Integrating Causality in Messaging Channels

Shan Chen

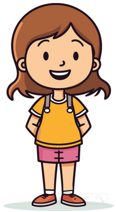
南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Marc Fischlin

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Motivation

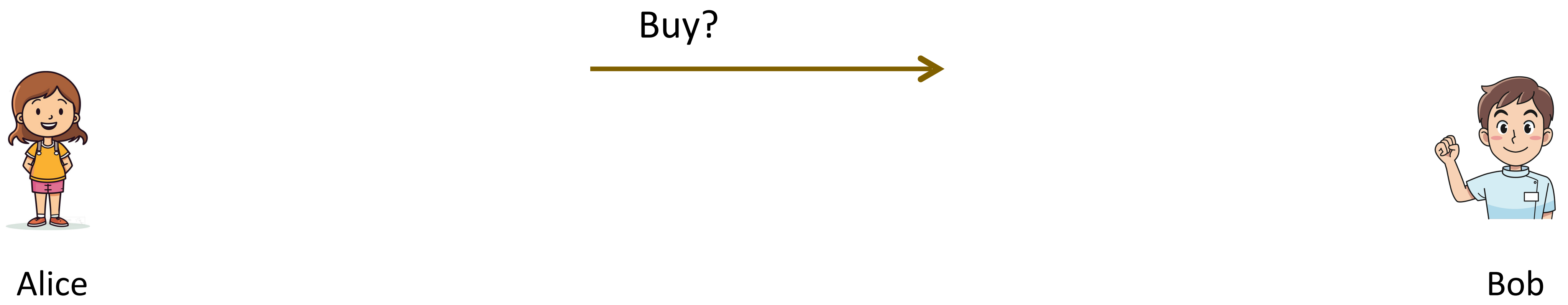
Secure (End-to-End) Messaging: Causality



Alice

Bob

Secure (End-to-End) Messaging: Causality

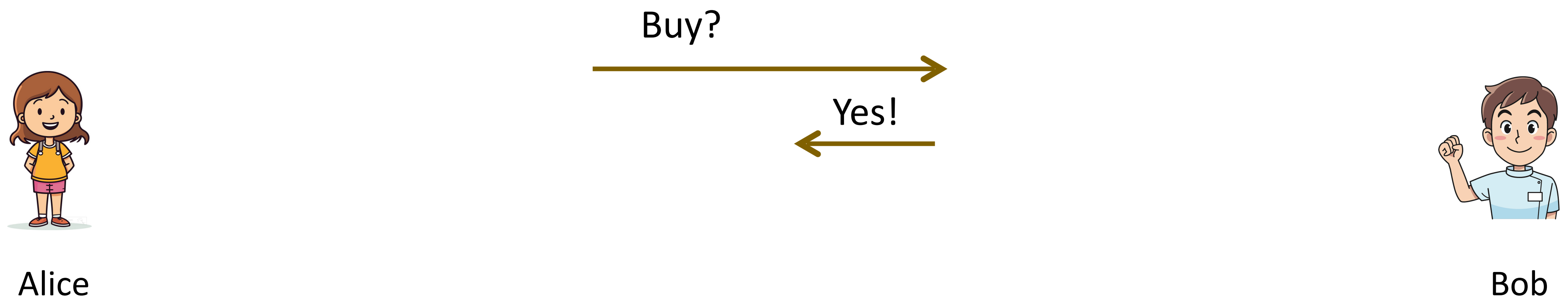


Buy?

Alice

Bob

Secure (End-to-End) Messaging: Causality



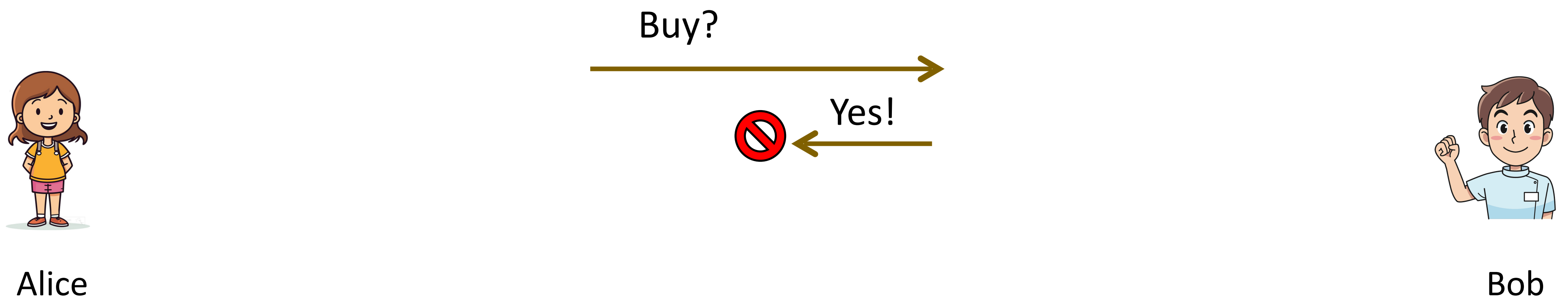
Buy?

Yes!

Alice

Bob

Secure (End-to-End) Messaging: Causality



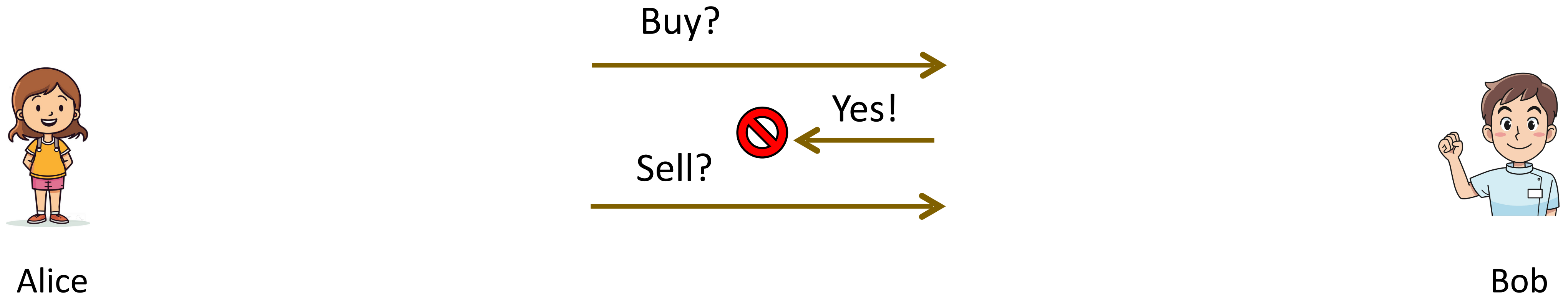
Buy?

Yes!

Alice

Bob

Secure (End-to-End) Messaging: Causality



Buy?

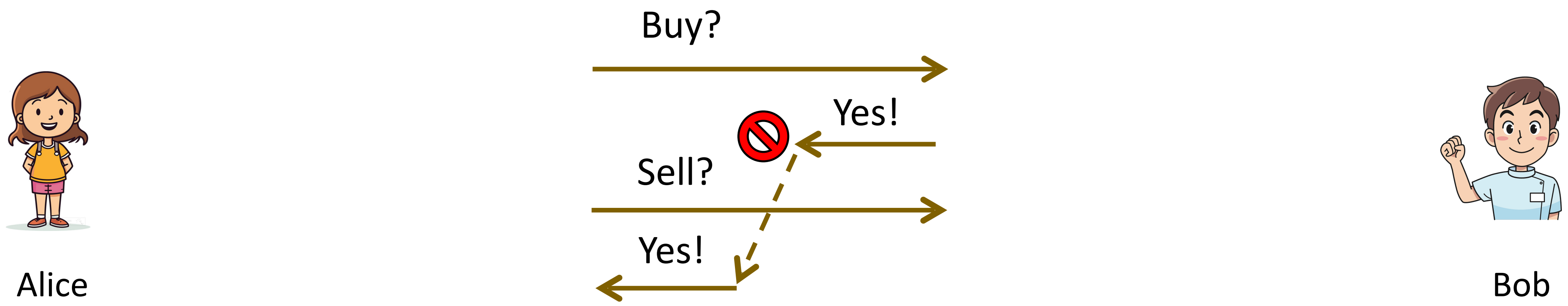
Yes!

Sell?

Alice

Bob

Secure (End-to-End) Messaging: Causality



Buy?

Yes!

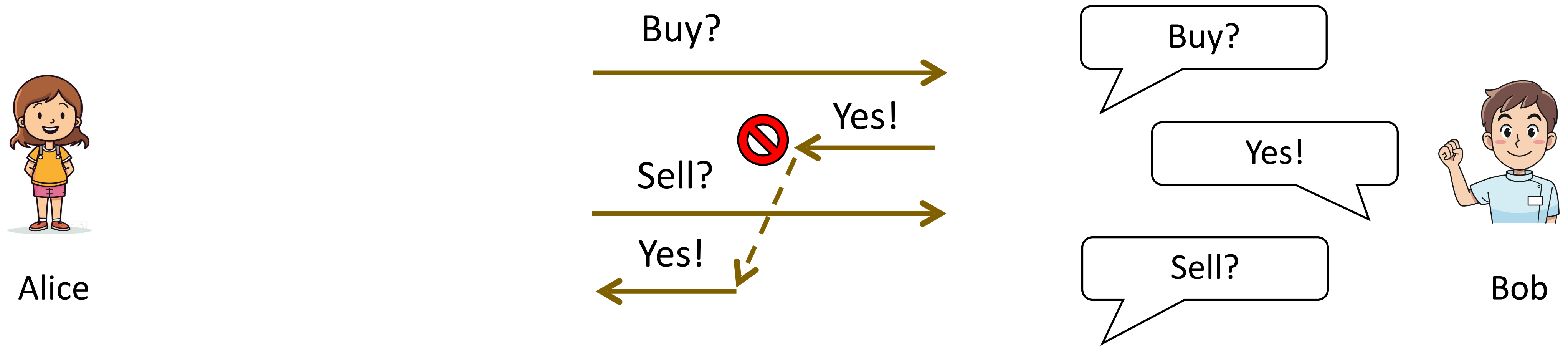
Sell?

Yes!

Alice

Bob

Secure (End-to-End) Messaging: Causality



Alice

Buy?

Yes!

Sell?

Yes!

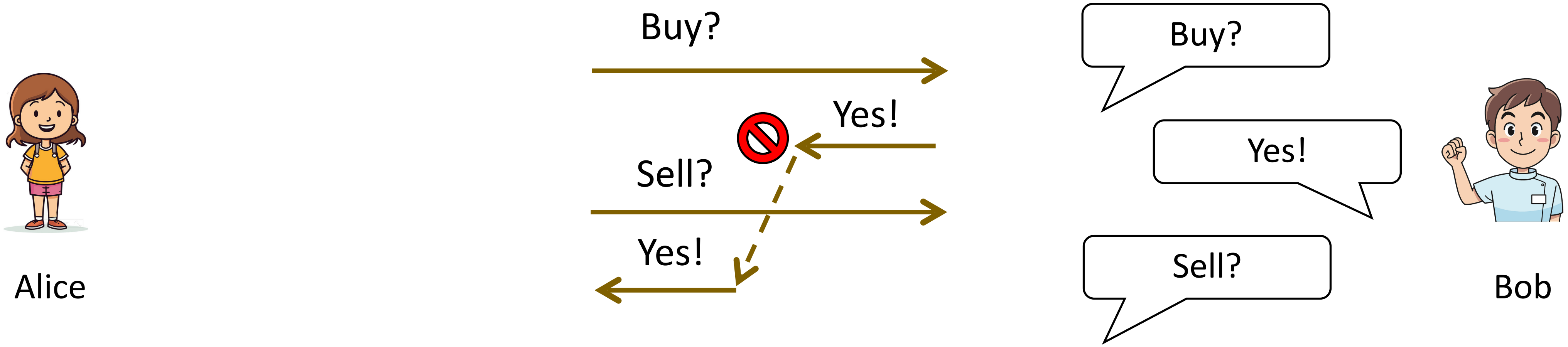
Buy?

Yes!

Sell?

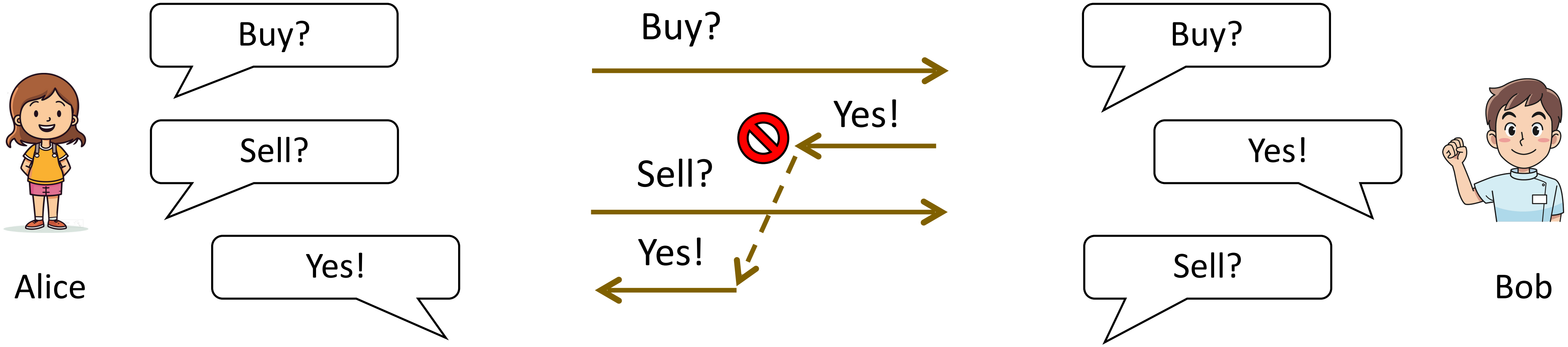
Bob

Secure (End-to-End) Messaging: Causality



Bob recommends buy

Secure (End-to-End) Messaging: Causality



Buy?

Sell?

Yes!

Alice

Buy?

Yes!

Sell?

Yes!

Buy?

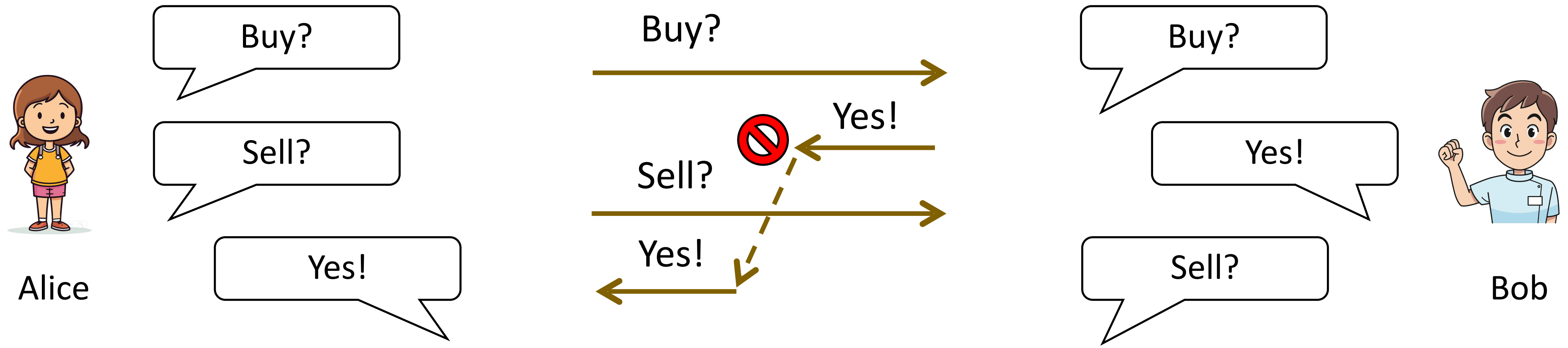
Yes!

Sell?

Bob

Bob recommends buy

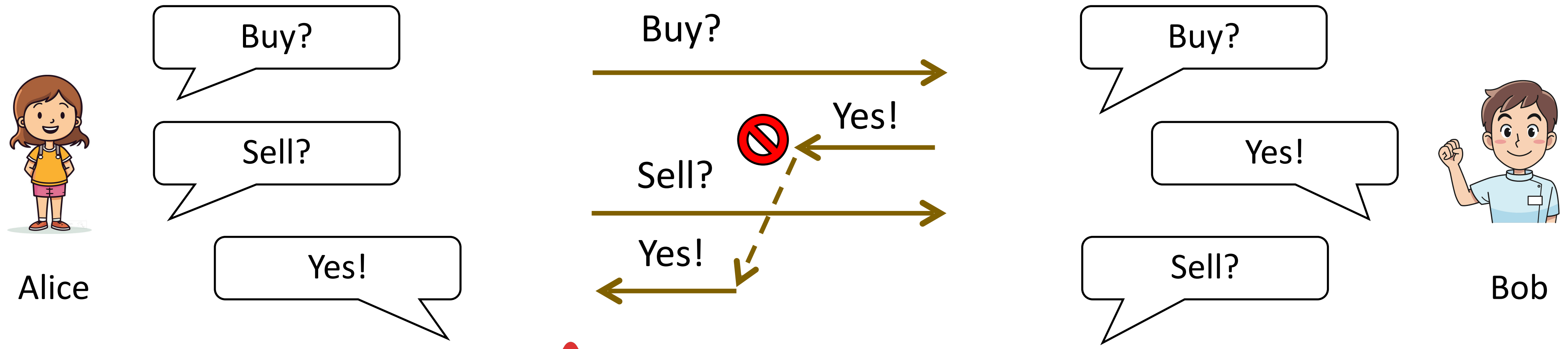
Secure (End-to-End) Messaging: Causality



Alice sells her stocks

Bob recommends buy

Secure (End-to-End) Messaging: Causality



Buy?

Sell?

Yes!

Alice

Buy?

Yes!

Sell?

Yes!

Buy?

Yes!

Sell?

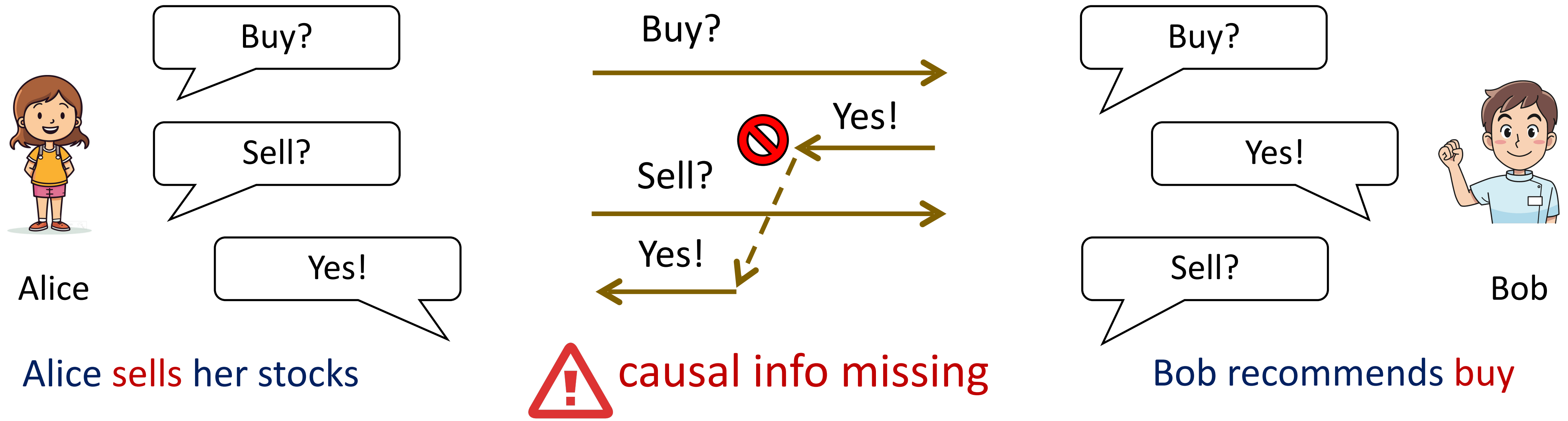
Bob

Alice sells her stocks

⚠ causal info missing

Bob recommends buy

Secure (End-to-End) Messaging: Causality



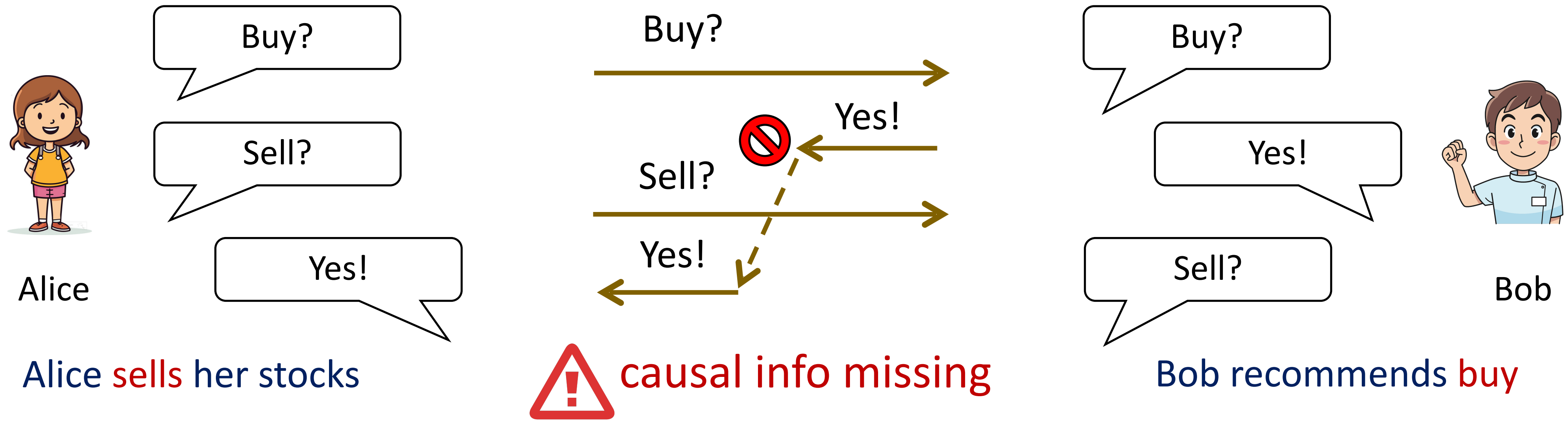
Alice sells her stocks

causal info missing

Bob recommends buy

- Causality confusion can still occur when **integrity** and **weak causality** hold:
  - Integrity: all sent and received messages are displayed intactly in order
  - Weak causality: for any message all messages that causally precede it are displayed before it

Secure (End-to-End) Messaging: Causality



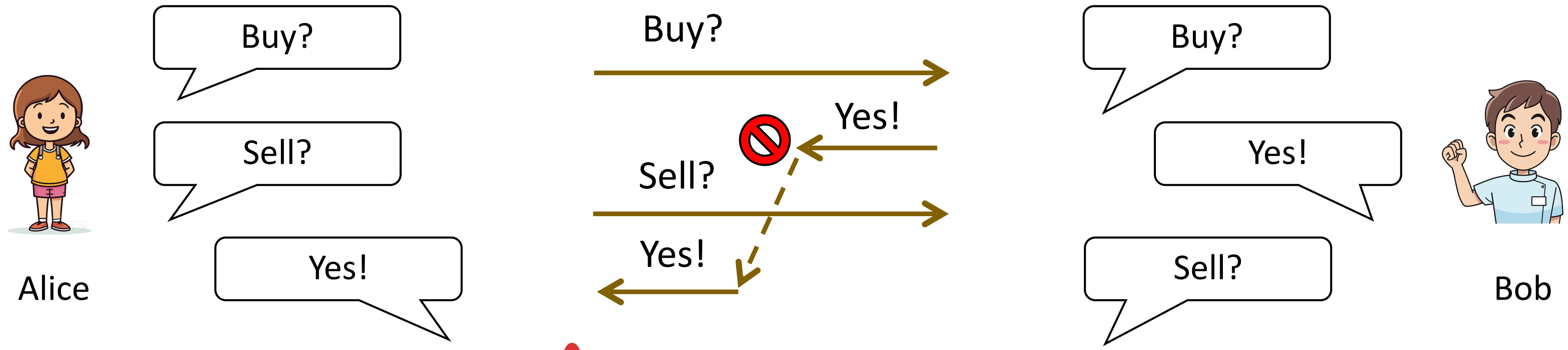
Alice sells her stocks

causal info missing

Bob recommends buy

- Causality confusion can still occur when **integrity** and **weak causality** hold:
  - Integrity: all sent and received messages are displayed intactly in order
  - Weak causality: for any message all messages that causally precede it are displayed before it
- What leads to confusion? Exact causal relations of real communication is missing.
  - E.g., Alice does not know “Yes!” is a response to “Buy?”.

Secure (End-to-End) Messaging: Causality



Buy?

Sell?

Yes!

Alice

Buy?

Yes!

Sell?

Yes!

Buy?

Yes!

Sell?

Bob

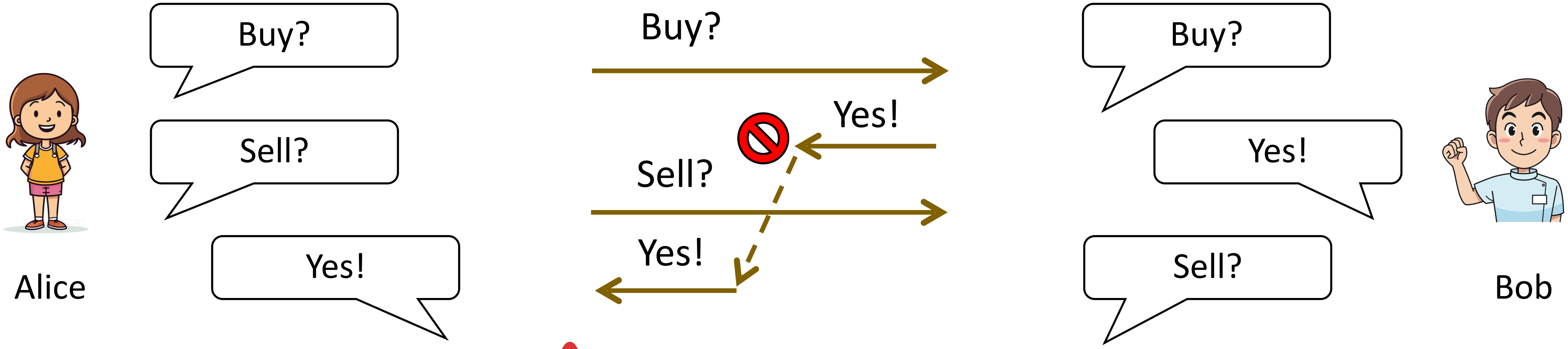
Alice sells her stocks

⚠ causal info missing

Bob recommends buy

- Cannot be easily resolved by the “reply-to” feature:
  - Bob’s view is unambiguous, so he does not know he should “reply-to” message “Buy?”.
  - Requiring users to “reply-to” every message is awkward for usability.
  - Usually “reply-to” does not support a message that depends on multiple messages.

Secure (End-to-End) Messaging: Causality



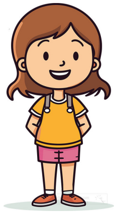
Alice sells her stocks

causal info missing

Bob recommends buy

- Cannot be easily resolved by the “reply-to” feature:
  - Bob’s view is unambiguous, so he does not know he should “reply-to” message “Buy?”.
  - Requiring users to “reply-to” every message is awkward for usability.
  - Usually “reply-to” does not support a message that depends on multiple messages.
- Would be nice if the messaging channel can provide missing causal info to users.

Secure Abuse Reporting: Causality

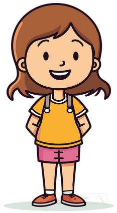


Alice

Bob

Secure Abuse Reporting: Causality

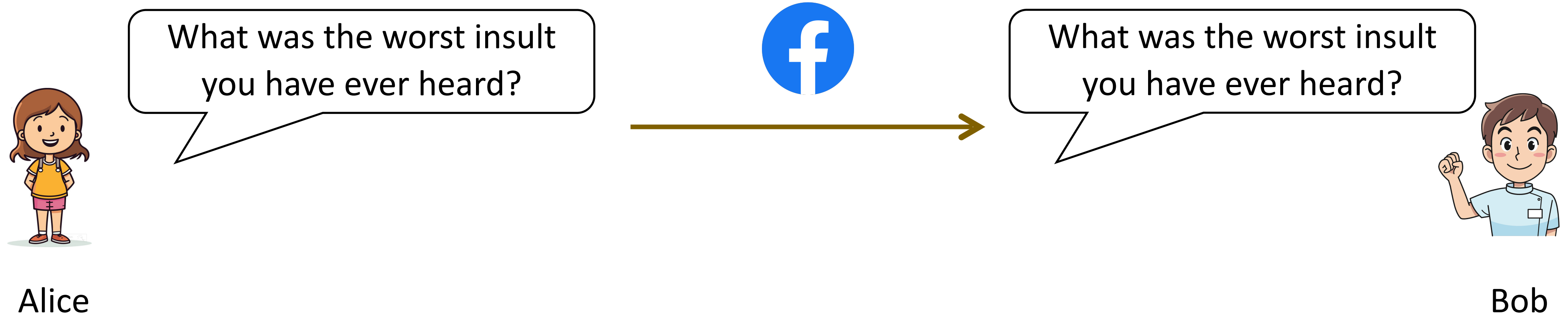
What was the worst insult you have ever heard?



Alice

Bob

Secure Abuse Reporting: Causality



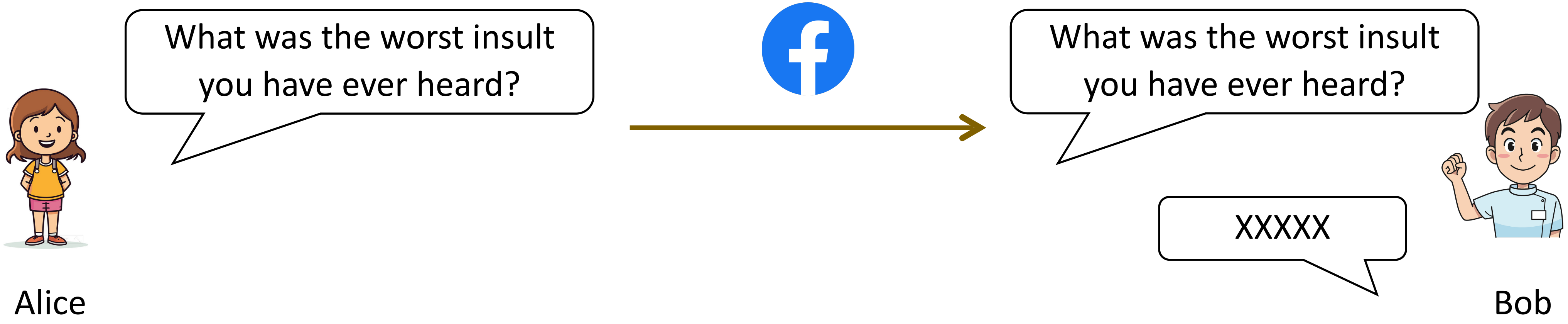
What was the worst insult you have ever heard?

What was the worst insult you have ever heard?

Alice

Bob

Secure Abuse Reporting: Causality



What was the worst insult you have ever heard?

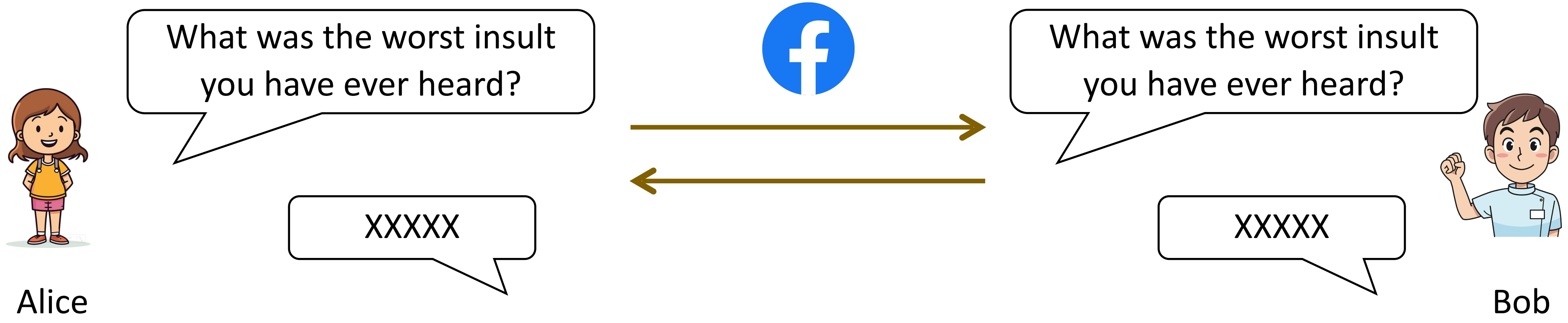
What was the worst insult you have ever heard?

XXXXX

Alice

Bob

Secure Abuse Reporting: Causality



What was the worst insult you have ever heard?

XXXXX

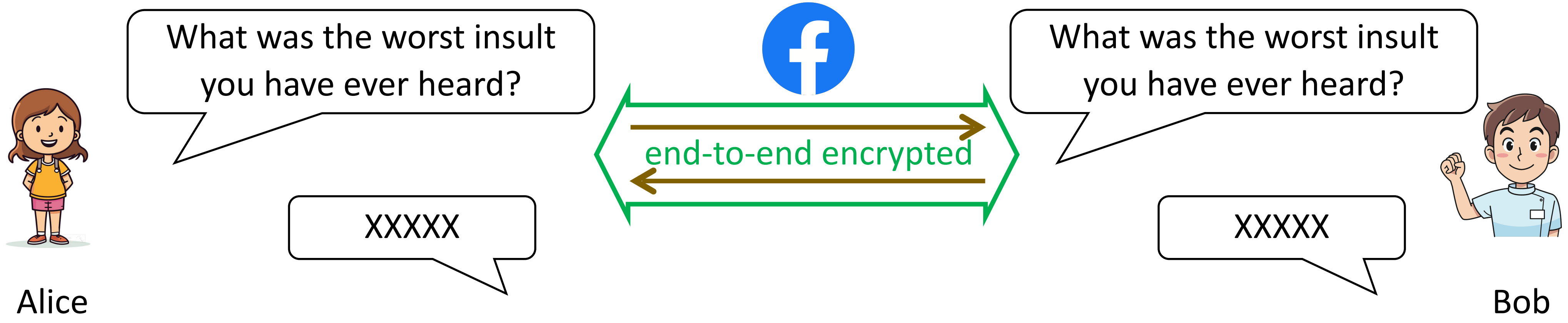
What was the worst insult you have ever heard?

XXXXX

Alice

Bob

Secure Abuse Reporting: Causality



What was the worst insult you have ever heard?

XXXXX

end-to-end encrypted

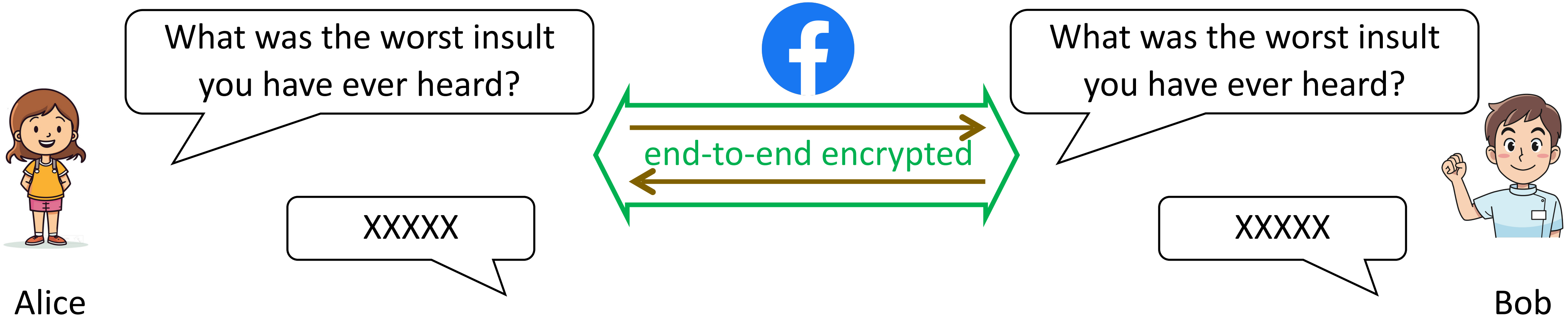
What was the worst insult you have ever heard?

XXXXX

Alice

Bob

Secure Abuse Reporting: Causality



What was the worst insult you have ever heard?

XXXXX

Alice

end-to-end encrypted

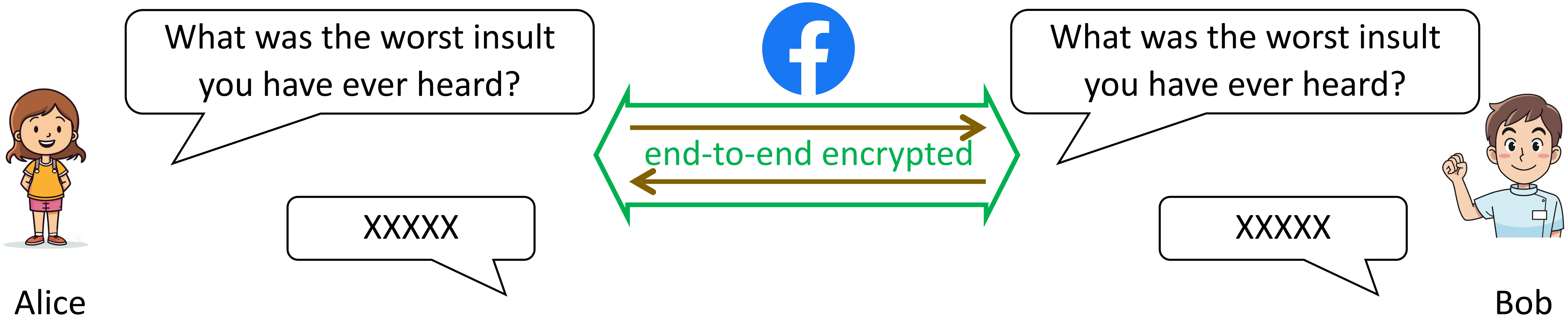
What was the worst insult you have ever heard?

XXXXX

Bob

Alice reports XXXXX as abusive

Secure Abuse Reporting: Causality



What was the worst insult you have ever heard?

XXXXX

Alice

end-to-end encrypted

What was the worst insult you have ever heard?

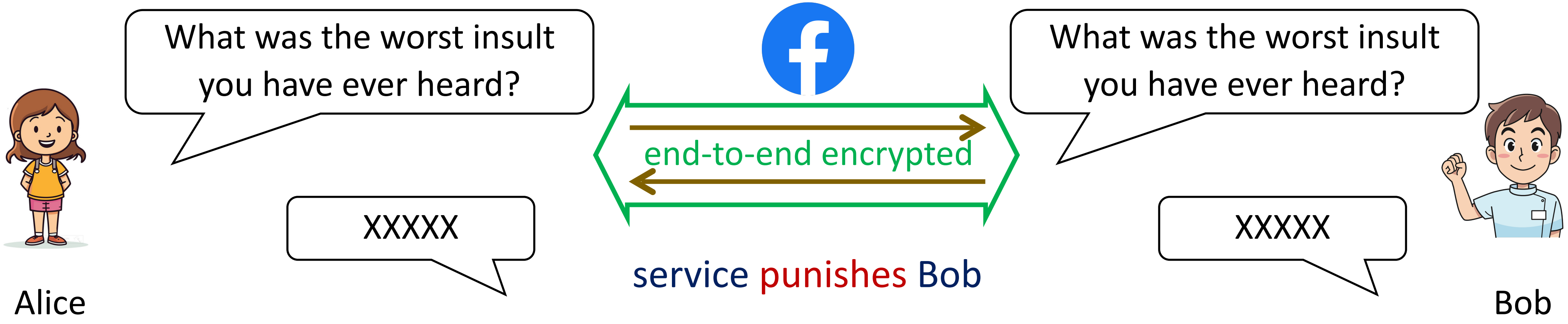
XXXXX

Bob

Alice reports XXXXX as abusive

Bob is innocent

Secure Abuse Reporting: Causality



What was the worst insult you have ever heard?

XXXXX

Alice

end-to-end encrypted

service punishes Bob

What was the worst insult you have ever heard?

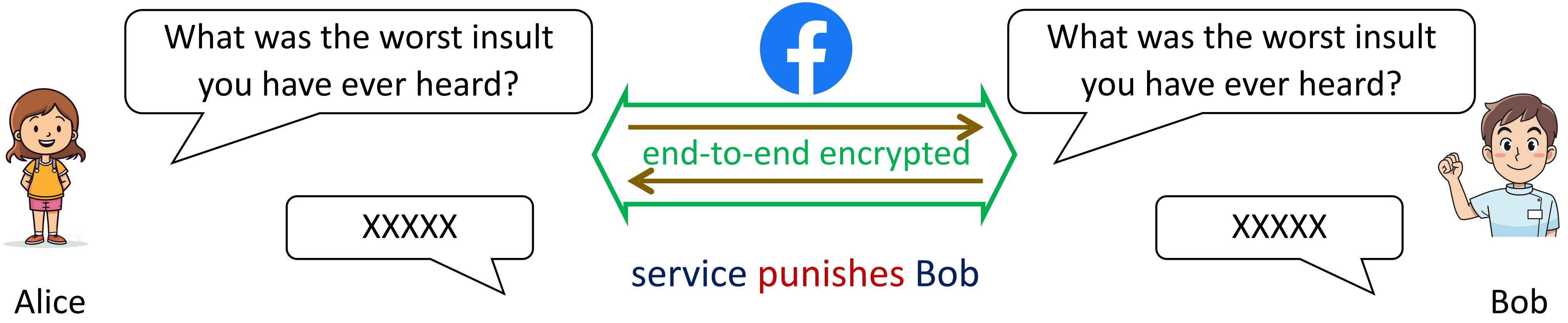
XXXXX

Bob

Alice reports XXXXX as abusive

Bob is innocent

Secure Abuse Reporting: Causality



What was the worst insult you have ever heard?

XXXXX

Alice

end-to-end encrypted

service punishes Bob

What was the worst insult you have ever heard?

XXXXX

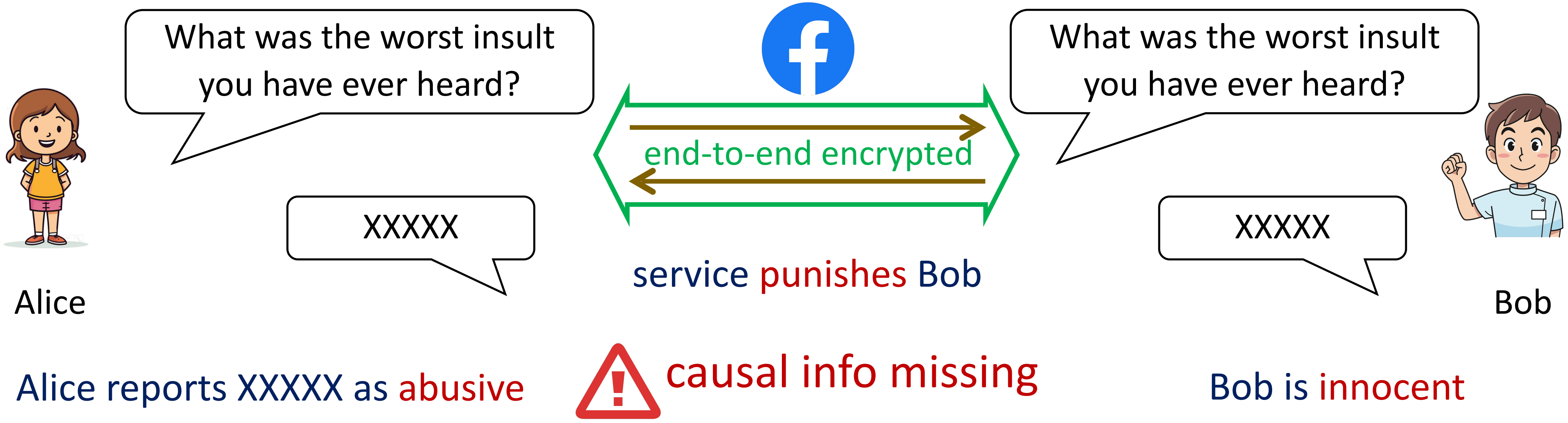
Bob

Alice reports XXXXX as abusive

⚠ causal info missing

Bob is innocent

Secure Abuse Reporting: Causality



What was the worst insult you have ever heard?

XXXXX

Alice

end-to-end encrypted

service punishes Bob

What was the worst insult you have ever heard?

XXXXX

Bob

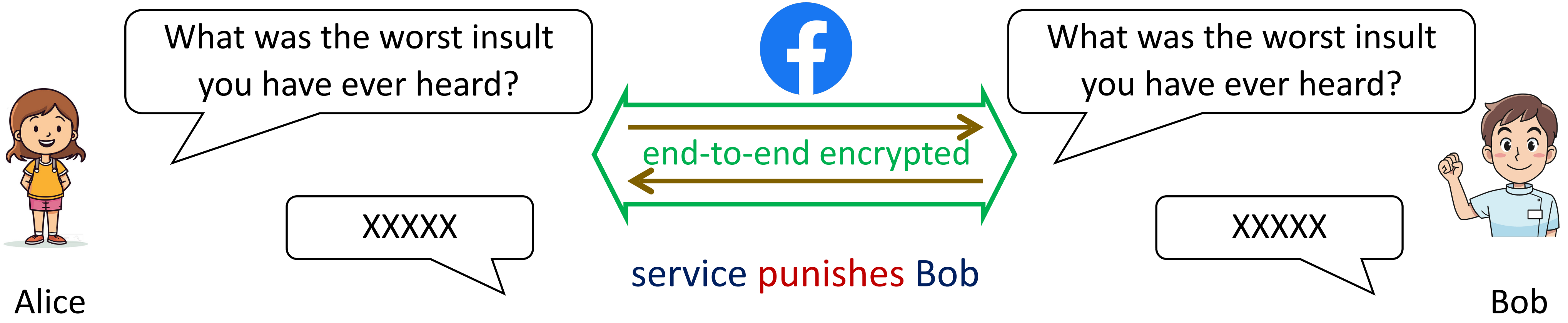
Alice reports XXXXX as abusive

causal info missing

Bob is innocent

- Cannot be easily resolved by server adding timestamps to relayed messages:
  - Timestamps indicate only time received by server, but not time sent and delivered to users.
  - E.g., if Alice’s message got delayed/lost from server, Bob’s message is indeed abusive.
  - Another motivating example is described in our paper.

Secure Abuse Reporting: Causality



What was the worst insult you have ever heard?

XXXXX

Alice

end-to-end encrypted

service punishes Bob

What was the worst insult you have ever heard?

XXXXX

Bob

Alice reports XXXXX as abusive

⚠ causal info missing

Bob is innocent

- Cannot be easily resolved by server adding timestamps to relayed messages:
  - Timestamps indicate only time received by server, but not time sent and delivered to users.
  - E.g., if Alice's message got delayed/lost from server, Bob's message is indeed abusive.
  - Another motivating example is described in our paper.
- Would be nice if the messaging channel can provide missing causal info to server.

Causality in Channels: Prior Work

- Prior work on cryptographic channels but not on causality:
  - Unidirectional primitive (e.g., stateful AEAD for TLS channel): [BKN02] [JKSS12] [BHMS16]
  - Secure-messaging channels (as bidirectional channels): [MP17] [JS18] [ACD19][BCC+23]
  - Message-franking (secure-abuse-reporting) channels: [HDL21]

Causality in Channels: Prior Work

- Prior work on cryptographic channels but not on causality:
  - Unidirectional primitive (e.g., stateful AEAD for TLS channel): [BKN02] [JKSS12] [BHMS16]
  - Secure-messaging channels (as bidirectional channels): [MP17] [JS18] [ACD19][BCC+23]
  - Message-franking (secure-abuse-reporting) channels: [HDL21]
- Prior work on formal causality analysis: [Marson17] [EMP18]
  - Causality not well defined: either implied by integrity or for weak causality only.
  - Do not apply to secure messaging: cannot handle message loss or immediate decryption.
  - No causality analysis for message franking (secure abuse reporting).

Causality in Channels: Prior Work

- Prior work on cryptographic channels but not on causality:
  - Unidirectional primitive (e.g., stateful AEAD for TLS channel): [BKN02] [JKSS12] [BHMS16]
  - Secure-messaging channels (as bidirectional channels): [MP17] [JS18] [ACD19][BCC+23]
  - Message-franking (secure-abuse-reporting) channels: [HDL21]
- Prior work on formal causality analysis: [Marson17] [EMP18]
  - Causality not well defined: either implied by integrity or for weak causality only.
  - Do not apply to secure messaging: cannot handle message loss or immediate decryption.
  - No causality analysis for message franking (secure abuse reporting).

Causality in messaging channels requires formal analysis!

Causality in Channels: Prior Work

- Prior work on cryptographic channels but not on causality:
  - Unidirectional primitive (e.g., stateful AEAD for TLS channel): [BKN02] [JKSS12] [BHMS16]
  - Secure-messaging channels (as bidirectional channels): [MP17] [JS18] [ACD19][BCC+23]
  - Message-franking (secure-abuse-reporting) channels: [HDL21]
- Prior work on formal causality analysis: [Marson17] [EMP18]
  - Causality not well defined: either implied by integrity or for weak causality only.
  - Do not apply to secure messaging: cannot handle message loss or immediate decryption.
  - No causality analysis for message franking (secure abuse reporting).

Causality in messaging channels requires formal analysis!

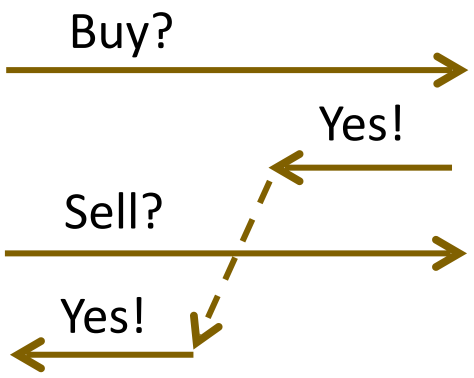
This talk will focus on causality in **secure messaging** due to time limit…

Causality Model

Causality Model: Causality Graphs

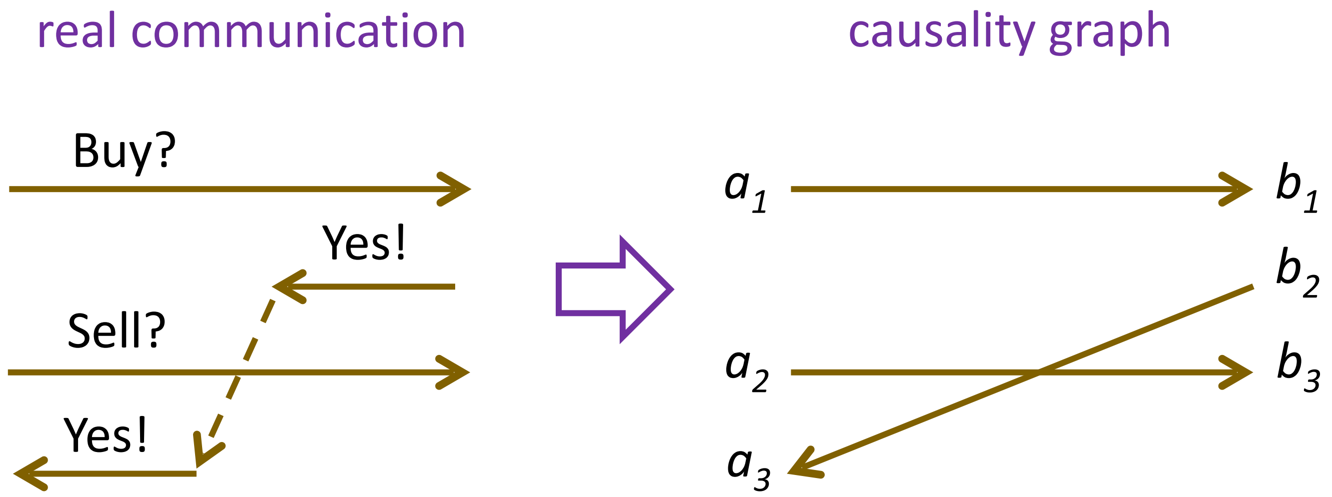
- We define **causality graphs** to model global causal info of real communication:

real communication



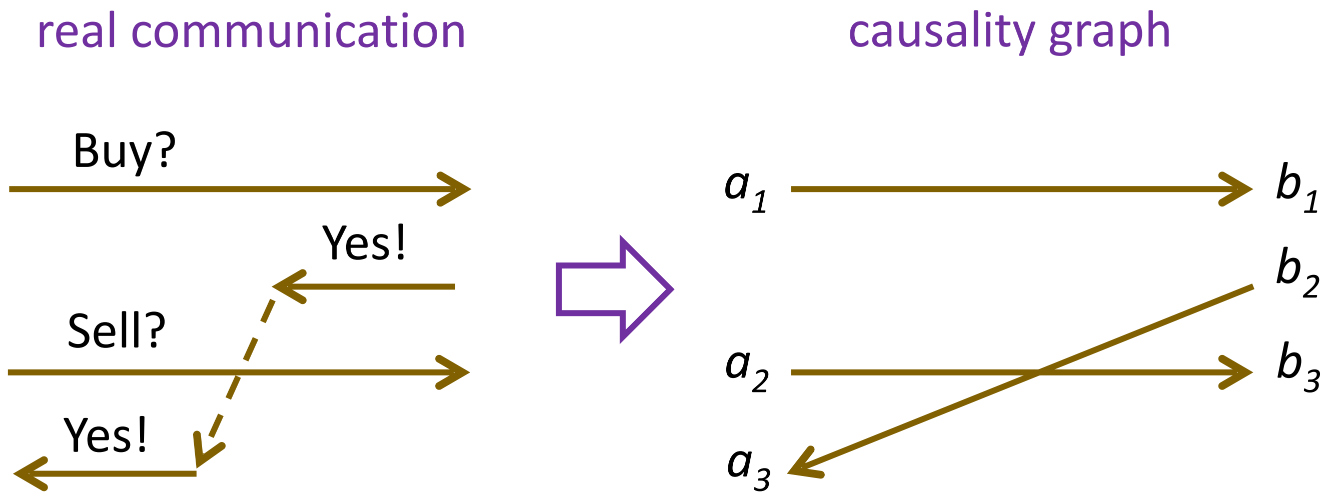
Causality Model: Causality Graphs

- We define **causality graphs** to model global causal info of real communication:



Causality Model: Causality Graphs

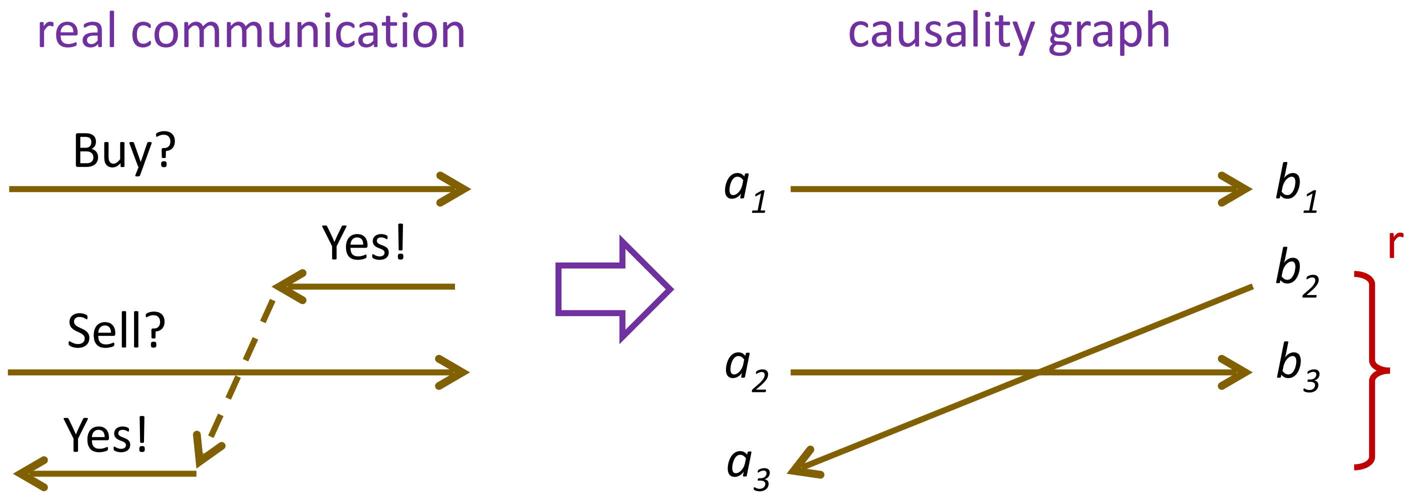
- We define **causality graphs** to model global causal info of real communication:



a_i, b_j represent sending or receiving action of a message

Causality Model: Causality Graphs

- We define **causality graphs** to model global causal info of real communication:



real communication

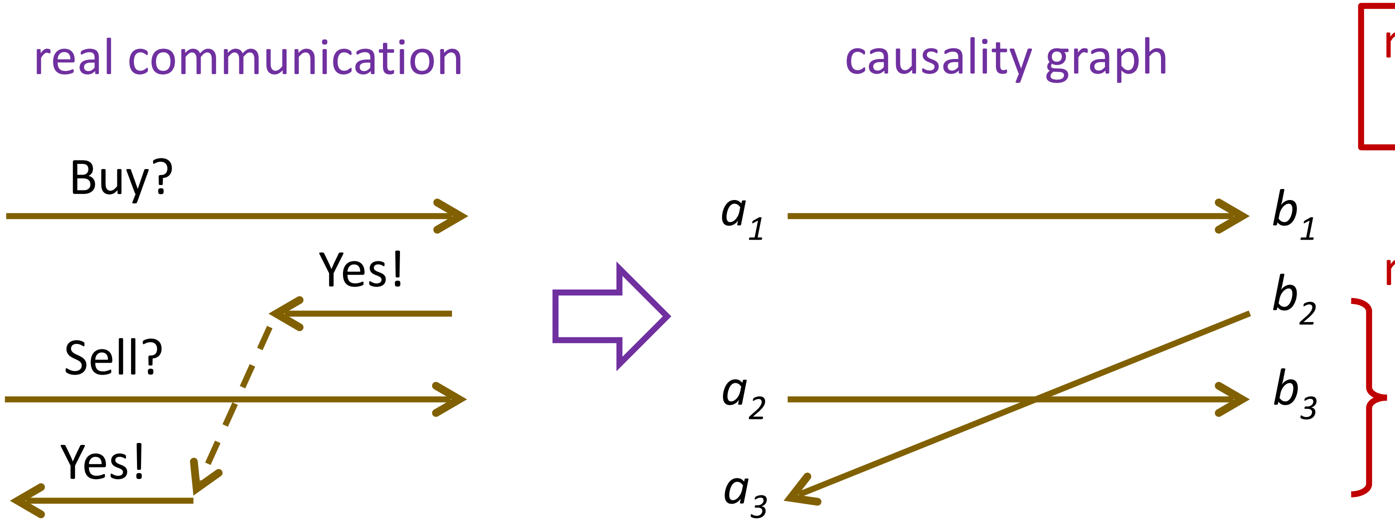
causality graph

a_i, b_j represent sending or receiving action of a message

concurrent messages that may cause confusion

Causality Model: Causality Graphs

- We define **causality graphs** to model global causal info of real communication:



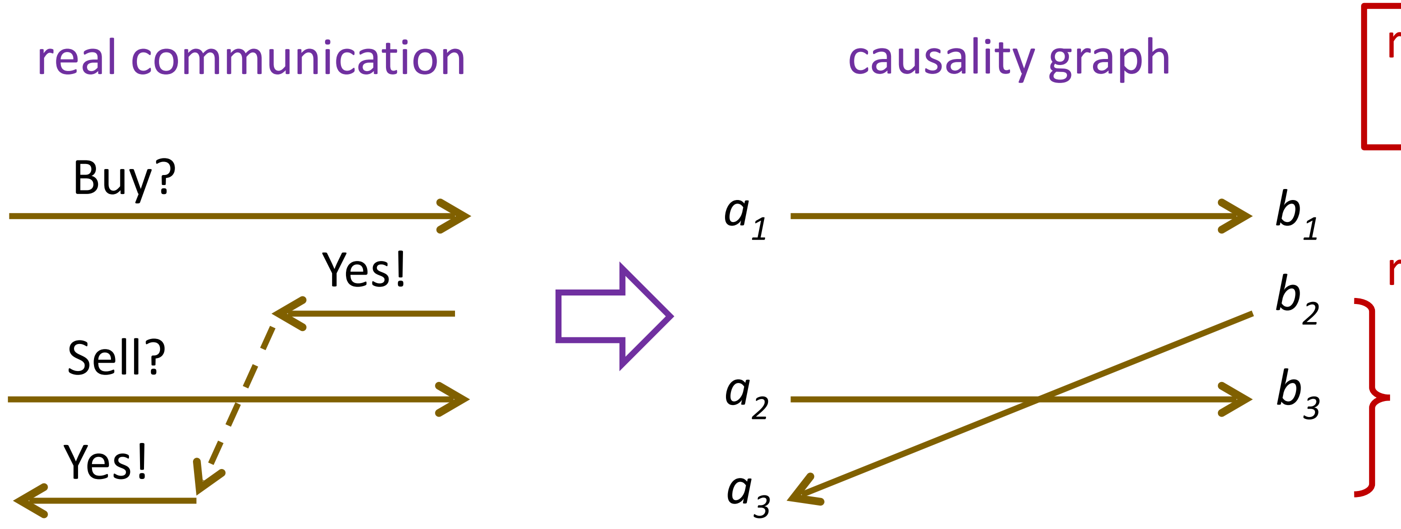
records causal dependencies
but not message contents

a_i, b_j represent sending or receiving action of a message

concurrent messages that may cause confusion

Causality Model: Causality Graphs

- We define **causality graphs** to model global causal info of real communication:



records causal dependencies
but not message contents

a_i, b_j represent sending or receiving action of a message

concurrent messages that may cause confusion

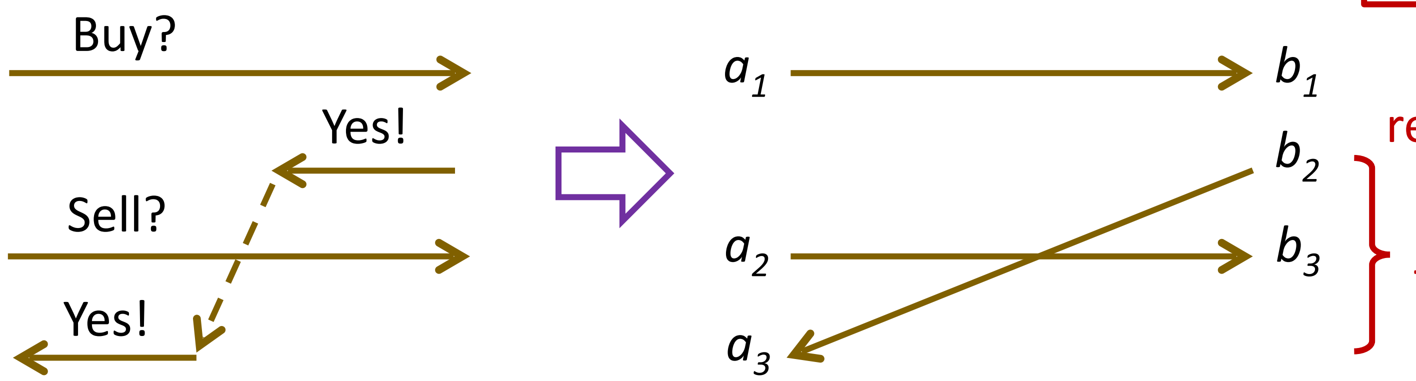
- Causality graphs capture exact causal dependencies between messages:
  - Concurrent messages (illustrated above)

Causality Model: Causality Graphs

- We define **causality graphs** to model global causal info of real communication:

real communication

causality graph



records causal dependencies **but not message contents**

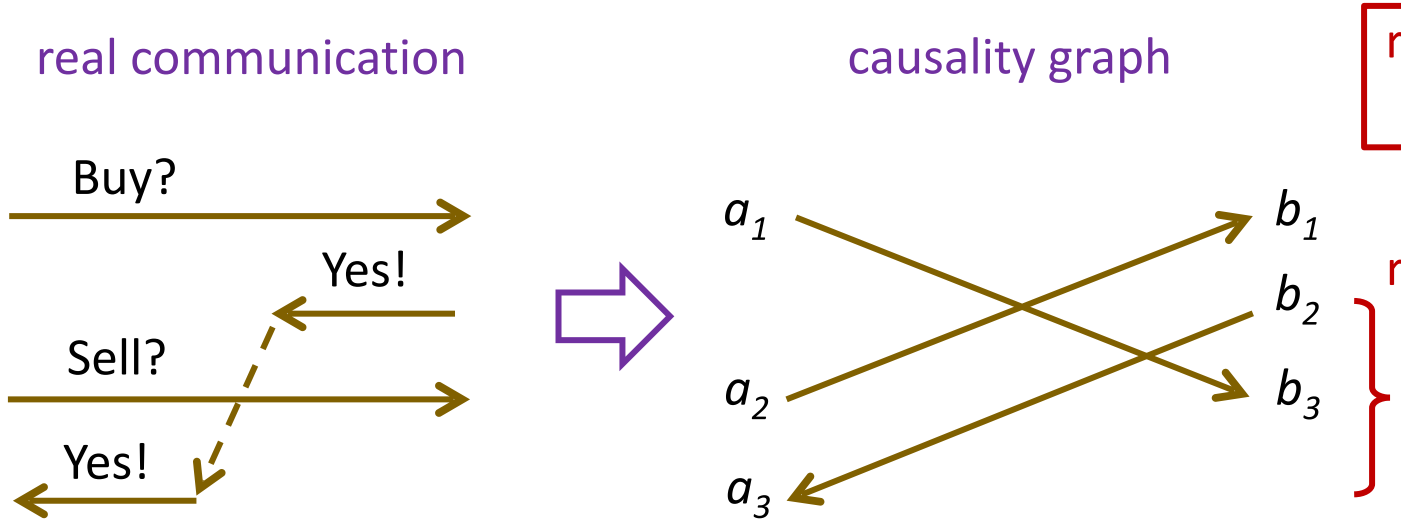
a_i, b_j represent sending or receiving action of a message

concurrent messages that may cause confusion

- Causality graphs capture exact causal dependencies between messages:
  - Concurrent messages (illustrated above)
  - Out-of-order delivery (e.g., message sent from a_1 delivered later than message sent from a_2)

Causality Model: Causality Graphs

- We define **causality graphs** to model global causal info of real communication:



records causal dependencies
but not message contents

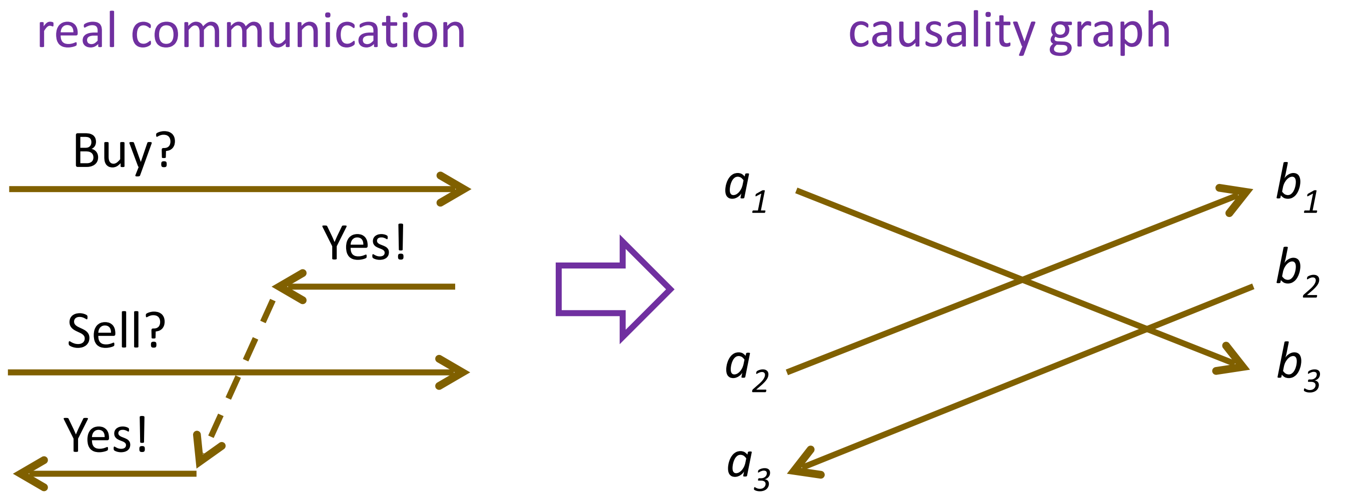
a_i, b_j represent sending or receiving action of a message

concurrent messages that may cause confusion

- Causality graphs capture exact causal dependencies between messages:
  - Concurrent messages (illustrated above)
  - Out-of-order delivery (e.g., message sent from a_1 delivered later than message sent from a_2)

Causality Model: Causality Graphs

- We define **causality graphs** to model global causal info of real communication:



records causal dependencies
but not message contents

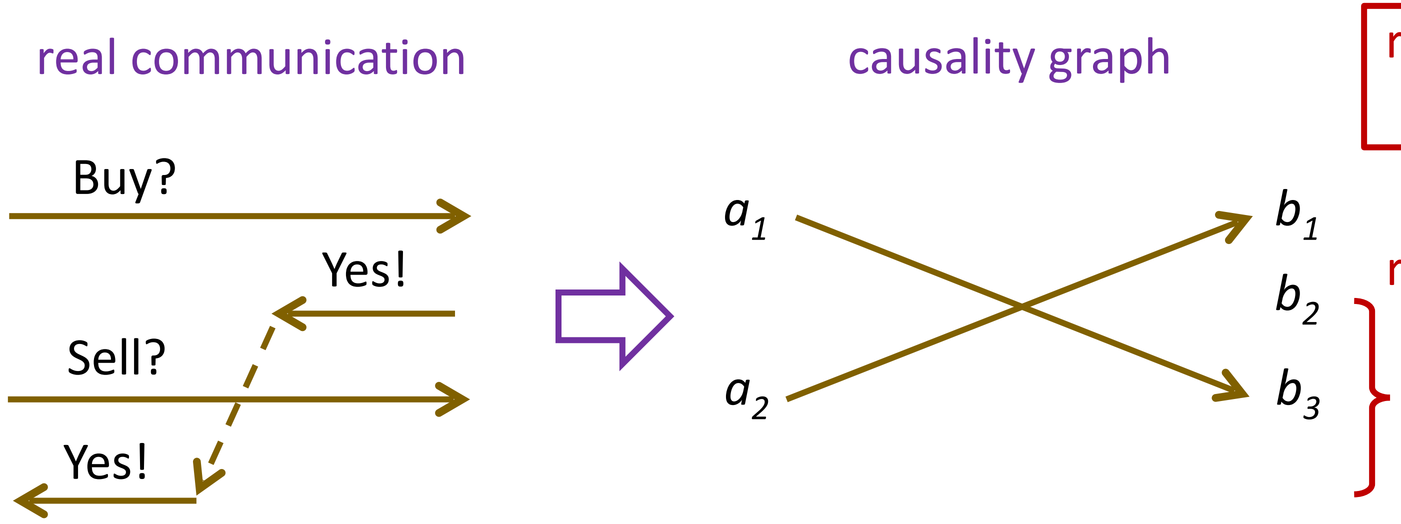
a_i, b_j represent sending or receiving action of a message

concurrent messages that may cause confusion

- Causality graphs capture exact causal dependencies between messages:
  - Concurrent messages (illustrated above)
  - Out-of-order delivery (e.g., message sent from a_1 delivered later than message sent from a_2)
  - Message loss (e.g., message sent from b_2 never delivered)

Causality Model: Causality Graphs

- We define **causality graphs** to model global causal info of real communication:



records causal dependencies
but not message contents

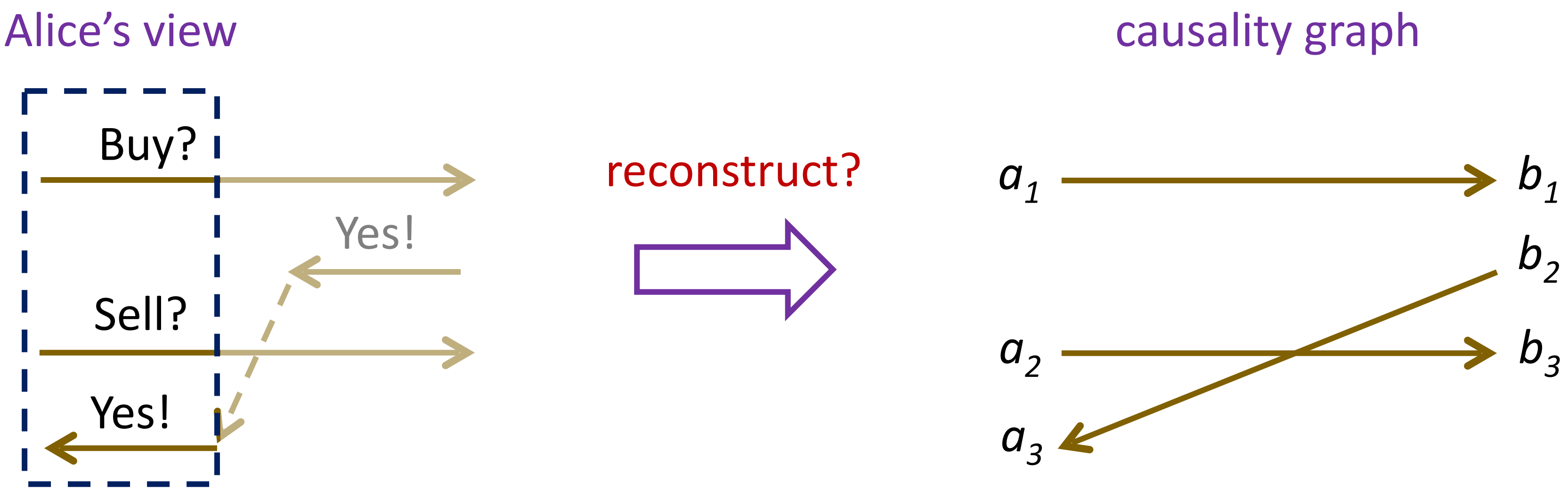
a_i, b_j represent sending or receiving action of a message

concurrent messages that may cause confusion

- Causality graphs capture exact causal dependencies between messages:
  - Concurrent messages (illustrated above)
  - Out-of-order delivery (e.g., message sent from a_1 delivered later than message sent from a_2)
  - Message loss (e.g., message sent from b_2 never delivered)

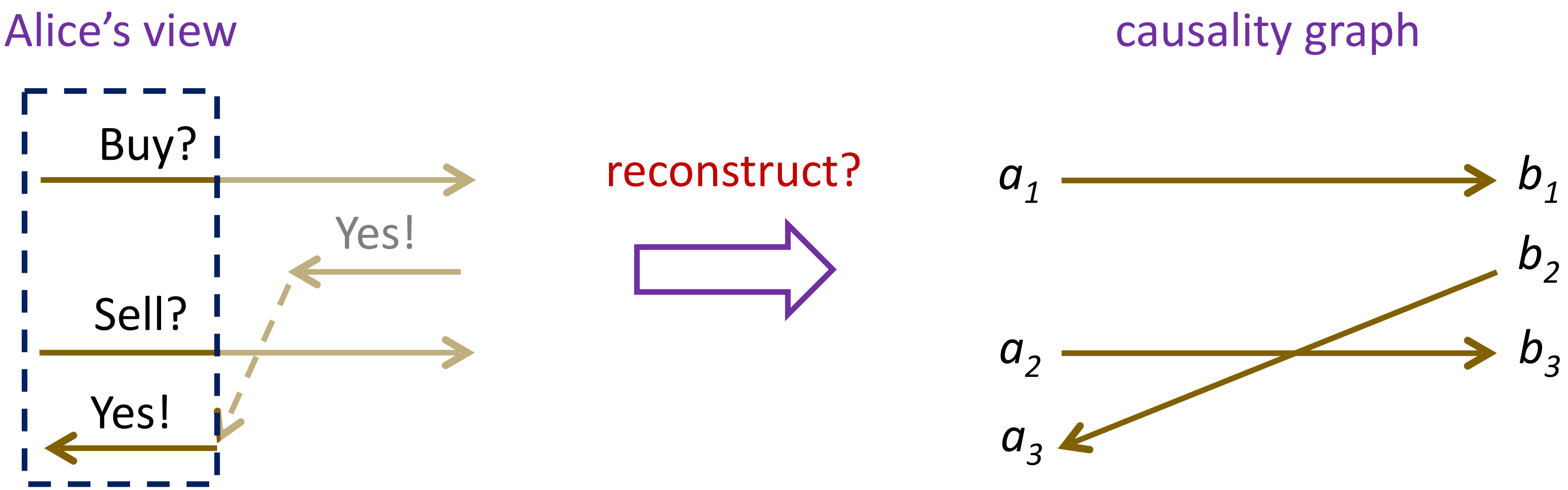
Causality Model: Security Definitions

- **Causality Preservation (CP):** Users able to locally reconstruct global causal info



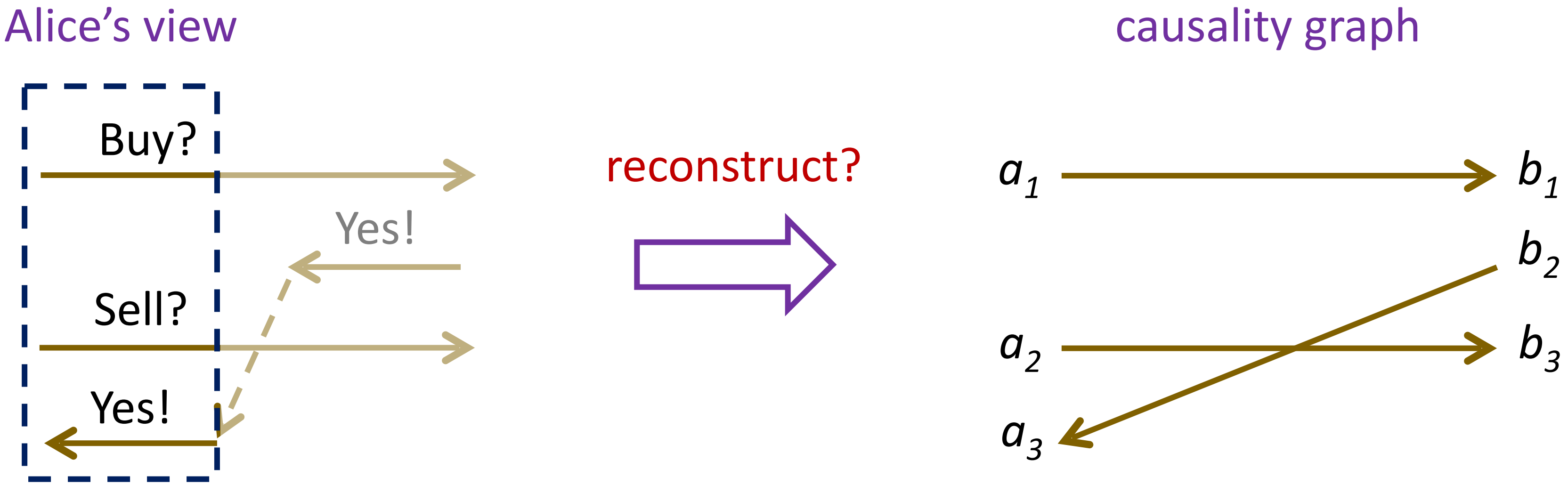
Causality Model: Security Definitions

- **Causality Preservation (CP):** Users able to locally reconstruct global causal info
  - **CP $\Rightarrow$ INT-PTXT** (plaintext integrity): can trivially manipulate causality by altering messages
  - **CP $\nLeftarrow$ INT-CTXT** (ciphertext integrity): integrity does not ensure users getting all causal info



Causality Model: Security Definitions

- **Causality Preservation (CP):** Users able to locally reconstruct global causal info
  - **CP ⇒ INT-PTXT** (plaintext integrity): can trivially manipulate causality by altering messages
  - **CP ⇍ INT-CTXT** (ciphertext integrity): integrity does not ensure users getting all causal info
- **Strong Causality Preservation (SCP): CP + post-compromise security (PCS)**
  - **PCS** is important for secure messaging: can recover security after long-term key is corrupted



Alice's view

Buy?

Yes!

Sell?

Yes!

reconstruct?

causality graph

a_1

b_1

b_2

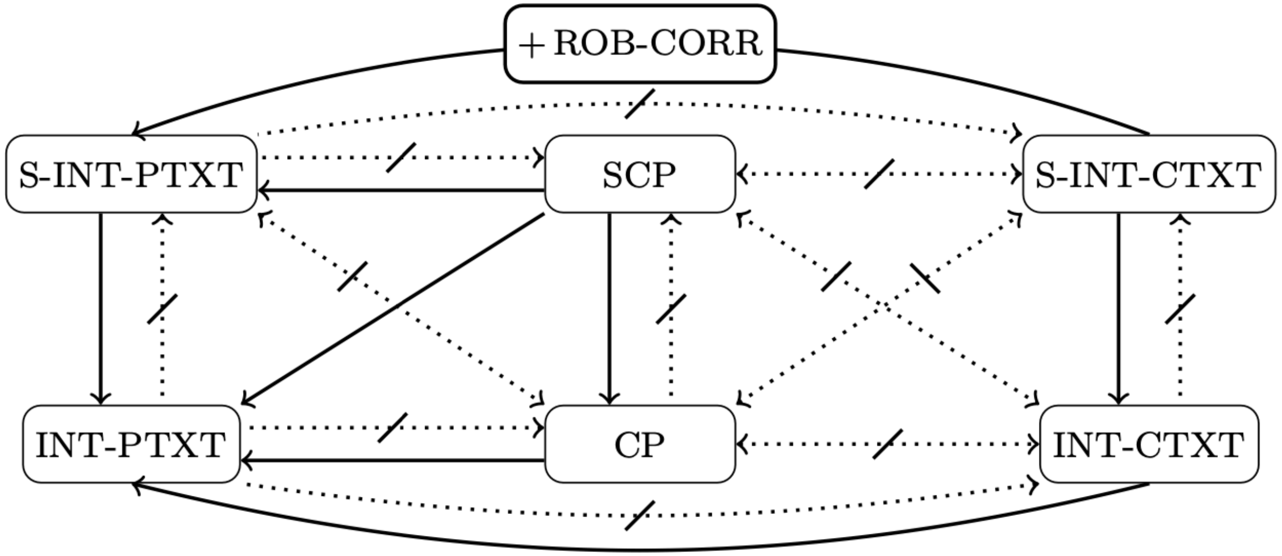
a_2

b_3

a_3

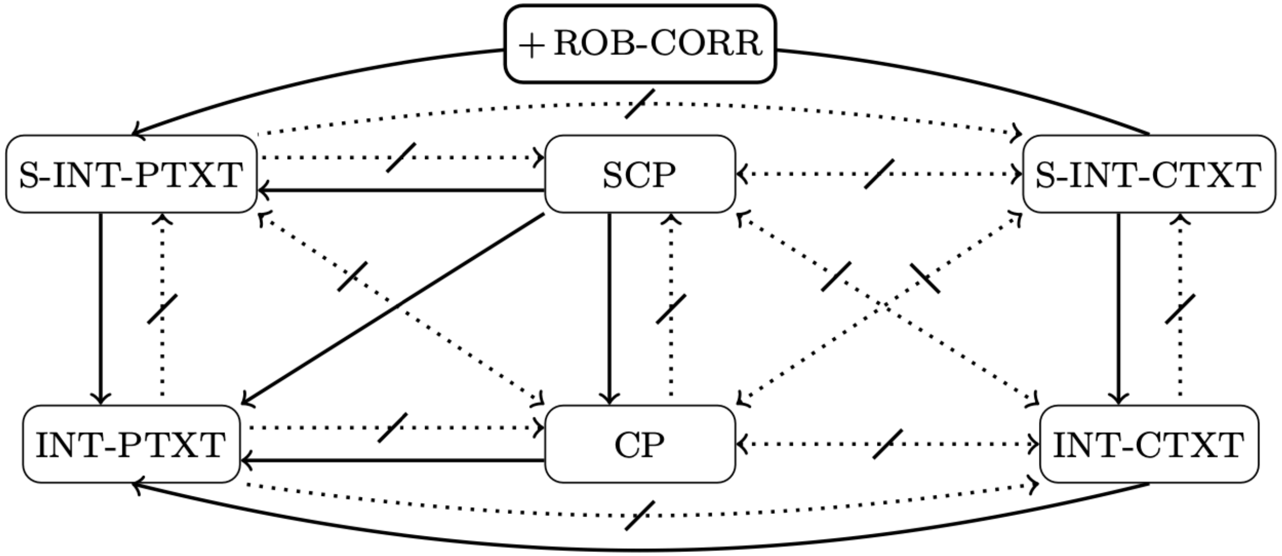
Causality Model: Security Definitions

- Relations to integrity notions (leading “S-” refers to post-compromise security):



Causality Model: Security Definitions

- Relations to integrity notions (leading “S-” refers to post-compromise security):

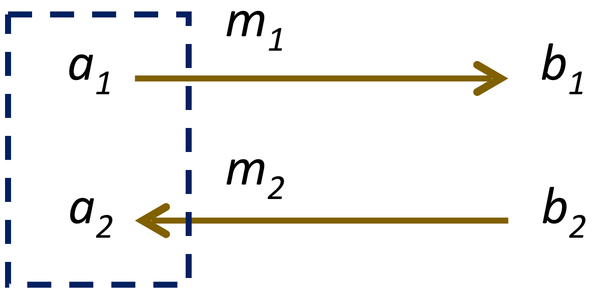


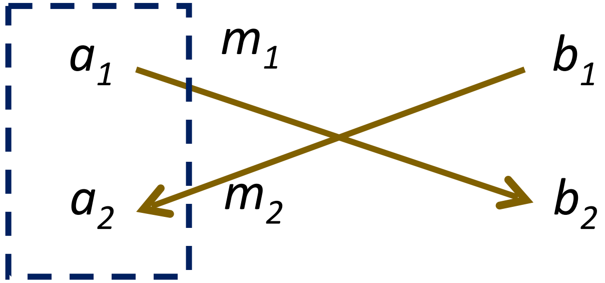
- We have complete separation results between causality notions **(S)CP** and ciphertext integrity notions **(S-)INT-CTXT** as expected.

Analysis of Real-World Protocols

Causality Preservation of TLS 1.3

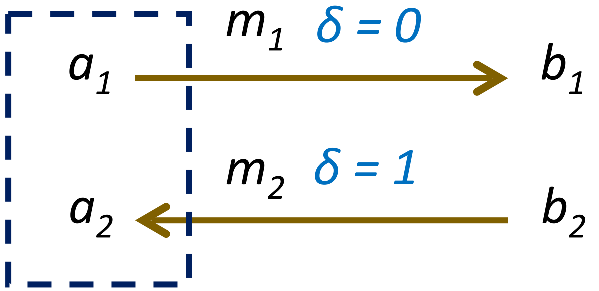
- TLS 1.3 does not preserve causality:
  - E.g., left user's view is identical in both cases, so cannot reconstruct the correct causal info

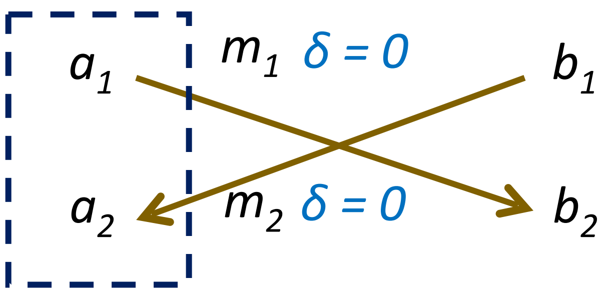




Causality Preservation of TLS 1.3

- TLS 1.3 does not preserve causality:
  - E.g., left user's view is identical in both cases, so cannot reconstruct the correct causal info

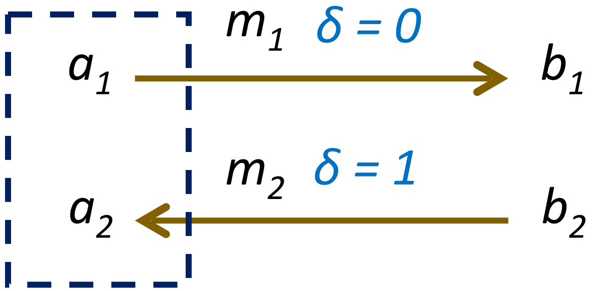


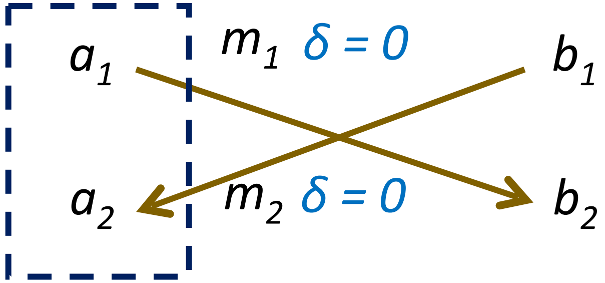


- Solution: Add necessary causal info (authenticated) along with sent messages.
  - δ: the number of consecutively received messages before the sent message [Marson17]

Causality Preservation of TLS 1.3

- TLS 1.3 does not preserve causality:
  - E.g., left user's view is identical in both cases, so cannot reconstruct the correct causal info

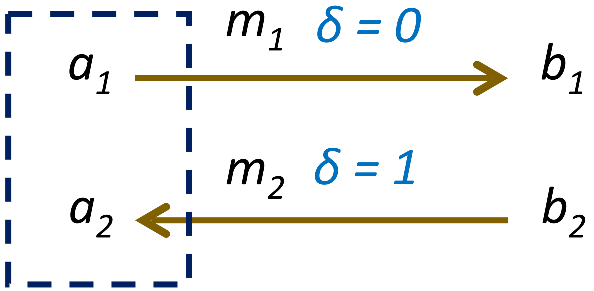


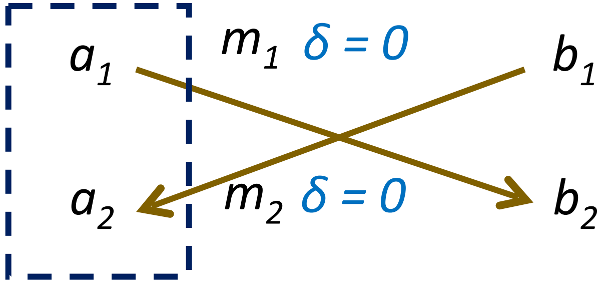


- Solution: Add necessary causal info (authenticated) along with sent messages.
  - δ: the number of consecutively received messages before the sent message [Marson17]
  - Note: This simple idea works because TLS 1.3 runs on a reliable in-order network, i.e., all sent messages are assumed to be reliably delivered and in order.

Causality Preservation of TLS 1.3

- TLS 1.3 does not preserve causality:
  - E.g., left user's view is identical in both cases, so cannot reconstruct the correct causal info





- Solution: Add necessary causal info (authenticated) along with sent messages.
  - δ: the number of consecutively received messages before the sent message [Marson17]
  - Note: This simple idea works because TLS 1.3 runs on a reliable in-order network, i.e., all sent messages are assumed to be reliably delivered and in order.
- We formally prove that this fixed so-called causal TLS 1.3 channel is **CP-secure**.
  - Very simple fix, interesting to find practical use cases for causal TLS 1.3 channel.

Causality Preservation of Signal

- Signal does not preserve causality either (for reasons similar to TLS 1.3).

Causality Preservation of Signal

- Signal does not preserve causality either (for reasons similar to TLS 1.3).

- Solution: Add necessary causal info (authenticated) along with sent messages.
  - Need to consider out-of-order message delivery and message loss.
  - *Q*: a queue that records entire causal info (i.e., causality graph with no message content) before the sent message
  - Can reduce *Q* size by recording only the causal info not yet confirmed by the receiver.

Causality Preservation of Signal

- Signal does not preserve causality either (for reasons similar to TLS 1.3).

- Solution: Add necessary causal info (authenticated) along with sent messages.
  - Need to consider out-of-order message delivery and message loss.
  - *Q*: a queue that records entire causal info (i.e., causality graph with no message content) before the sent message
  - Can reduce *Q* size by recording only the causal info not yet confirmed by the receiver.
- We formally prove that this fixed so-called causal Signal channel is **SCP-secure**.
  - Example messaging app UI with causality: “press-and-hold” highlights causal dependencies.

Causality Preservation of Signal

- Signal does not preserve causality either (for reasons similar to TLS 1.3).

- Solution: Add necessary causal info (authenticated) along with sent messages.
  - Need to consider out-of-order message delivery and message loss.
  - *Q*: a queue that records entire causal info (i.e., causality graph with no message content) before the sent message
  - Can reduce *Q* size by recording only the causal info not yet confirmed by the receiver.
- We formally prove that this fixed so-called causal Signal channel is **SCP-secure**.
  - Example messaging app UI with causality: "press-and-hold" highlights causal dependencies.

- Our fix is generic: can apply to any secure-messaging channel to gain SCP security.
  - Assume underlying channel achieves ciphertext integrity with post-compromise security.

Summary

Summary

- Integrating causality in messaging channels:
    - Define causality model with desired security for secure messaging (as bidirectional channel).
    - Propose provable secure fix to add causality to TLS 1.3 and Signal.
    - Our causality fix is generic and can be applied to any secure-messaging channels.

Summary

- Integrating causality in messaging channels:
  - Define causality model with desired security for secure messaging (as bidirectional channel).
  - Propose provable secure fix to add causality to TLS 1.3 and Signal.
  - Our causality fix is generic and can be applied to any secure-messaging channels.
- Integrating causality in messaging-franking channels (see our paper):
  - Define causality model with desired security for message franking.
  - Propose provable secure fix to add causality to Facebook’s message franking.

Summary

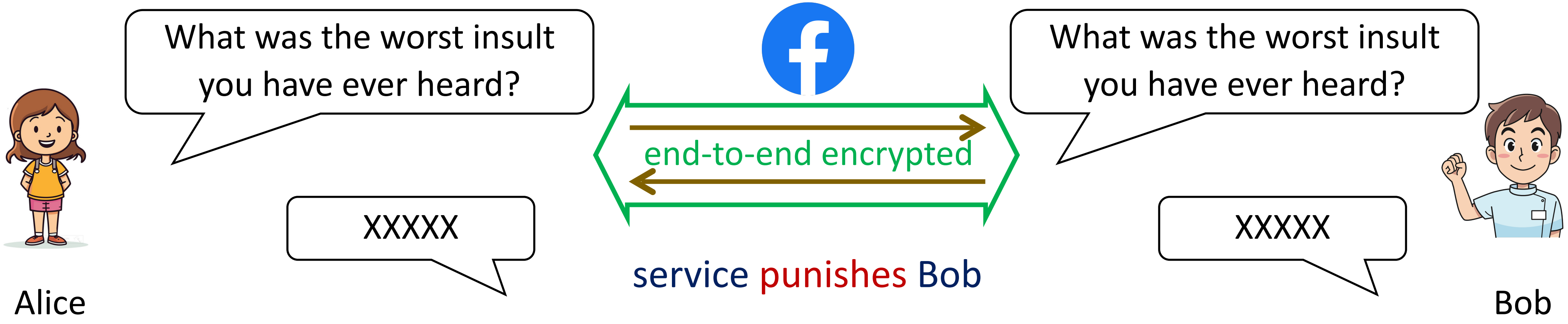
- Integrating causality in messaging channels:
  - Define causality model with desired security for secure messaging (as bidirectional channel).
  - Propose provable secure fix to add causality to TLS 1.3 and Signal.
  - Our causality fix is generic and can be applied to any secure-messaging channels.
- Integrating causality in messaging-franking channels (see our paper):
  - Define causality model with desired security for message franking.
  - Propose provable secure fix to add causality to Facebook's message franking.
- Future work:
  - Investigate how causality can be better visualized for users.
  - Lower bound on the overhead for messaging channels to preserve causality.
  - Extend causality preservation to secure group messaging.

Summary

- Integrating causality in messaging channels:
  - Define causality model with desired security for secure messaging (as bidirectional channel).
  - Propose provable secure fix to add causality to TLS 1.3 and Signal.
  - Our causality fix is generic and can be applied to any secure-messaging channels.
- Integrating causality in messaging-franking channels (see our paper):
  - Define causality model with desired security for message franking.
  - Propose provable secure fix to add causality to Facebook’s message franking.
- Future work:
  - Investigate how causality can be better visualized for users.
  - Lower bound on the overhead for messaging channels to preserve causality.
  - Extend causality preservation to secure group messaging.
- Thanks! Questions? (Our paper: https://eprint.iacr.org/2024/362)

Additional Slides on Message Franking

Recall: Causality in Message Franking



What was the worst insult you have ever heard?

XXXXX

Alice

end-to-end encrypted

service punishes Bob

What was the worst insult you have ever heard?

XXXXX

Bob

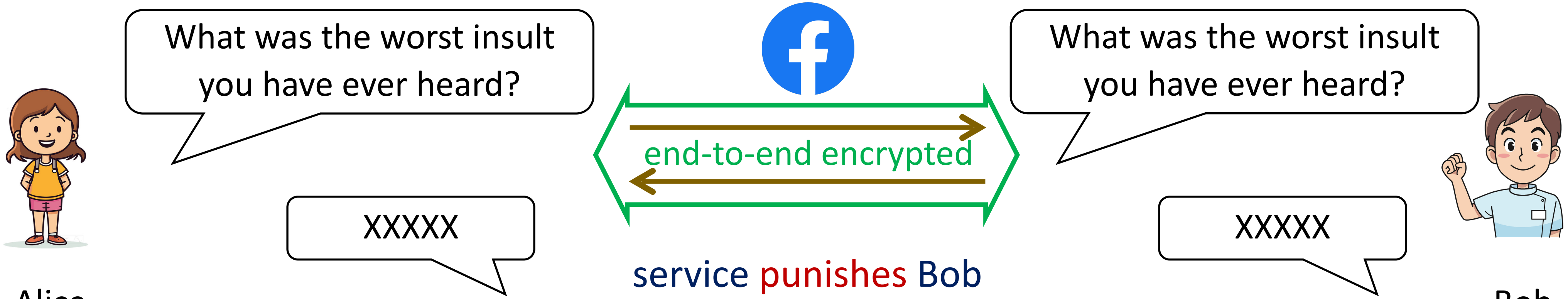
Alice reports XXXXX as abusive

⚠ causal info missing

Bob is innocent

- Would be nice if the messaging channel can provide missing causal info to server.

Recall: Causality in Message Franking



What was the worst insult you have ever heard?

end-to-end encrypted

What was the worst insult you have ever heard?

XXXXX

service punishes Bob

XXXXX

Alice

Bob

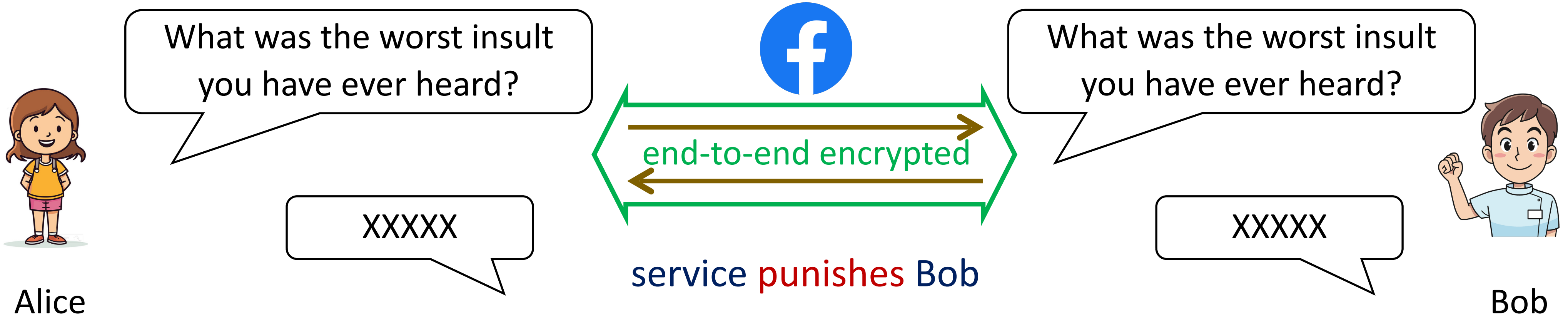
Alice reports XXXXX as abusive

⚠ causal info missing

Bob is innocent

- Would be nice if the messaging channel can provide missing causal info to server.
- Message-franking channel = secure-messaging channel + abuse reporting scheme

Recall: Causality in Message Franking


What was the worst insult you have ever heard?
XXXXX
Alice
end-to-end encrypted
service punishes Bob
What was the worst insult you have ever heard?
XXXXX
Bob

Alice reports XXXXX as abusive

⚠ causal info missing

Bob is innocent

- Would be nice if the messaging channel can provide missing causal info to server.
- Message-franking channel = secure-messaging channel + abuse reporting scheme
- Causality security guarantees for message franking channel is actually two-fold: one for end users in secure messaging and one for server in abuse reporting.

Causality Model: Security Definitions

- **Channel Causality Preservation (CCP)** for underlying secure-messaging channel:
  - **CP** and **SCP** same as before but extended to the syntax of message-franking channels.

malicious server against honest users

- **Report Causality Preservation (RCP)** for message franking (abuse reporting):
  - **RCP-S:** malicious sender cannot make the other user accept unreportable messages
  - **RCP-R:** malicious reporter (receiver) cannot report a message never sent by the other user
    + malicious reporter cannot report message with incorrect or insufficient causal info

malicious users against honest server

Causality Preservation of FB Message Franking

- Facebook’s message franking scheme does not preserve causality:
  - Not CCP-secure: underlying secure-messaging channel Signal is not CP-secure.
  - Not RCP-secure: server receives isolated reported message but not its causal dependencies.

Causality Preservation of FB Message Franking

- Facebook’s message franking scheme does not preserve causality:
  - Not CCP-secure: underlying secure-messaging channel Signal is not CP-secure.
  - Not RCP-secure: server receives isolated reported message but not its causal dependencies.

- Solution: Add necessary causal info (authenticated) along with sent messages.
  - Add necessary causal info recorded in queue *Q* to the underlying Signal channel as before.
  - Add this *Q* and the index of the reported message when reporting a message to server.

Causality Preservation of FB Message Franking

- Facebook’s message franking scheme does not preserve causality:
  - Not CCP-secure: underlying secure-messaging channel Signal is not CP-secure.
  - Not RCP-secure: server receives isolated reported message but not its causal dependencies.

- Solution: Add necessary causal info (authenticated) along with sent messages.
  - Add necessary causal info recorded in queue *Q* to the underlying Signal channel as before.
  - Add this *Q* and the index of the reported message when reporting a message to server.
- We formally prove that this fixed so-called causal message-franking channel is both **SCP-secure** and **RCP-secure**.

Causality Preservation of FB Message Franking

- Facebook's message franking scheme does not preserve causality:
  - Not CCP-secure: underlying secure-messaging channel Signal is not CP-secure.
  - Not RCP-secure: server receives isolated reported message but not its causal dependencies.

- Solution: Add necessary causal info (authenticated) along with sent messages.
  - Add necessary causal info recorded in queue *Q* to the underlying Signal channel as before.
  - Add this *Q* and the index of the reported message when reporting a message to server.
- We formally prove that this fixed so-called causal message-franking channel is both **SCP-secure** and **RCP-secure**.

- With our causality fix, abuse-reporting server can handle disputes in context.