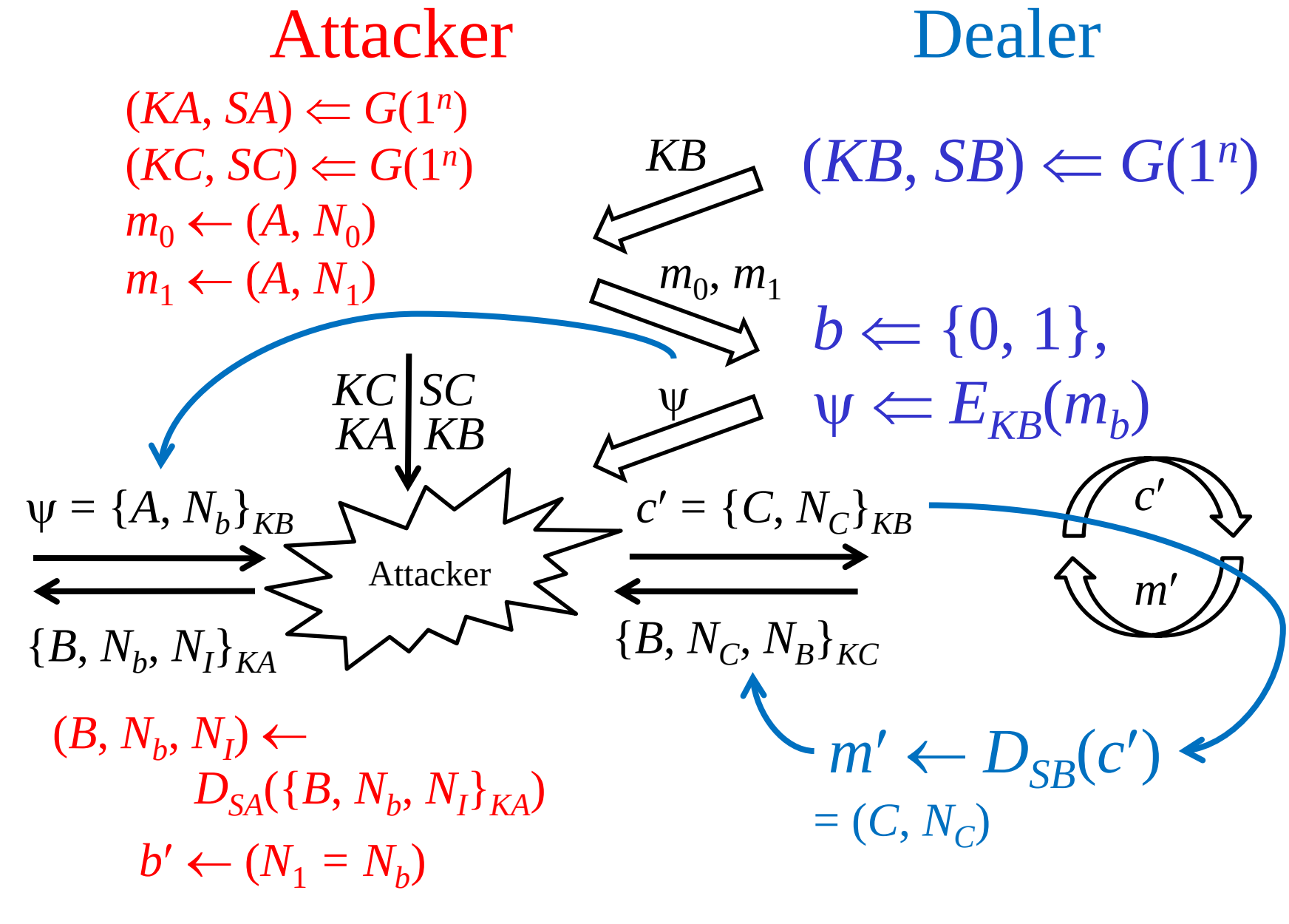
- Dealer: $(pk, sk) \Leftarrow G(1^n)$

  $D_{sk}(\cdot)$: decryption oracle
- Attacker: $r \Leftarrow R$, $(m_0, m_1) \leftarrow A^{D_{sk}}(r, pk)$
- Dealer: $b \Leftarrow \{0, 1\}$, $\psi \Leftarrow E_{pk}(m_b)$
- Attacker: $b' \leftarrow A^{D_{sk}}(r, pk, \psi)$

$|m_0| = |m_1|$

$\left| \Pr[b=b'] - 1/2 \right|$: negligible?

IND-CCA2 Game



Summary (Computational Approach)

- Probabilistic polynomial-time Turing machine
- Security of encryption schemes
  - IND-CPA security
    - Negligible
  - IND-CCA security
- Security of Needham-Schroeder

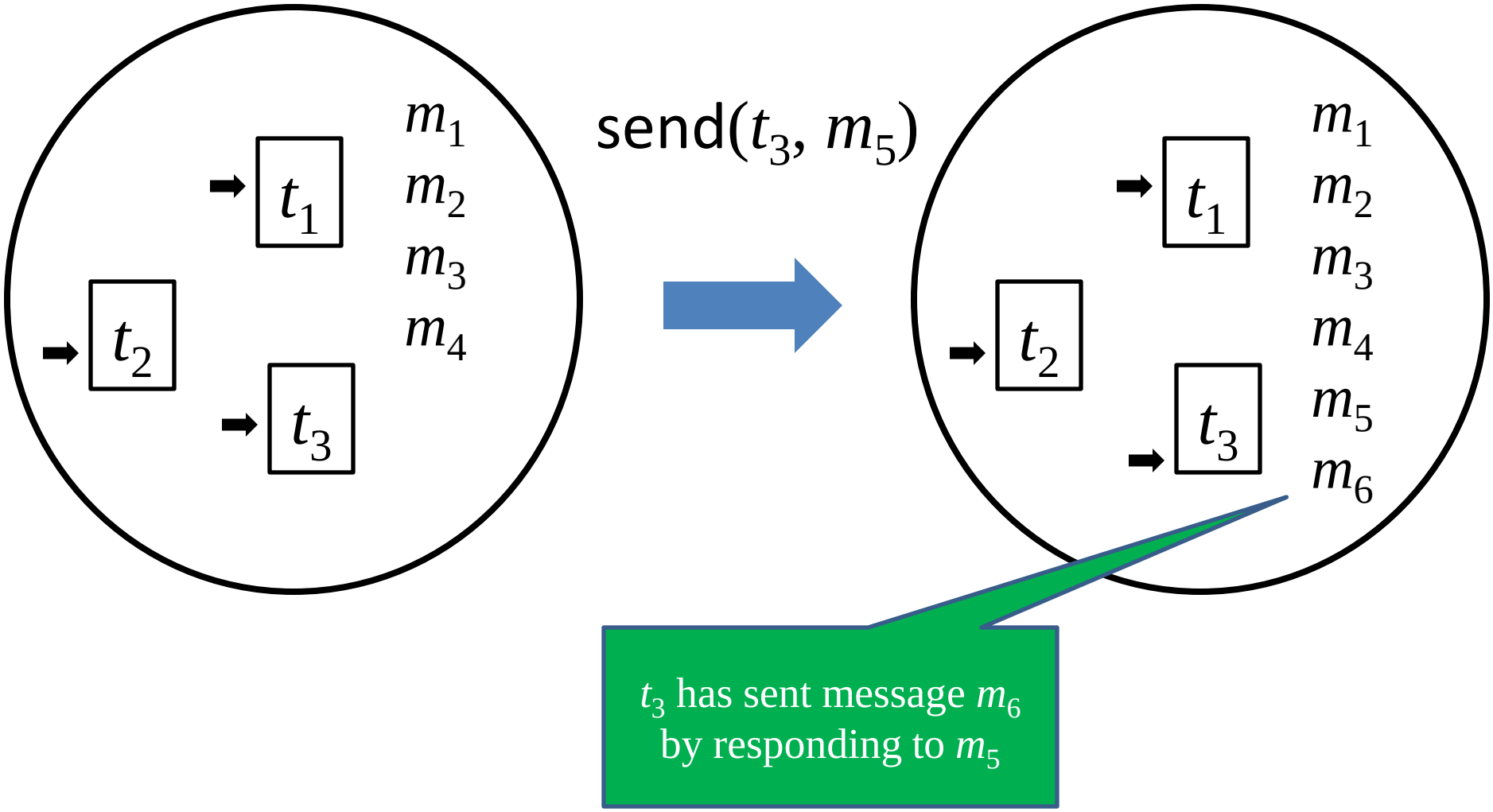
  (Attacker cannot impersonate the responder)
  - Corrected Needham-Schroeder
  - The case only with one Initiator --- IND-CPA assumed
  - The case with one responder --- IND-CCA2 assumed

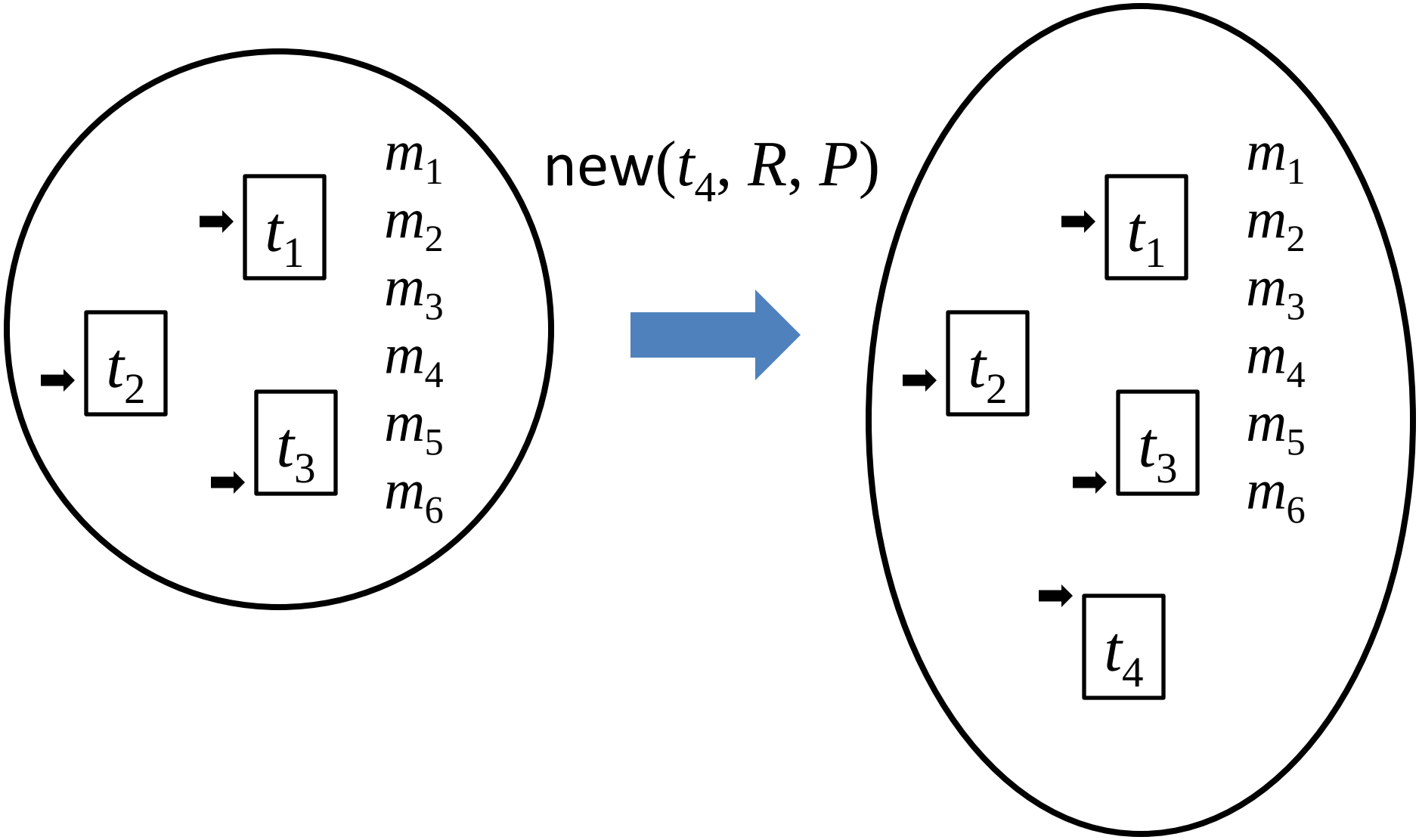
Mapping Lemma

- Generalizes arguments such as the above
- Symbolic traces (executions) always correspond to computational traces (with negligible exceptions)
- Therefore, if a protocol is shown to be secure (e.g., mutually authentic) by the symbolic approach (e.g., model checking or strand space model) then it is also secure in the computational sense

Execution Model of Protocols

- Attacker executes the following instructions
  - send(*t*, *m*): sends a message *m* to a thread *t*
  - new(*t*, *R*, *P*): starts a thread *t* of a regular principal in the role *R* with the parameter *P*
- Local state of a regular principal's thread
  - Current position in (her role of) the protocol
  - Values of parameters and variables
- State of the whole system
  - Collection of states of regular principals' threads
  - Set of messages sent to the network
  - Local state of Attacker


t_1
t_2
t_3
m_1
m_2
m_3
m_4
send(t_3, m_5)
t_1
t_2
t_3
m_1
m_2
m_3
m_4
m_5
m_6
t_3 has sent message m_6
by responding to m_5


t_1
t_2
t_3
m_1
m_2
m_3
m_4
m_5
m_6
new(t_4, R, P)
t_1
t_2
t_3
t_4
m_1
m_2
m_3
m_4
m_5
m_6

Computational Model and Symbolic Model

	computational	symbolic
thread	number (bit vector)	name
message	bit vector	term
constant	tagged bit vector $\langle \mathsf{bit}, 0 \rangle$, $\langle \mathsf{bit}, 1 \rangle$	constant 0, 1
principal name	tagged bit vector $\langle \mathsf{id}, a \rangle$	name (constant) A
key/nonce	tagged bit vector $\langle \mathsf{key}, k \rangle$, $\langle \mathsf{nonce}, n \rangle$	name (constant) K, N
pair	tagged bit vector $\langle \mathsf{pair}, m_1, m_2 \rangle$	constructor term $\langle m_1, m_2 \rangle$
ciphertext	tagged bit vector $\langle \mathsf{enc}, \langle \mathsf{key}, k \rangle, m \rangle$	constructor term $\{m\}_K^r$
parameter/variable value	bit vector	term

Cryptographic Scheme

- Here, a cryptographic scheme with public keys is assumed to satisfy IND-CCA2

Randomness Symbols

- Since encryption is randomized, randomness symbols, denoted by *r*, are introduced to distinguish ciphertexts with the same plaintext

$$\{m\}_K^r$$

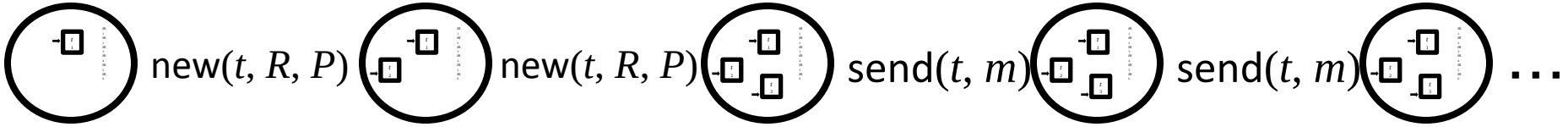
- Each randomness symbol corresponds to a unique ciphertext

Operations by a Regular Principal

- Encrypt a message
- Decrypt a ciphertext (if she owns the secret key)
- Check if two messages are identical → conditional branch
  - Symbolic: identical w.r.t. randomness symbols?
  - Computational: identical as bit vectors?
- Generate a nonce
- Use a constant
- Make a pair (and projections)
- Check the type of a message → conditional branch

Computational/Symbolic Traces

- Sequence of alternating system states and Attacker's operations



- Regular principals follow the protocol description
- Attacker
  - Computational: probabilistic polynomial-time Turing machine
  - Symbolic: follows the Dolev-Yao model
    - Attacker in the strand space model

Mapping Lemma

- Each computational trace corresponds to a symbolic trace
  - With negligible probability, computational traces may deviate from symbolic traces
- Mapping lemma is shown by actually executing the symbolic protocol for a computational attacker and constructing a symbolic trace

From Mapping Lemma

- Properties like authenticity can be proven
  - For example, assume there is a thread t_c of Responder in a computational trace
  - The corresponding symbolic trace should also contain a symbolic thread t_s of Responder
  - If authenticity holds in the symbolic model, there must exist a thread t_s' of Initiator
  - t_s' should correspond to a computational thread t_c' of Initiator
  - t_c' corresponds to t_c, and authenticity also holds in the computational model