



# ***Authentication and Key Distribution***

Breno de Medeiros

Department of Computer Science  
Florida State University

# *Authentication protocols*

- ⑥ Authentication and key-exchange protocols are some of the most fundamental security tasks; they enable the establishment of secure channels and other security services.

# *Authentication protocols*

- ⑥ Authentication and key-exchange protocols are some of the most fundamental security tasks; they enable the establishment of secure channels and other security services.
- ⑥ Have received long attention by the research community; however, the earlier development of authentication protocols was more art than science.

# *Authentication protocols*

- ⑥ Authentication and key-exchange protocols are some of the most fundamental security tasks; they enable the establishment of secure channels and other security services.
- ⑥ Have received long attention by the research community; however, the earlier development of authentication protocols was more art than science.
  - △ Several protocols, such as Needham-Schroeder, that were believed secure for years were then found to contain subtle flaws that could lead to real vulnerabilities in specific scenarios.

# Authentication protocols

- ⑥ Authentication and key-exchange protocols are some of the most fundamental security tasks; they enable the establishment of secure channels and other security services.
- ⑥ Have received long attention by the research community; however, the earlier development of authentication protocols was more art than science.
  - △ Several protocols, such as Needham-Schroeder, that were believed secure for years were then found to contain subtle flaws that could lead to real vulnerabilities in specific scenarios.
  - △ Some protocols that are secure in isolation become insecure when several protocol runs are instantiated simultaneously and interleaved.

# *How to model secure entity authentication and key exchange*

- ⑥ Defining a model for authentication is complex. Unlike attacks against encryption, an attacker against authentication can always “succeed” by relaying messages between honest users.

# *How to model secure entity authentication and key exchange*

- ⑥ Defining a model for authentication is complex. Unlike attacks against encryption, an attacker against authentication can always “succeed” by relaying messages between honest users.
  - △ In some cases this leads to real attacks—for instance, if authentication leads directly to access to resources not further protected by encryption or other security mechanisms.

# *How to model secure entity authentication and key exchange*

- ⑥ Defining a model for authentication is complex. Unlike attacks against encryption, an attacker against authentication can always “succeed” by relaying messages between honest users.
  - △ In some cases this leads to real attacks—for instance, if authentication leads directly to access to resources not further protected by encryption or other security mechanisms.
  - △ In designing authentication mechanisms, the possibility of such attacks must be considered and mitigated using other techniques in addition to secure authentication protocols.

# How to model secure entity authentication and key exchange

- ⑥ Defining a model for authentication is complex. Unlike attacks against encryption, an attacker against authentication can always “succeed” by relaying messages between honest users.
  - △ In some cases this leads to real attacks—for instance, if authentication leads directly to access to resources not further protected by encryption or other security mechanisms.
  - △ In designing authentication mechanisms, the possibility of such attacks must be considered and mitigated using other techniques in addition to secure authentication protocols.
  - △ Herein, an attack is only considered as such if it modifies messages. The notion of *matching conversations* was introduced in [?] to characterize situations where an adversarial success is not due to a protocol break.

# How to model secure entity authentication and key exchange

- ⑥ Defining a model for authentication is complex. Unlike attacks against encryption, an attacker against authentication can always “succeed” by relaying messages between honest users.
  - △ In some cases this leads to real attacks—for instance, if authentication leads directly to access to resources not further protected by encryption or other security mechanisms.
  - △ In designing authentication mechanisms, the possibility of such attacks must be considered and mitigated using other techniques in addition to secure authentication protocols.
  - △ Herein, an attack is only considered as such if it modifies messages. The notion of *matching conversations* was introduced in [?] to characterize situations where an adversarial success is not due to a protocol break.
  - △ This paper only covers two-party authentication protocols.

# *Two-party protocols: Notation*

- Parties  $i, j \in I \subset \{0, 1\}^k$ .

# Two-party protocols: Notation

- Parties  $i, j \in I \subset \{0, 1\}^k$ .
- $a \in \mathcal{K}$ : Secret information (long-lived/long-term key) of sender.

# Two-party protocols: Notation

- ⑥ Parties  $i, j \in I \subset \{0, 1\}^k$ .
- ⑥  $a \in \mathcal{K}$ : Secret information (long-lived/long-term key) of sender.
- ⑥  $\kappa$ : The conversation (view) of the parties, up to the current status of protocol execution.

# Two-party protocols: Notation

- ⊗ Parties  $i, j \in I \subset \{0, 1\}^k$ .
- ⊗  $a \in \mathcal{K}$ : Secret information (long-lived/long-term key) of sender.
- ⊗  $\kappa$ : The conversation (view) of the parties, up to the current status of protocol execution.
- ⊗  $r$ : random value used by the sender.

# Two-party protocols: Notation

- Parties  $i, j \in I \subset \{0, 1\}^k$ .
- $a \in \mathcal{K}$ : Secret information (long-lived/long-term key) of sender.
- $\kappa$ : The conversation (view) of the parties, up to the current status of protocol execution.
- $r$ : random value used by the sender.
- The protocol  $\Pi$  is modeled as a “transition function”  
 $\Pi(1^k, i, j, a, \kappa, r) = (m, \delta, \alpha)$ , where:

# Two-party protocols: Notation

- ⑥ Parties  $i, j \in I \subset \{0, 1\}^k$ .
- ⑥  $a \in \mathcal{K}$ : Secret information (long-lived/long-term key) of sender.
- ⑥  $\kappa$ : The conversation (view) of the parties, up to the current status of protocol execution.
- ⑥  $r$ : random value used by the sender.
- ⑥ The protocol  $\Pi$  is modeled as a “transition function”  
 $\Pi(1^k, i, j, a, \kappa, r) = (m, \delta, \alpha)$ , where:
  - △  $m \in \{0, 1\}^*$ : The next message to send.

# Two-party protocols: Notation

- ⑥ Parties  $i, j \in I \subset \{0, 1\}^k$ .
- ⑥  $a \in \mathcal{K}$ : Secret information (long-lived/long-term key) of sender.
- ⑥  $\kappa$ : The conversation (view) of the parties, up to the current status of protocol execution.
- ⑥  $r$ : random value used by the sender.
- ⑥ The protocol  $\Pi$  is modeled as a “transition function”  
 $\Pi(1^k, i, j, a, \kappa, r) = (m, \delta, \alpha)$ , where:
  - △  $m \in \{0, 1\}^*$ : The next message to send.
  - △  $\delta \in \{A, R, *\}$ : Decision to make.

# Two-party protocols: Notation

- ⑥ Parties  $i, j \in I \subset \{0, 1\}^k$ .
- ⑥  $a \in \mathcal{K}$ : Secret information (long-lived/long-term key) of sender.
- ⑥  $\kappa$ : The conversation (view) of the parties, up to the current status of protocol execution.
- ⑥  $r$ : random value used by the sender.
- ⑥ The protocol  $\Pi$  is modeled as a “transition function”  
 $\Pi(1^k, i, j, a, \kappa, r) = (m, \delta, \alpha)$ , where:
  - △  $m \in \{0, 1\}^*$ : The next message to send.
  - △  $\delta \in \{A, R, *\}$ : Decision to make.
  - △  $\alpha \in \{0, 1\}^*$ : Private output of the executing party.

# Adversarial model

- ⑥  $\mathcal{A}$  is a probabilistic machine with an infinite collection of oracles  $\Pi_{i,j}^s$ , which represent attempt by party  $i$  to authenticate itself to  $j$  during session  $s \in \mathbb{N}$ .

# Adversarial model

- ⑥  $\mathcal{A}$  is a probabilistic machine with an infinite collection of oracles  $\Pi_{i,j}^s$ , which represent attempt by party  $i$  to authenticate itself to  $j$  during session  $s \in \mathbb{N}$ .
- ⑥  $\mathcal{A}$ 's queries take the form of tuples  $(i, j, s, x)$ , meaning that  $\mathcal{A}$  sends  $x$  to  $i$  in session  $s$  claiming to have come from  $j$ .

# Adversarial model

- ⑥  $\mathcal{A}$  is a probabilistic machine with an infinite collection of oracles  $\Pi_{i,j}^s$ , which represent attempt by party  $i$  to authenticate itself to  $j$  during session  $s \in \mathbb{N}$ .
- ⑥  $\mathcal{A}$ 's queries take the form of tuples  $(i, j, s, x)$ , meaning that  $\mathcal{A}$  sends  $x$  to  $i$  in session  $s$  claiming to have come from  $j$ .
- ⑥ Initialization:

# Adversarial model

- ⑥  $\mathcal{A}$  is a probabilistic machine with an infinite collection of oracles  $\Pi_{i,j}^s$ , which represent attempt by party  $i$  to authenticate itself to  $j$  during session  $s \in \mathbb{N}$ .
- ⑥  $\mathcal{A}$ 's queries take the form of tuples  $(i, j, s, x)$ , meaning that  $\mathcal{A}$  sends  $x$  to  $i$  in session  $s$  claiming to have come from  $j$ .
- ⑥ Initialization:
  - △ The simulator  $\mathcal{S}$  initiates long-term keys for all the honest parties, as random or pseudo-randomly chosen values.

# Adversarial model

- ⑥  $\mathcal{A}$  is a probabilistic machine with an infinite collection of oracles  $\Pi_{i,j}^s$ , which represent attempt by party  $i$  to authenticate itself to  $j$  during session  $s \in \mathbb{N}$ .
- ⑥  $\mathcal{A}$ 's queries take the form of tuples  $(i, j, s, x)$ , meaning that  $\mathcal{A}$  sends  $x$  to  $i$  in session  $s$  claiming to have come from  $j$ .
- ⑥ Initialization:
  - △ The simulator  $\mathcal{S}$  initiates long-term keys for all the honest parties, as random or pseudo-randomly chosen values.
  - △  $\mathcal{S}$  initializes a source of random bits for all honest parties.

# Adversarial model

- ⑥  $\mathcal{A}$  is a probabilistic machine with an infinite collection of oracles  $\Pi_{i,j}^s$ , which represent attempt by party  $i$  to authenticate itself to  $j$  during session  $s \in \mathbb{N}$ .
- ⑥  $\mathcal{A}$ 's queries take the form of tuples  $(i, j, s, x)$ , meaning that  $\mathcal{A}$  sends  $x$  to  $i$  in session  $s$  claiming to have come from  $j$ .
- ⑥ Initialization:
  - △ The simulator  $\mathcal{S}$  initiates long-term keys for all the honest parties, as random or pseudo-randomly chosen values.
  - △  $\mathcal{S}$  initializes a source of random bits for all honest parties.
  - △  $\kappa_{i,j}^s = \text{start}$  are initialized, and will accumulate the conversations between  $(i, j)$  in session  $s$ .

# Simulations



- ⑥  $S$  runs  $\mathcal{A}$ , which is initialized with a source of random bits.

# Simulations

- ⑥  $S$  runs  $\mathcal{A}$ , which is initialized with a source of random bits.
- ⑥ When  $\mathcal{A}$  makes a query  $(i, j, s, x)$ ,  $\Pi_{i,j}^s$  computes  $(m, \delta, \alpha) = \Pi(1^k, i, j, a_i, \kappa_{i,j}^s \cdot x, r_{i,j}^s)$ , returning  $(m, \delta)$  to  $\mathcal{A}$  and updating the conversation with  $\kappa_{i,j}^s \cdot x$ .

# Simulations

- ⑥  $S$  runs  $\mathcal{A}$ , which is initialized with a source of random bits.
- ⑥ When  $\mathcal{A}$  makes a query  $(i, j, s, x)$ ,  $\Pi_{i,j}^s$  computes  $(m, \delta, \alpha) = \Pi(1^k, i, j, a_i, \kappa_{i,j}^s \cdot x, r_{i,j}^s)$ , returning  $(m, \delta)$  to  $\mathcal{A}$  and updating the conversation with  $\kappa_{i,j}^s \cdot x$ .
- ⑥ A *benign* adversary  $\mathcal{A}$  is restricted to:

# Simulations

- ⑥  $S$  runs  $\mathcal{A}$ , which is initialized with a source of random bits.
- ⑥ When  $\mathcal{A}$  makes a query  $(i, j, s, x)$ ,  $\Pi_{i,j}^s$  computes  $(m, \delta, \alpha) = \Pi(1^k, i, j, a_i, \kappa_{i,j}^s \cdot x, r_{i,j}^s)$ , returning  $(m, \delta)$  to  $\mathcal{A}$  and updating the conversation with  $\kappa_{i,j}^s \cdot x$ .
- ⑥ A *benign* adversary  $\mathcal{A}$  is restricted to:
  - △ Choosing two oracles  $\Pi_{i,j}^{s_1}$  and  $\Pi_{j,i}^{s_2}$ .

# Simulations

- ⑥  $S$  runs  $\mathcal{A}$ , which is initialized with a source of random bits.
- ⑥ When  $\mathcal{A}$  makes a query  $(i, j, s, x)$ ,  $\Pi_{i,j}^s$  computes  $(m, \delta, \alpha) = \Pi(1^k, i, j, a_i, \kappa_{i,j}^s \cdot x, r_{i,j}^s)$ , returning  $(m, \delta)$  to  $\mathcal{A}$  and updating the conversation with  $\kappa_{i,j}^s \cdot x$ .
- ⑥ A *benign* adversary  $\mathcal{A}$  is restricted to:
  - △ Choosing two oracles  $\Pi_{i,j}^{s_1}$  and  $\Pi_{j,i}^{s_2}$ .
  - △ Causes  $\Pi_{i,j}^{s_1}$  to start, obtaining  $(m_1, \delta_1)$  as response to query  $(i, j, s_1, \text{start})$ .

# Simulations

- ⑥  $S$  runs  $\mathcal{A}$ , which is initialized with a source of random bits.
- ⑥ When  $\mathcal{A}$  makes a query  $(i, j, s, x)$ ,  $\Pi_{i,j}^s$  computes  $(m, \delta, \alpha) = \Pi(1^k, i, j, a_i, \kappa_{i,j}^s \cdot x, r_{i,j}^s)$ , returning  $(m, \delta)$  to  $\mathcal{A}$  and updating the conversation with  $\kappa_{i,j}^s \cdot x$ .
- ⑥ A *benign* adversary  $\mathcal{A}$  is restricted to:
  - △ Choosing two oracles  $\Pi_{i,j}^{s_1}$  and  $\Pi_{j,i}^{s_2}$ .
  - △ Causes  $\Pi_{i,j}^{s_1}$  to start, obtaining  $(m_1, \delta_1)$  as response to query  $(i, j, s_1, \text{start})$ .
  - △ Conveys flows between the oracles, sending query  $(j, i, s_2, m_i)$ , if  $(m_i, \delta_i)$  was the response to a preceding query  $(i, j, s_1, m_{i-1})$ , obtaining answer  $(m_{i+1}, \delta_{i+1})$ , which is then forwarded as query  $(i, j, s_1, m_{i+1})$ , and so forth.

# Conversations

- ⑥ At the end of each simulation, the transcripts  $\kappa_{i,j}^s$  of each oracle  $\Pi_{i,j}^s$  are examined, including its final decision  $\delta_{i,j}^s$ .

# Conversations

- ⑥ At the end of each simulation, the transcripts  $\kappa_{i,j}^s$  of each oracle  $\Pi_{i,j}^s$  are examined, including its final decision  $\delta_{i,j}^s$ .
- ⑥ Each conversation is marked as *good*, or *bad*, depending on whether  $\mathcal{A}$  subverted the protocol goals during that run.

# Conversations

- ⑥ At the end of each simulation, the transcripts  $\kappa_{i,j}^s$  of each oracle  $\Pi_{i,j}^s$  are examined, including its final decision  $\delta_{i,j}^s$ .
- ⑥ Each conversation is marked as *good*, or *bad*, depending on whether  $\mathcal{A}$  subverted the protocol goals during that run.
- ⑥ Security is defined in terms of probabilities of bad conversations, computed as distributions over the random input bits to honest parties, and the adversary.

# Picture of matching conversations

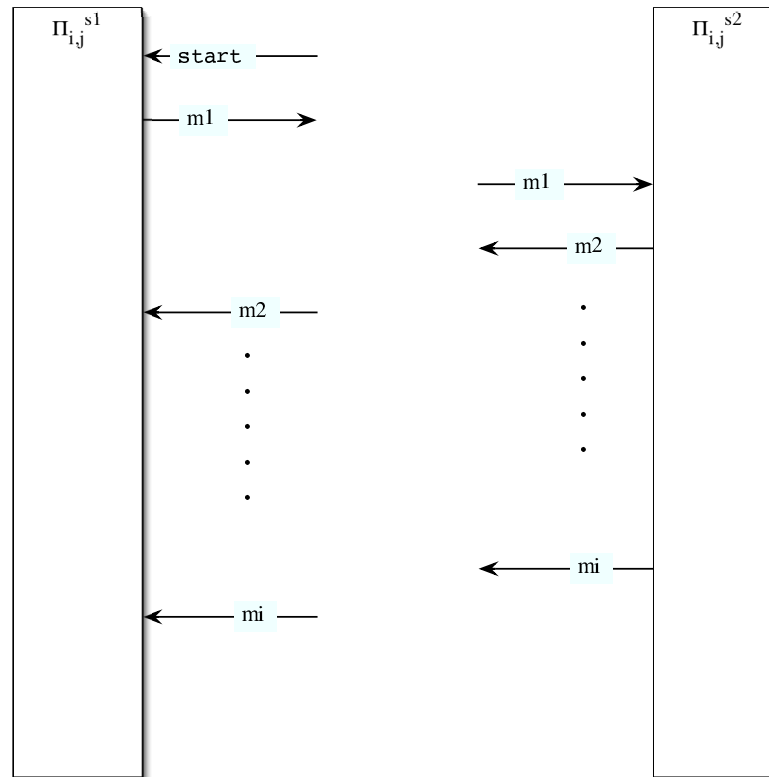


Figure 0: A matching conversation between two oracles.

# ***Mutual authentication***

- ⑥  $\Pi$  is a secure mutual authentication protocol if:

# Mutual authentication

- ⑥  $\Pi$  is a secure mutual authentication protocol if:

**Correctness:** If oracles  $\Pi_{i,j}^{s_1}$  and  $\Pi_{i,j}^{s_2}$  have matching conversations, their final state  $\delta_{i,j}^{s_1}$  and  $\delta_{i,j}^{s_2}$  must both be ACCEPT:  $A$ .

# Mutual authentication

- ⑥  $\Pi$  is a secure mutual authentication protocol if:

**Correctness:** If oracles  $\Pi_{i,j}^{s_1}$  and  $\Pi_{i,j}^{s_2}$  have matching conversations, their final state  $\delta_{i,j}^{s_1}$  and  $\delta_{i,j}^{s_2}$  must both be ACCEPT:  $A$ .

**Soundness:** The probability that there is an oracle  $\Pi_{i,j}^{s_1}$  with decision  $\delta_{i,j}^s = A$  and no oracle  $\Pi_{j,i}^{s'}$  with a matching conversation is negligible.

# Mutual authentication

- ⑥  $\Pi$  is a secure mutual authentication protocol if:

**Correctness:** If oracles  $\Pi_{i,j}^{s_1}$  and  $\Pi_{i,j}^{s_2}$  have matching conversations, their final state  $\delta_{i,j}^{s_1}$  and  $\delta_{i,j}^{s_2}$  must both be ACCEPT:  $A$ .

**Soundness:** The probability that there is an oracle  $\Pi_{i,j}^{s_1}$  with decision  $\delta_{i,j}^s = A$  and no oracle  $\Pi_{j,i}^{s'}$  with a matching conversation is negligible.

- ⑥ In the following, some secure mutual authentication protocols are introduced. Let  $\{f_a(\cdot)\}_{a \in \mathcal{K}}$  denote a pseudo-random function family, and  $[x]_a := (x, f_a(x))$ .

# A secure MAP



Figure 1: Protocol MAP2: Flows for an authenticated exchange of three text strings.

# A secure MAP

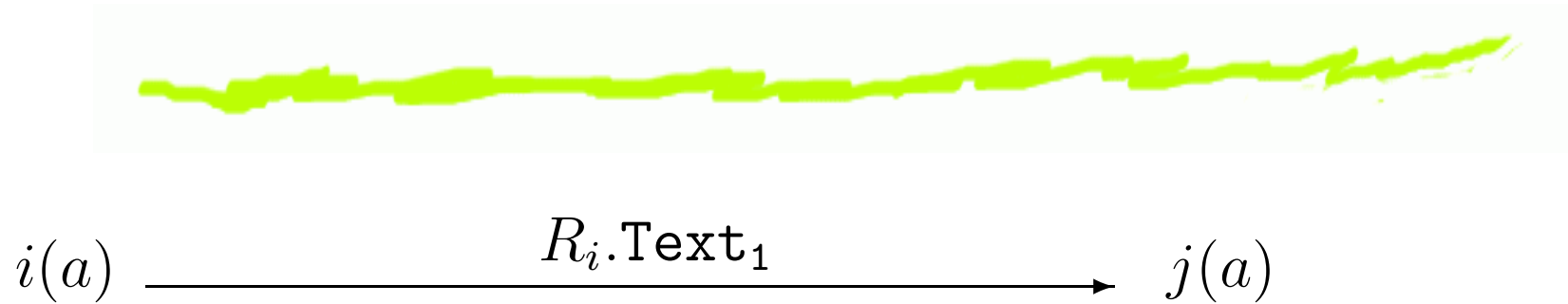


Figure 1: Protocol MAP2: Flows for an authenticated exchange of three text strings.

# A secure MAP

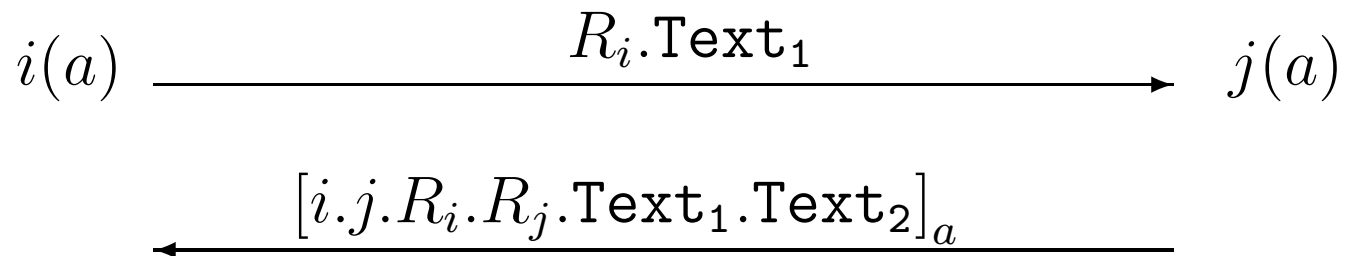


Figure 1: Protocol MAP2: Flows for an authenticated exchange of three text strings.

# A secure MAP

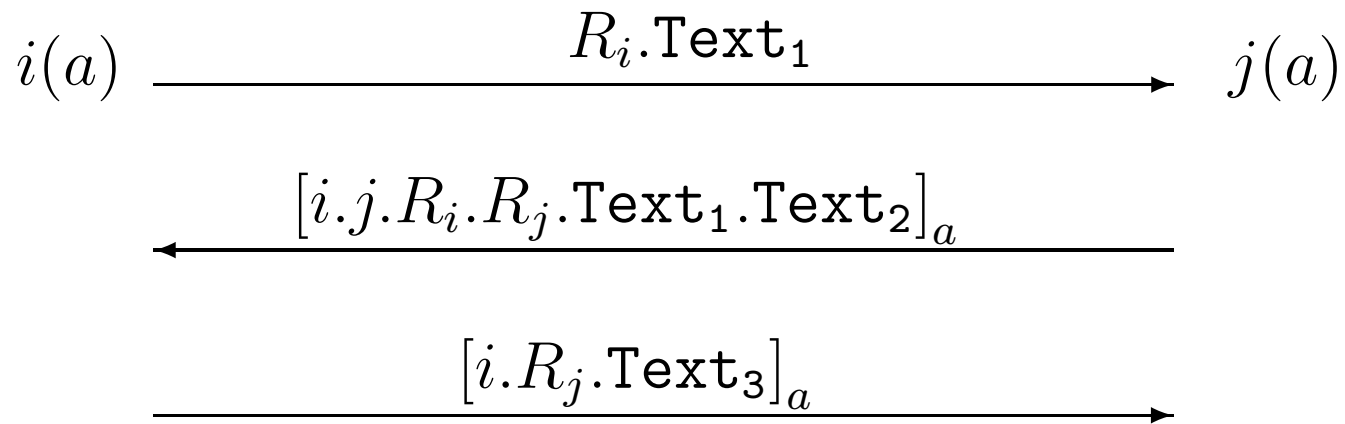


Figure 1: Protocol MAP2: Flows for an authenticated exchange of three text strings.

# Key exchange protocols

- ⑥ Parties have private outputs  $\alpha_{i,j}^s$  (*session keys*).

# Key exchange protocols

- ⑥ Parties have private outputs  $\alpha_{i,j}^s$  (*session keys*).
- ⑥ Compromise of some session keys should not affect the security of other session keys.

# Key exchange protocols

- ⑥ Parties have private outputs  $\alpha_{i,j}^s$  (*session keys*).
- ⑥ Compromise of some session keys should not affect the security of other session keys.
  - △  $\mathcal{A}$  is able to ask for queries:  $(i, j, s, \text{reveal})$ , which result in the private output  $\alpha_{i,j}^s$  computed by  $\Pi_{i,j}^s$  being returned to  $\mathcal{A}$  (assuming that  $\Pi_{i,j}$  has accepted in session  $s$ ).

# Key exchange protocols

- ⑥ Parties have private outputs  $\alpha_{i,j}^s$  (*session keys*).
- ⑥ Compromise of some session keys should not affect the security of other session keys.
  - △  $\mathcal{A}$  is able to ask for queries:  $(i, j, s, \text{reveal})$ , which result in the private output  $\alpha_{i,j}^s$  computed by  $\Pi_{i,j}^s$  being returned to  $\mathcal{A}$  (assuming that  $\Pi_{i,j}$  has accepted in session  $s$ ).
  - △ At some point,  $\mathcal{A}$  selects a *fresh* oracle  $\Pi_{i,j}^s$  (i.e., an  $\Pi_{i,j}^s$  accepted session  $s$  and  $\alpha_{i,j}^s$  has not been revealed to  $\mathcal{A}$ ) and sends it a query  $(i, j, s, \text{test})$ .

# Key exchange protocols

- ⑥ Parties have private outputs  $\alpha_{i,j}^s$  (*session keys*).
- ⑥ Compromise of some session keys should not affect the security of other session keys.
  - △  $\mathcal{A}$  is able to ask for queries:  $(i, j, s, \text{reveal})$ , which result in the private output  $\alpha_{i,j}^s$  computed by  $\Pi_{i,j}^s$  being returned to  $\mathcal{A}$  (assuming that  $\Pi_{i,j}$  has accepted in session  $s$ ).
  - △ At some point,  $\mathcal{A}$  selects a *fresh* oracle  $\Pi_{i,j}^s$  (i.e., an  $\Pi_{i,j}^s$  accepted session  $s$  and  $\alpha_{i,j}^s$  has not been revealed to  $\mathcal{A}$ ) and sends it a query  $(i, j, s, \text{test})$ .
  - △ The answer to the *test* query is, with equal probability, either  $\alpha_{i,j}^s$  or chosen according to the “native” distribution of session keys. The adversary produces a guess bit  $b = 0$  (random) or  $b = 1$  (real) and wins if his guess is correct.

# Secure key exchange



- ⑥  $\Pi$  is a secure key exchange protocol if:

# Secure key exchange

- ⑥  $\Pi$  is a secure key exchange protocol if:

**Correctness:** If  $\mathcal{A}$  is a benign adversary, and has conveyed messages between oracles  $\Pi_{i,j}^{s_1}$  and  $\Pi_{i,j}^{s_2}$ , then both accept,  $\alpha_{i,j}^{s_1} = \alpha_{i,j}^{s_2}$ , and the distribution of this common output follows the prescribed distribution for session keys.

# Secure key exchange

- ⑥  $\Pi$  is a secure key exchange protocol if:

**Correctness:** If  $\mathcal{A}$  is a benign adversary, and has conveyed messages between oracles  $\Pi_{i,j}^{s_1}$  and  $\Pi_{i,j}^{s_2}$ , then both accept,  $\alpha_{i,j}^{s_1} = \alpha_{i,j}^{s_2}$ , and the distribution of this common output follows the prescribed distribution for session keys.

**Soundness:**  $\mathcal{A}$  has only negligible advantage over a random guess in distinguishing real from random session keys.

# Secure key exchange


- ⑥  $\Pi$  is a secure key exchange protocol if:

**Correctness:** If  $\mathcal{A}$  is a benign adversary, and has conveyed messages between oracles  $\Pi_{i,j}^{s_1}$  and  $\Pi_{i,j}^{s_2}$ , then both accept,  $\alpha_{i,j}^{s_1} = \alpha_{i,j}^{s_2}$ , and the distribution of this common output follows the prescribed distribution for session keys.

**Soundness:**  $\mathcal{A}$  has only negligible advantage over a random guess in distinguishing real from random session keys.

- ⑥ In the following, let  $\{f'_{a_2}(\cdot)\}_{a_2}$  be a pseudo-random function family, and  $\{x\}_{a_2} := (r, x \oplus f_{a_2}(r))$  (XOR encryption), with random  $r$ .

# An AKEP



$i(a_1, a_2)$

$j(a_1, a_2)$

Figure 2: Protocol AKEP1: Flows for an authenticated key exchange;  $\alpha$  is the agreed session key.

# An AKEP



Figure 2: Protocol AKEP1: Flows for an authenticated key exchange;  $\alpha$  is the agreed session key.

# An AKEP

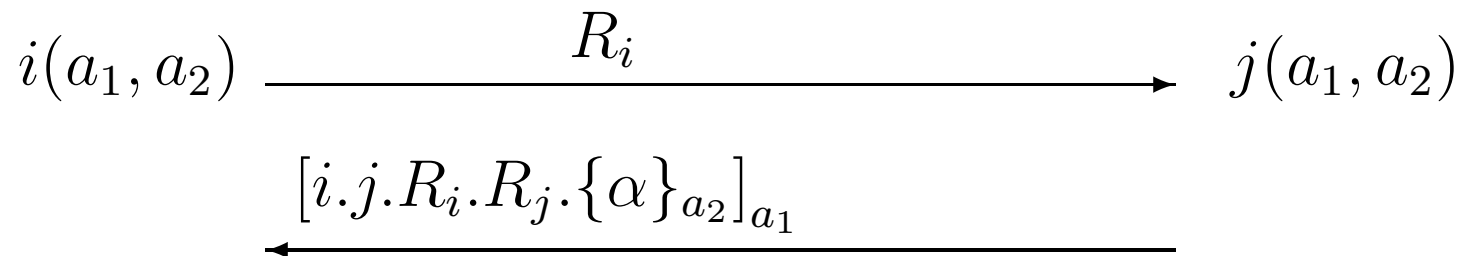


Figure 2: Protocol AKEP1: Flows for an authenticated key exchange;  $\alpha$  is the agreed session key.

# An AKEP

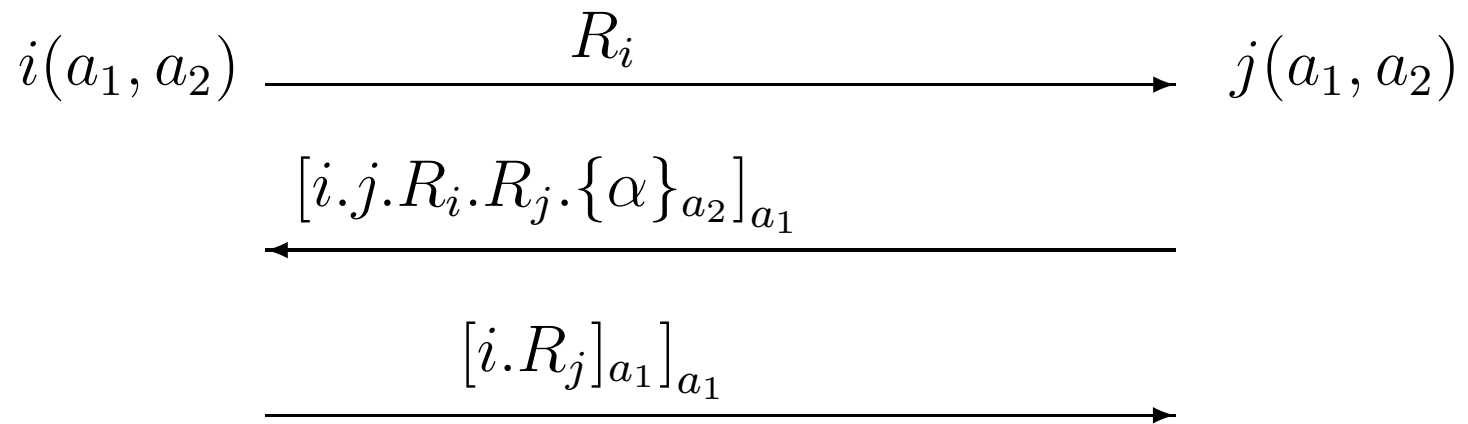


Figure 2: Protocol AKEP1: Flows for an authenticated key exchange;  $\alpha$  is the agreed session key.