Epistemic Logics for Authentication and Secrecy

1st Workshop on Logic, Language and Informationa UFF

Mario Benevides

Federal University of Rio de Janeiro - Brazil

October-2015

Overview

- Motivation

Overview

- Motivation
- Modal Logics

Overview

- Motivation
- Modal Logics
- Multi-Agent Epistemic Logic

Overview

- Motivation
- Modal Logics
- Multi-Agent Epistemic Logic
- Algebraic and Logic Approaches for Autheticity and Secrecy

Outline

- Modelo de Dolev Yao: On the Security of Public Key Protocols - 1983

Outline

- Modelo de Dolev Yao: On the Security of Public Key Protocols - 1983
- SPi Calculus: A Calculus for Cryptographic Protocols: The Spi Calculus - M. Abadi and A. Gordon - 1999

Outline

- Modelo de Dolev Yao: On the Security of Public Key Protocols - 1983
- SPi Calculus: A Calculus for Cryptographic Protocols: The Spi Calculus - M. Abadi and A. Gordon - 1999
- BAN Logic: A Logic of Authentication - M. Burrows, M. Abadi, and R. Needham - 1990

Outline

- Modelo de Dolev Yao: On the Security of Public Key Protocols - 1983
- SPi Calculus: A Calculus for Cryptographic Protocols: The Spi Calculus - M. Abadi and A. Gordon - 1999
- BAN Logic: A Logic of Authentication - M. Burrows, M. Abadi, and R. Needham - 1990
- Dolev/Yao Multi-Agent Epistemic Logic

Motivation

- Two Big Groups

Motivation

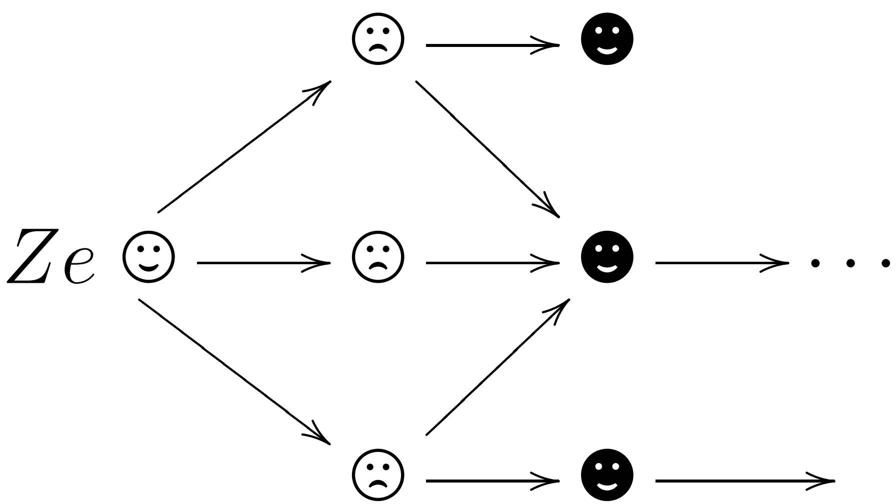
- Two Big Groups
- Logical $\Longrightarrow$ Deductions
  - BAN Logic: A Logic of Authentication - M. Burrows, M. Abadi, and R. Needham - 1990
  - Dolev/Yao Multi-Agent Epistemic Logic

Motivation

- Two Big Groups
- Logical $\Longrightarrow$ Deductions
  - BAN Logic: A Logic of Authentication - M. Burrows, M. Abadi, and R. Needham - 1990
  - Dolev/Yao Multi-Agent Epistemic Logic
- Algebraic $\Longrightarrow$ Equivalence
  - SPi Calculus: - M. Abadi and A. Gordon - 1999
  - Reconciling Two Views of Cryptography - M. Abadi and P. Rogaway - 2000

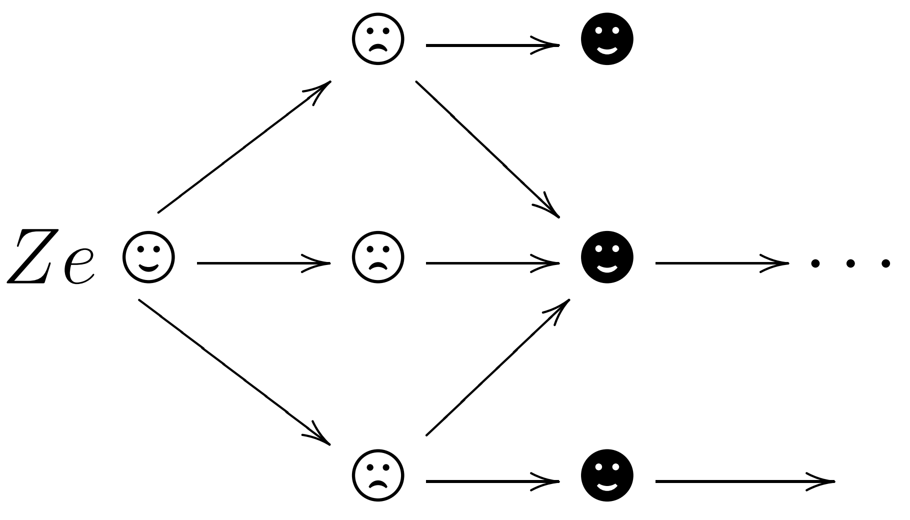
Lógicas Modais

- Teste com questões de V ou F



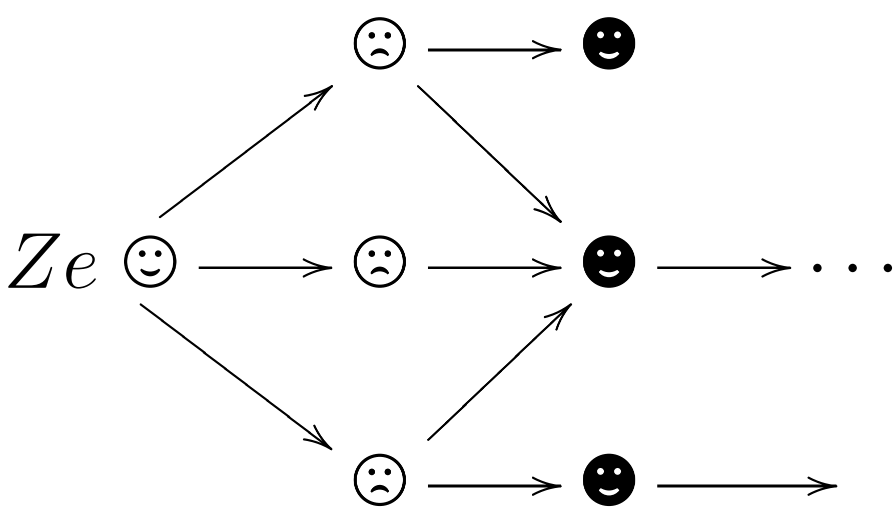
Lógicas Modais

- Teste com questões de V ou F
- aluno não estudou e faz a seguinte notação



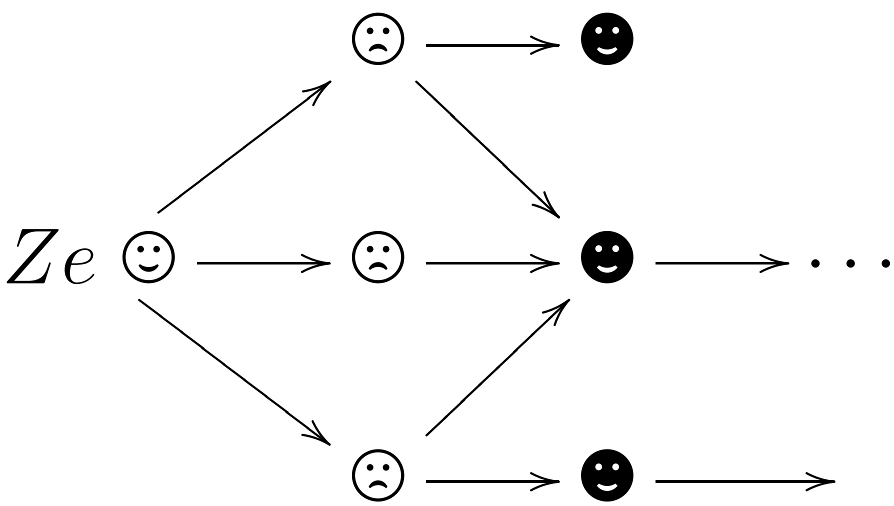
Lógicas Modais

- Teste com questões de V ou F
- aluno não estudou e faz a seguinte notação
- $\Box p$: todo mundo que eu vejo marcou p como V



Lógicas Modais

- Teste com questões de V ou F
- aluno não estudou e faz a seguinte notação
- $\Box p$: todo mundo que eu vejo marcou p como V
- $\Diamond p$: alguém que eu vejo marcou p como V



Lógicas Modais

- **Linguagem Modal**
- Conjunto de Proposições atômicas

$$\varphi ::= p \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \rightarrow \varphi_2 \mid \neg\varphi \mid \Box\varphi \mid \Diamond\varphi$$

- $\Box p$: todo mundo que eu vejo marcou p como V
- $\Diamond p$: alguém que eu vejo marcou p como V

Semântica Modal

- Mundos Possíveis/Estados
- Fórmulas são avaliadas em grafos $F = (W, R)$
  W é um conjunto não-vazio de *estados* e
  R é uma relação binária em W

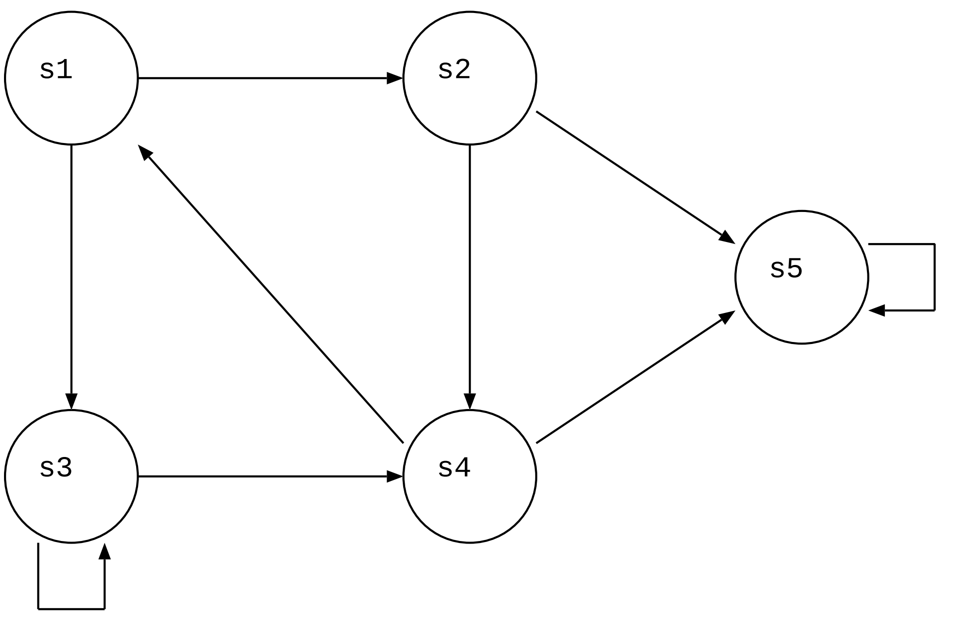
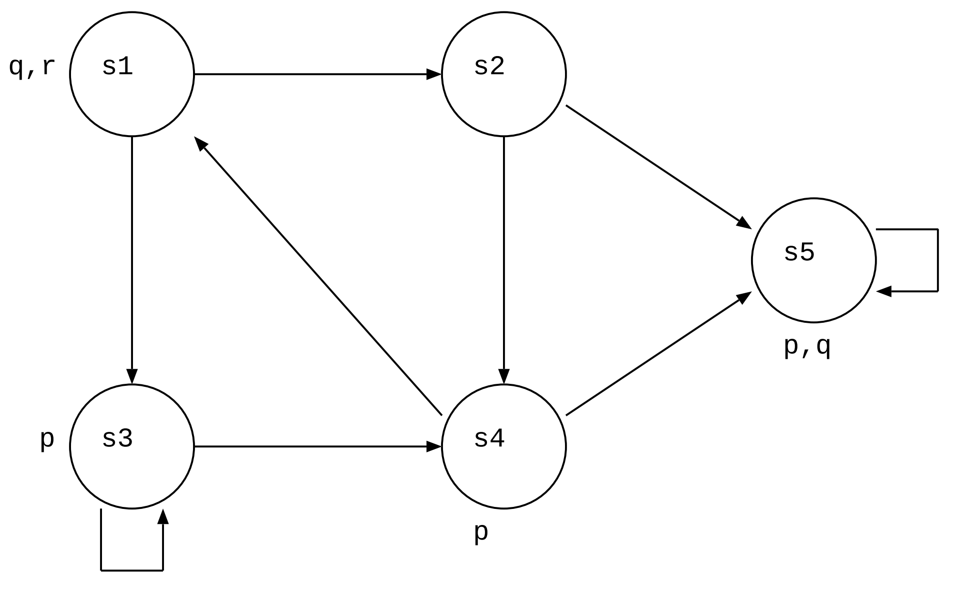


Figura 1: Exemplo de um Frame.

Semântica Modal

- Fórmulas são avaliadas em grafos $F = (W, R)$ rotulados com proposições atômicas
- *modelo* $M = (F, V)$ onde
  $F = (W, R)$ é um *frame* e
  V é uma função associa a cada p o conjunto de estados nos quais p é verdadeiro



Satisfação

- Booleanos: Padrão
- $M, w \Vdash \Box\varphi$ sse para todo $w' \in W$
  se wRw' **implica** $M, w' \Vdash \varphi$

$$\begin{array}{llll} \Box\varphi, & w \xrightarrow{R} w', & \varphi & \\ & \searrow^{R} & & \\ & \quad w', & \varphi & \end{array}$$

- $M, w \Vdash \Diamond\varphi$ sse existe $w' \in W$,
  wRw' **e** $M, w' \Vdash \varphi$

$$\Diamond\varphi, \quad w \xrightarrow{R} w', \quad \varphi$$

Comlexidade e Expressividade

- **Verificação de Modelos** $O(|\varphi| \times (|W| + |R|))$

Comlexidade e Expressividade

- **Verificação de Modelos** $O(|\varphi| \times (|W| + |R|))$
- **Validade**: para **K**, **T** e **S4** é **PSPACE**-Completo.

Comlexidade e Expressividade

- **Verificação de Modelos** $O(|\varphi| \times (|W| + |R|))$
- **Validade**: para **K**, **T** e **S4** é **PSPACE**-Completo.
- **Validade**: para **S5** é **NP**-Completo.

Comlexidade e Expressividade

- **Verificação de Modelos** $O(|\varphi| \times (|W| + |R|))$
- **Validade**: para **K**, **T** e **S4** é **PSPACE**-Completo.
- **Validade**: para **S5** é **NP**-Completo.
- **P** $\subseteq$ **NP** $\subseteq$ **PSPACE** $\subseteq$ **EXPTIME**

Comlexidade e Expressividade

- **Verificação de Modelos** $O(|\varphi| \times (|W| + |R|))$
- **Validade**: para **K**, **T** e **S4** é **PSPACE**-Completo.
- **Validade**: para **S5** é **NP**-Completo.
- **P** $\subseteq$ **NP** $\subseteq$ **PSPACE** $\subseteq$ **EXPTIME**
- **Validade**: **EXPTIME**-Completo,
  - Lógica Dinâmica Proposicional PDL
  - Lógica Epistêmica Multi-agente (c/ Conhecimento Comum)
  - CTL - Computation Tree Logic (Tempotal)
  - μ-Calculus (Menor Ponto Fixo)

Lóg. Epistêmica Multi-agentes

- Conjunto Finito de Agentes $G = a, b, c, ...$
- Duas modalidades para cada agente
- $K_a\varphi$ - $a$na saber φ
- $B_a\varphi$ - $a$na acredita em φ
- $B_a\varphi = \neg K_a \neg\varphi$
- Modalidades de Grupos:
- $E_G\varphi$ - o grupo G sabe φ
- $C_G\varphi$ - φ é de conhecimento comum do grupo G

Modelando Conhecimento

- **Agentes**: **a**na e **b**eto

Modelando Conhecimento

- **Agentes**: **a**na e **b**eto
- Carta contendo a informação:

  $p =$ "ana ganhou R$ 1,00"

  $\neg p =$ "ana **não** ganhou R$ 1,00"

Modelando Conhecimento

- **Agentes**: **a**na e **b**eto
- Carta contendo a informação:

  $p =$ "ana ganhou R$ 1,00"

  $\neg p =$ "ana **não** ganhou R$ 1,00"
- envelope lacrado e sobre a mesa

Modelando Conhecimento

- **Agentes**: **a**na e **b**eto
- Carta contendo a informação:

  $p =$ "ana ganhou R$ 1,00"

  $\neg p =$ "ana **não** ganhou R$ 1,00"
- envelope lacrado e sobre a mesa
- O que ana e beto sabem?

Modelando Conhecimento

- Dois estados possíveis para ana e beto

$$\underline{s_1} - a, b - s_2$$

Modelando Conhecimento

- Dois estados possíveis para ana e beto

- $s_1 =$ "ana ganhou R\$ 1,00"$\Rightarrow\ p$

$\underline{s_1} - a, b - s_2$

Modelando Conhecimento

- Dois estados possíveis para ana e beto

- $s_1 =$ "ana ganhou R$ 1,00"$\Rightarrow\ p$
- $s_2 =$ "ana não ganhou R$ 1,00"$\Rightarrow\ \neg p$

$$\underline{s_1} - a, b - s_2$$

Modelando Conhecimento

- Dois estados possíveis para ana e beto

- $s_1 =$ "ana ganhou R$ 1,00"$\Rightarrow\ p$
- $s_2 =$ "ana não ganhou R$ 1,00"$\Rightarrow\ \neg p$
- ana e beto não sabem se estão em s_1 ou s_2

$$\underline{s_1} - a, b - s_2$$

Modelando Conhecimento

- Dois estados possíveis para ana e beto

- $s_1 =$ "ana ganhou R$ 1,00"$\Rightarrow\ p$
- $s_2 =$ "ana não ganhou R$ 1,00"$\Rightarrow\ \neg p$
- ana e beto não sabem se estão em s_1 ou s_2
- $K_a \neg K_b p$ - ana sabe que beto não sabe p

$$\underline{s_1} - a, b - s_2$$

Modelando Conhecimento

- Dois estados possíveis para ana e beto

- $s_1 =$ "ana ganhou R$ 1,00"$\Rightarrow\ p$
- $s_2 =$ "ana não ganhou R$ 1,00"$\Rightarrow\ \neg p$
- ana e beto não sabem se estão em s_1 ou s_2
- $K_a \neg K_b p$ - ana sabe que beto não sabe p
- $E_G \neg K_b p$ - o grupo sabe que beto não sabe p

$$\underline{s_1} - a, b - s_2$$

Modelando Conhecimento

- Dois estados possíveis para ana e beto

- $s_1 =$ "ana ganhou R$ 1,00"$\Rightarrow\ p$
- $s_2 =$ "ana não ganhou R$ 1,00"$\Rightarrow\ \neg p$
- ana e beto não sabem se estão em s_1 ou s_2
- $K_a \neg K_b p$ - ana sabe que beto não sabe p
- $E_G \neg K_b p$ - o grupo sabe que beto não sabe p
- $C_G \neg K_b p$ - conhecimento comum q beto ñ sabe p

$\underline{s_1} - a, b - s_2$

Paradoxo dos Aniversários

- **A**na e **B**eto querem saber o aniversário de **C**arla

Paradoxo dos Aniversários

- **A**na e **B**eto querem saber o aniversário de **C**arla
- **C**arla dá as seguintes opções de datas:

15 maio	16 maio	19 maio
17 junho	18 junho	
14 julho	16 julho	
14 agosto	15 agosto	17 agosto

Paradoxo dos Aniversários

- **A**na e **B**eto querem saber o aniversário de **C**arla
- **C**arla dá as seguintes opções de datas:

15 maio	16 maio	19 maio
17 junho	18 junho	
14 julho	16 julho	
14 agosto	15 agosto	17 agosto

- **C**arla conta no ouvido de **A**na o mes

Paradoxo dos Aniversários

- **A**na e **B**eto querem saber o aniversário de **C**arla
- **C**arla dá as seguintes opções de datas:

15 maio	16 maio	19 maio
17 junho	18 junho	
14 julho	16 julho	
14 agosto	15 agosto	17 agosto

- **C**arla conta no ouvido de **A**na o mes
- **C**arla conta no ouvido de **B**eto o dia

Paradoxo dos Aniversários

- **A**na e **B**eto querem saber o aniversário de **C**arla
- **C**arla dá as seguintes opções de datas:

15 maio	16 maio	19 maio
17 junho	18 junho	
14 julho	16 julho	
14 agosto	15 agosto	17 agosto

- **C**arla conta no ouvido de **A**na o mes
- **C**arla conta no ouvido de **B**eto o dia
- **A**na: "Eu não sei mas **B**eto também não sabe!"

Paradoxo dos Aniversários

- **A**na e **B**eto querem saber o aniversário de **C**arla
- **C**arla dá as seguintes opções de datas:

15 maio	16 maio	19 maio
17 junho	18 junho	
14 julho	16 julho	
14 agosto	15 agosto	17 agosto

- **C**arla conta no ouvido de **A**na o mes
- **C**arla conta no ouvido de **B**eto o dia
- **A**na: "Eu não sei mas **B**eto também não sabe!"
- **B**eto diz: "Eu já sei data!"

Paradoxo dos Aniversários

- **A**na e **B**eto querem saber o aniversário de **C**arla
- **C**arla dá as seguintes opções de datas:

15 maio	16 maio	19 maio
17 junho	18 junho	
14 julho	16 julho	
14 agosto	15 agosto	17 agosto

- **C**arla conta no ouvido de **A**na o mes
- **C**arla conta no ouvido de **B**eto o dia
- **A**na: "Eu não sei mas **B**eto também não sabe!"
- **B**eto diz: "Eu já sei data!"
- **A**na diz: "Eu já sei data!"

Paradoxo dos Aniversários

- **A**na e **B**eto querem saber o aniversário de **C**arla
- **C**arla dá as seguintes opções de datas:

15 maio	16 maio	19 maio
17 junho	18 junho	
14 julho	16 julho	
14 agosto	15 agosto	17 agosto

- **C**arla conta no ouvido de **A**na o mes
- **C**arla conta no ouvido de **B**eto o dia
- **A**na: "Eu não sei mas **B**eto também não sabe!"
- **B**eto diz: "Eu já sei data!"
- **A**na diz: "Eu já sei data!"

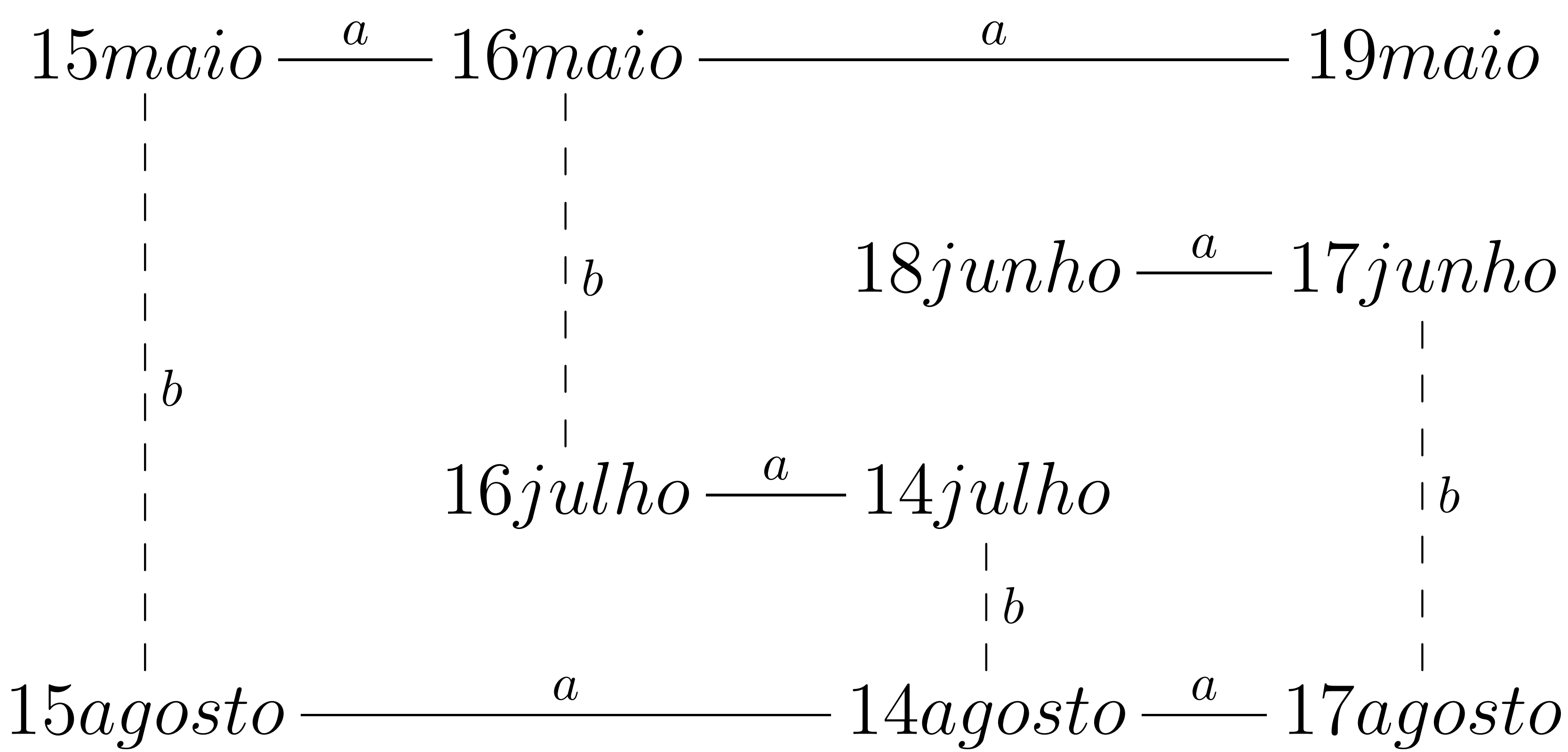
Paradoxo dos Aniversários

- **C**arla dá as seguintes opções de datas:

$15maio$	$16maio$		$19maio$
		$18junho$	$17junho$
	$16julho$	$14julho$	
$15agosto$		$14agosto$	$17agosto$

Paradoxo dos Aniversários

- **C**arla conta no ouvido de **A**na o mes
- **C**arla conta no ouvido de **B**eto o dia



Lógica Epistêmica - Linguagem

- Alfabeto
  - Φ conj. contável de símbolos prop.,
  - $\mathcal{A}$ conjunto finito de agentes,
  - $\neg$ e $\wedge$ conectivos booleanos,
  - K_a uma modalidade para cada agente a,
  - C_G uma modalidade para cada agente a.
- **Linguagem**

$$\varphi ::= p \mid \top \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid K_a\varphi \mid C_G\varphi$$

onde $p \in \Phi$, $a \in \mathcal{A}$.

Semântica

- Um *frame* é um par $F = (W, \sim_a)$ onde
  - W é um conjunto não-vazio de *estados*;
  - $\sim_a$ é uma relação binária para cada agente a
    - Reflexiva
    - Transitiva
    - Simétrica
  - $\sim_G = \bigcup_{a \in G} \sim_a$
  - $\sim_G^\star$ fecho reflexivo transitivo de $\sim_G$
- Um *modelo* é um par $M = (F, V)$ onde
  - $F = (W, R)$ é um *frame* e
  - V é uma função que faz corresponder a todo símb. prop. $p \in \Phi$ o conjunto de estados nos quais p é satisfeito, i.e., $V : \Phi \longmapsto Pow(W)$.

Semântica

- Dada uma estrutura $\mathcal{M} = \langle S, \sim_a, V \rangle$

$$
\begin{array}{lll}
\mathcal{M}, s \models p & iff & s \in V(p) \\
\mathcal{M}, s \models \neg\phi & iff & \mathcal{M}, s \not\models \phi \\
\mathcal{M}, s \models \phi \wedge \psi & iff & \mathcal{M}, s \models \phi \text{ e } \mathcal{M}, s \models \psi \\
\mathcal{M}, s \models K_a\phi & iff & \forall t : s \sim_a t \text{ implica } \mathcal{M}, t \models \phi \\
\mathcal{M}, s \models C_G\phi & iff & \forall t : s \sim_G^{\star} t \text{ implica } \mathcal{M}, t \models \phi
\end{array}
$$

Axiomatização

- **Axiomas**
  1. Tautologias proposicionais,
  2. $K_a(\varphi \to \psi) \to (K_a\varphi \to K_a\psi)$,
  3. $K_a\varphi \to \varphi$,
  4. $K_a\varphi \to K_aK_a\varphi \qquad (+\, introspection)$,
  5. $\neg K_a\varphi \to K_a\neg K_a\varphi \qquad (-\, introspection)$,
  6. $C_G(\varphi \to \psi) \to (C_G\varphi \to C_G\psi)$,
  7. $C_G\varphi \to (\varphi \wedge E_GC_G\varphi)$
  8. $C_G(\varphi \to E_G\varphi) \to (\varphi \to C_G\varphi)$ Indução

- **Regras de inferência**
  M.P. $\varphi, \varphi \to \psi / \psi$ U.G. $\varphi / K_a\varphi$

Complexity

- **Model Checking** $O(|\varphi| \times (|W| + |R|))$
  Polinomial

Complexity

- **Model Checking** $O(|\varphi| \times (|W| + |R|))$
  Polinomial

- **Validity**: **EXPTIME-**Complete

Complexity

- **Model Checking** $O(|\varphi| \times (|W| + |R|))$
  Polinomial
- **Validity**: **EXPTIME**-Complete
- **Finite Model Property**

Modelo de Dolev & Yao

- On the Security of Public Key Protocols

Modelo de Dolev & Yao

- On the Security of Public Key Protocols
- D. Dolev and A. Yao, EEE Transactions on Information Theory, 29(2):198–208, 1983.

Modelo de Dolev & Yao

- On the Security of Public Key Protocols
- D. Dolev and A. Yao, EEE Transactions on Information Theory, 29(2):198–208, 1983.
- **Model: Public Key Protocols**

Modelo de Dolev & Yao

- On the Security of Public Key Protocols
- D. Dolev and A. Yao, EEE Transactions on Information Theory, 29(2):198–208, 1983.

- **Model: Public Key Protocols**

- Perfect Criptography

Modelo de Dolev & Yao

- On the Security of Public Key Protocols
- D. Dolev and A. Yao, EEE Transactions on Information Theory, 29(2):198–208, 1983.

- **Model: Public Key Protocols**

- Perfect Criptography
- Formal Model to Verify Protocols

Modelo de Dolev & Yao

- On the Security of Public Key Protocols
- D. Dolev and A. Yao, EEE Transactions on Information Theory, 29(2):198–208, 1983.

- **Model: Public Key Protocols**

- Perfect Criptography
- Formal Model to Verify Protocols
- Logical Level

Modelo de Dolev & Yao

- On the Security of Public Key Protocols
- D. Dolev and A. Yao, EEE Transactions on Information Theory, 29(2):198–208, 1983.

- **Model: Public Key Protocols**

- Perfect Criptography
- Formal Model to Verify Protocols
- Logical Level
- **NOT** Encription Level.

Modelo de Dolev & Yao

- **Model: Public Key Protocols**

Modelo de Dolev & Yao

- **Model: Public Key Protocols**

- encryption function E_X (public)

Modelo de Dolev & Yao

- **Model: Public Key Protocols**

- encryption function E_X (public)
- decryption function D_X (known only by user X)

Modelo de Dolev & Yao

- **Model: Public Key Protocols**

- encryption function E_X (public)
- decryption function D_X (known only by user X)
- Requirements:

Modelo de Dolev & Yao

- **Model: Public Key Protocols**

- encryption function E_X (public)
- decryption function D_X (known only by user X)
- Requirements:

- $D_X E_X(M) = M$

Modelo de Dolev & Yao

- **Model: Public Key Protocols**

- encryption function E_X (public)
- decryption function D_X (known only by user X)
- Requirements:

- $D_X E_X(M) = M$
- for any user Y knowing $E_X(M)$ does not reveal anything about M

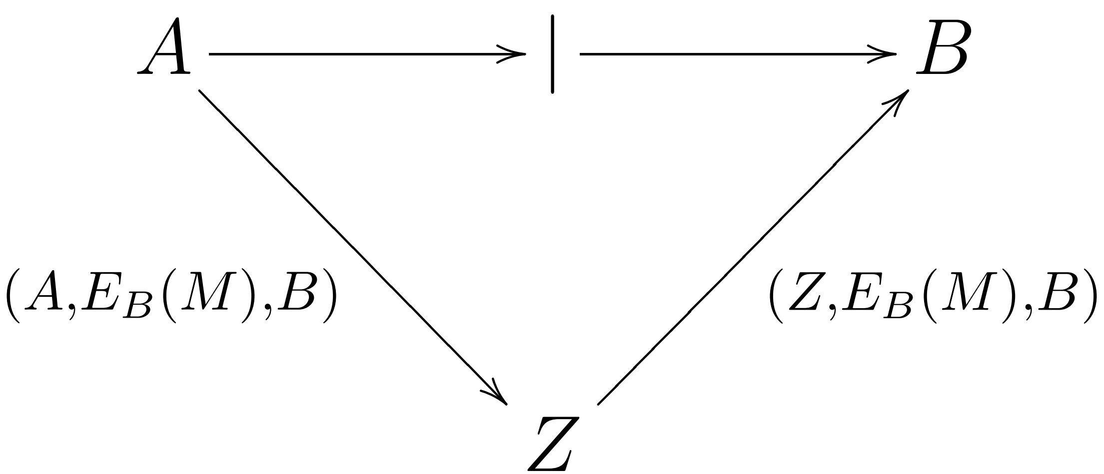
Example 1: Dolev & Yao Model

A sends msg M to B

$$A \longrightarrow (A, E_B(M), B) \longrightarrow B$$

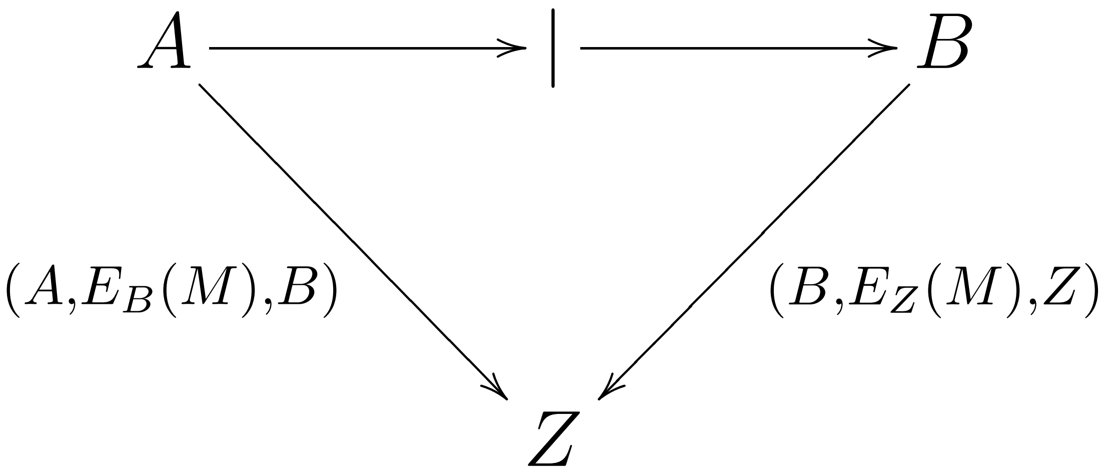
Intruder Z intercepts the message sent from A to B

Intruder Z sends message $(Z, E_B(M), B)$ to B



Example 1: Dolev & Yao Model

B sends message $(B, E_Z(M), Z)$ to Z



Intruder Z decodes $E_Z(M)$ and obtains M

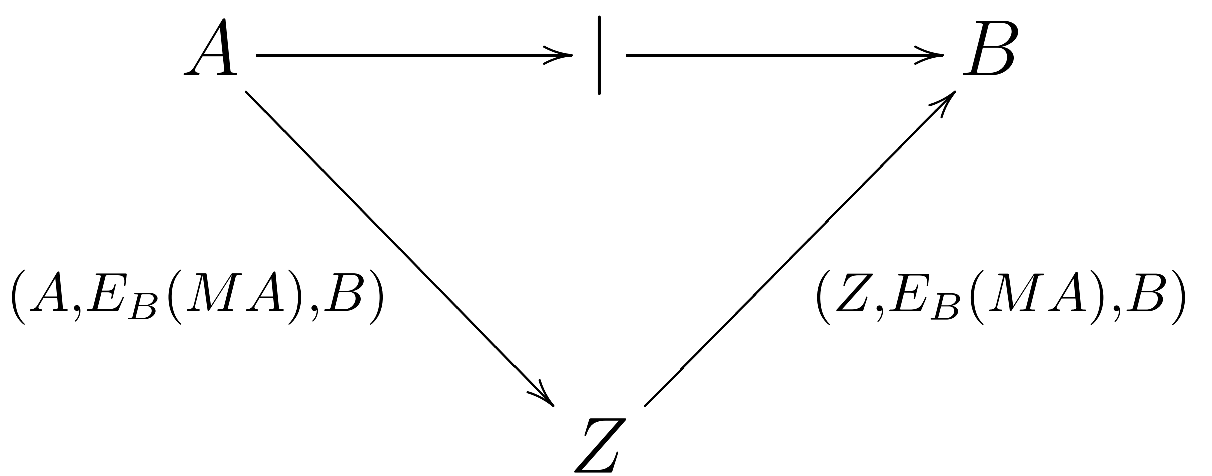
Example 2: Dolev & Yao Model

A sends msg MA to B and B replies to the user that is encrypted with the message M and not to the sender

$$A \longrightarrow (A, E_B(MA), B) \longrightarrow B$$

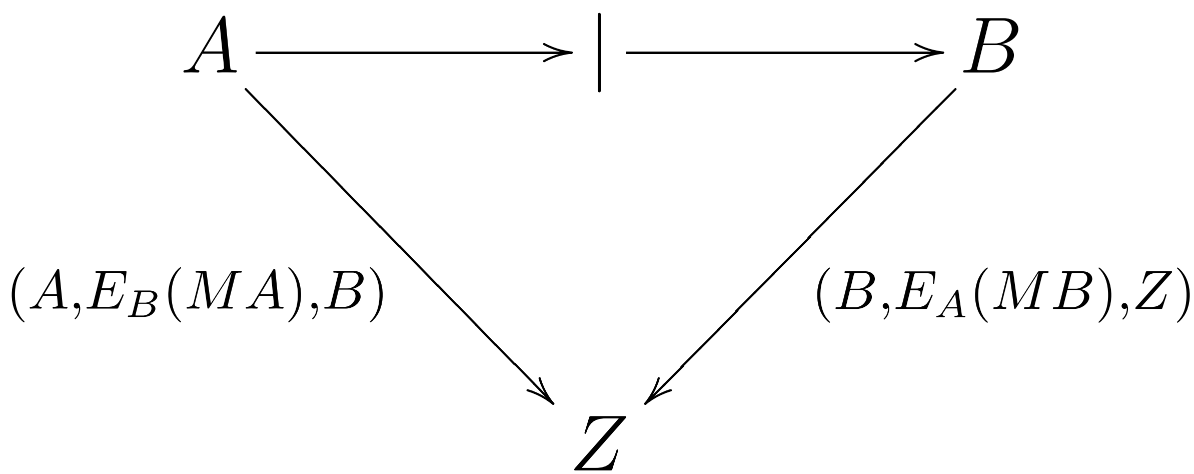
Intruder Z intercepts the message sent from A to B

Intruder Z sends message $(Z, E_B(MA), B)$ to B



Example 2: Dolev & Yao Model

B sends message $(B, E_A(MB), Z)$ to Z



Intruder Z **cannot** decode $E_A(MB)$ to obtain M

It can be proved that this protocol in secure against arbitrary behaviour of the intruder.

Rules : Dolev & Yao Model

- These rules are not presented in the original paper

Rules : Dolev & Yao Model

- These rules are not presented in the original paper
- but they can easily be obtained from the theory presented there.

Rules : Dolev & Yao Model

- These rules are not presented in the original paper
- but they can easily be obtained from the theory presented there.

- We are assuming a set $\mathcal{K} = \{K_1, ...\}$ of keys

Rules : Dolev & Yao Model

- These rules are not presented in the original paper
- but they can easily be obtained from the theory presented there.

- We are assuming a set $\mathcal{K} = \{K_1, ...\}$ of keys

- Encryption function $\{M\}_K$, which encrypt a message M under key K.

Rules : Dolev & Yao Model

- These rules are not presented in the original paper
- but they can easily be obtained from the theory presented there.

- We are assuming a set $\mathcal{K} = \{K_1, ...\}$ of keys

- Encryption function $\{M\}_K$, which encrypt a message M under key K.

- An user can only decrypt a encrypted message $\{M\}_K$ if He knows the key K.

Rules : Dolev & Yao Model

- These rules are not presented in the original paper
- but they can easily be obtained from the theory presented there.

- We are assuming a set $\mathcal{K} = \{K_1, ...\}$ of keys

- Encryption function $\{M\}_K$, which encrypt a message M under key K.

- An user can only decrypt a encrypted message $\{M\}_K$ if He knows the key K.
- Let T be all the information Z has.

Rules : Dolev & Yao Model

$Reflexivity$

$$\frac{M \in T}{T \vdash M}$$

$Encryption$

$$\frac{T \vdash K \quad T \vdash M}{T \vdash \{M\}_K}$$

$Decryption$

$$\frac{T \vdash \{M\}_K \quad T \vdash K}{T \vdash M}$$

$Pair - Composition$

$$\frac{T \vdash M \quad T \vdash N}{T \vdash (M, N)}$$

$Pair - Decomposition$

$$\frac{T \vdash (M, N)}{T \vdash M} \qquad \frac{T \vdash (M, N)}{T \vdash N}$$

Proving Example 1

1. $T = \{Z\}$

Proving Example 1

1. $T = \{Z\}$
2. Intruder Z intercepts the message sent from A to B: $T = \{Z, (A, (E_B(M), B))\}$

Proving Example 1

1. $T = \{Z\}$
2. Intruder Z intercepts the message sent from A to B: $T = \{Z, (A, (E_B(M), B))\}$

2.1 Applying reflexivity to 2.:
$T = \{Z, (A, (E_B(M), B))\} \vdash Z$

Proving Example 1

1. $T = \{Z\}$
2. Intruder Z intercepts the message sent from A to B: $T = \{Z, (A, (E_B(M), B))\}$

2.1 Applying reflexivity to 2.: $T = \{Z, (A, (E_B(M), B))\} \vdash Z$

2.2 Applying reflexivity to 2.: $T = \{Z, (A, (E_B(M), B))\} \vdash (A, (E_B(M), B)$

Proving Example 1

1. $T = \{Z\}$
2. Intruder Z intercepts the message sent from A to B: $T = \{Z, (A, (E_B(M), B))\}$

2.1 Applying reflexivity to 2.: $T = \{Z, (A, (E_B(M), B))\} \vdash Z$

2.2 Applying reflexivity to 2.: $T = \{Z, (A, (E_B(M), B))\} \vdash (A, (E_B(M), B)$

2.3 Applying pair decomposition to 2.2: $T = \{Z, (A, (E_B(M), B))\} \vdash (E_B(M), B)$

Proving Example 1

1. $T = \{Z\}$
2. Intruder Z intercepts the message sent from A to B: $T = \{Z, (A, (E_B(M), B))\}$

2.1 Applying reflexivity to 2.:
$T = \{Z, (A, (E_B(M), B))\} \vdash Z$

2.2 Applying reflexivity to 2.:
$T = \{Z, (A, (E_B(M), B))\} \vdash (A, (E_B(M), B)$

2.3 Applying pair decomposition to 2.2:
$T = \{Z, (A, (E_B(M), B))\} \vdash (E_B(M), B)$

2.4 Applying pair composition to 2.1 and 2.3:
$T = \{Z, (A, (E_B(M), B))\} \vdash (Z, (E_B(M), B))$

Proving Example 1

3. Intruder Z sends message $(Z, E_B(M), B)$ to B:
   $T = \{Z, (A, (E_B(M), B))\}$

Proving Example 1

3. Intruder Z sends message $(Z, E_B(M), B)$ to B:
   $T = \{Z, (A, (E_B(M), B))\}$
4. B sends message $(B, E_Z(M), Z)$ to Z:
   $T = \{Z, (A, (E_B(M), B)), (B, (E_Z(M), Z))\}$

Proving Example 1

3. Intruder Z sends message $(Z, E_B(M), B)$ to B:
   $T = \{Z, (A, (E_B(M), B))\}$
4. B sends message $(B, E_Z(M), Z)$ to Z:
   $T = \{Z, (A, (E_B(M), B)), (B, (E_Z(M), Z))\}$

4.1 Reflexivity to 4.: $T \vdash (B, (E_Z(M), Z))$

Proving Example 1

3. Intruder Z sends message $(Z, E_B(M), B)$ to B:
   $T = \{Z, (A, (E_B(M), B))\}$
4. B sends message $(B, E_Z(M), Z)$ to Z:
   $T = \{Z, (A, (E_B(M), B)), (B, (E_Z(M), Z))\}$

4.1 Reflexivity to 4.: $T \vdash (B, (E_Z(M), Z))$

4.2 Pair decomposition to 4.1: $T \vdash (E_Z(M), Z)$

Proving Example 1

3. Intruder Z sends message $(Z, E_B(M), B)$ to B:
   $T = \{Z, (A, (E_B(M), B))\}$
4. B sends message $(B, E_Z(M), Z)$ to Z:
   $T = \{Z, (A, (E_B(M), B)), (B, (E_Z(M), Z))\}$

4.1 Reflexivity to 4.: $T \vdash (B, (E_Z(M), Z))$

4.2 Pair decomposition to 4.1: $T \vdash (E_Z(M), Z)$

4.3 Pair decomposition to 4.2: $T \vdash E_Z(M)$

Proving Example 1

3. Intruder Z sends message $(Z, E_B(M), B)$ to B:
   $T = \{Z, (A, (E_B(M), B))\}$
4. B sends message $(B, E_Z(M), Z)$ to Z:
   $T = \{Z, (A, (E_B(M), B)), (B, (E_Z(M), Z))\}$

4.1 Reflexivity to 4.: $T \vdash (B, (E_Z(M), Z))$

4.2 Pair decomposition to 4.1: $T \vdash (E_Z(M), Z)$

4.3 Pair decomposition to 4.2: $T \vdash E_Z(M)$

4.4 Pair decomposition to 4.2: $T \vdash Z$

Proving Example 1

3. Intruder Z sends message $(Z, E_B(M), B)$ to B:
   $T = \{Z, (A, (E_B(M), B))\}$
4. B sends message $(B, E_Z(M), Z)$ to Z:
   $T = \{Z, (A, (E_B(M), B)), (B, (E_Z(M), Z))\}$

4.1 Reflexivity to 4.: $T \vdash (B, (E_Z(M), Z))$

4.2 Pair decomposition to 4.1: $T \vdash (E_Z(M), Z)$

4.3 Pair decomposition to 4.2: $T \vdash E_Z(M)$

4.4 Pair decomposition to 4.2: $T \vdash Z$

4.5 Applying Decryption rule to 4.3 and 4.4 we obtain: $T \vdash M$

Spi Calculus

- **A Calculus for Cryptographic Protocols: The Spi Calculus**
  M. Abadi and A. Gordon. Information and Computation , 148(1): 1–70, 1999.

Spi Calculus

- **A Calculus for Cryptographic Protocols: The Spi Calculus**
  M. Abadi and A. Gordon. Information and Computation , 148(1): 1–70, 1999.

- Similar to the π-Calculus

Spi Calculus

- **A Calculus for Cryptographic Protocols: The Spi Calculus**
  M. Abadi and A. Gordon. Information and Computation , 148(1): 1–70, 1999.

- Similar to the π-Calculus
- Process Algebras

Spi Calculus

- **A Calculus for Cryptographic Protocols: The Spi Calculus**
  M. Abadi and A. Gordon. Information and Computation , 148(1): 1–70, 1999.

- Similar to the π-Calculus
- Process Algebras
- Terms are processes

Spi Calculus

- **A Calculus for Cryptographic Protocols: The Spi Calculus**
  M. Abadi and A. Gordon. Information and Computation , 148(1): 1–70, 1999.

- Similar to the π-Calculus
- Process Algebras
- Terms are processes
- Equivalence Relation: Bisimulation

Language - Spi Calculus

- The language of the Spi-Calculus is very similar to the Pi-Caculus.
- In the standard Pi-Calculus terms are only names.

Terms:

$L, M, N ::=$		Terms
	n	name
	(M,N)	pair
	0	zero
	suc(M)	successor
	x	variable
	$\{M\}_N$	shared-key encryption

Processes - Spi Calculus

$P, Q, R ::=$	Processes
$\bar{M}(N).P$	output
$M(x).P$	input
$P \mid Q$	parallel compos
$(\nu)P$	restriction
$!P$	replication
$[M\ is\ N]P$	match
$\mathbf{0}$	nul
$let\ (x, y) = M\ in\ P$	pair splitting
$case\ M\ of\ 0 :\ suc(x) : Q$	integer case
$case\ L\ of\ \{x\}_N\ in\ P$	shared-key decr

Language - Spi Calculus

- Process $[M\ is\ N]P$ behaves as P provided that terms M and N are the same; otherwise it is stuck, that is, it does nothing.

Language - Spi Calculus

- Process $[M\ is\ N]P$ behaves as P provided that terms M and N are the same; otherwise it is stuck, that is, it does nothing.
- Process $let\ (x, y) = M\ in\ P$ behaves as $P[N/x][L/y]$ if term M is the pair (N, L). Otherwise, the process is stuck.

Language - Spi Calculus

- Process $[M \ is \ N]P$ behaves as P provided that terms M and N are the same; otherwise it is stuck, that is, it does nothing.
- Process $let\ (x, y) = M\ in\ P$ behaves as $P[N/x][L/y]$ if term M is the pair (N, L). Otherwise, the process is stuck.
- Process $case\ M\ of\ 0 :\ suc(x) : Q$ behaves as P if term M is 0, as $Q[N/x]$ if M is $suc(N)$. Otherwise, the process is stuck.

Language - Spi Calculus

- Process $[M\ is\ N]P$ behaves as P provided that terms M and N are the same; otherwise it is stuck, that is, it does nothing.
- Process $let\ (x, y) = M\ in\ P$ behaves as $P[N/x][L/y]$ if term M is the pair (N, L). Otherwise, the process is stuck.
- Process $case\ M\ of\ 0 :\ suc(x) : Q$ behaves as P if term M is 0, as $Q[N/x]$ if M is $suc(N)$. Otherwise, the process is stuck.
- Process $case\ L\ of\ \{x\}_N\ in\ P$ attempts to decrypt the term L with the key N. If L is a ciphertext of the form $\{M\}_N$, then the process behaves as $P[M/x]$. Otherwise, the process is stuck.

Example - Spi Calculus

- Example with key establishment

Example - Spi Calculus

- Example with key establishment
- Wide Mouthed Frog

Example - Spi Calculus

- Example with key establishment
- Wide Mouthed Frog
- 3 agents: A, B and S (server)

Example - Spi Calculus

- Example with key establishment
- Wide Mouthed Frog
- 3 agents: A, B and S (server)
- A and B share keys K_{AS} and K_{SB} repectively with server S

Example - Spi Calculus

- Protocol:
- A creates a key K_{AB}
- A send key K_{AB} under encription K_{AS}
- S decript the mesage and send key K_{AB} under encription K_{SB}
- A send mesage M under encription K_{AB}

$$A \xrightarrow{\{K_{AB}\}_{K_{AS}}} S \xrightarrow{\{K_{AB}\}_{K_{SB}}} B$$

Example - Spi Calculus

- **Protocol**

$$
\begin{aligned}
&A(M) = \\
&(\nu K_{AB})(\bar{c}_{AS}\langle\{K_{AB}\}_{K_{AS}}\rangle.(\bar{c}_{AB}\langle\{M\}_{K_{AB}}\rangle) \\
&S = c_{AS}(x).case\ x\ of\ \{y\}_{K_{AS}}\ in\ \bar{c}_{SB}\langle\{y\}_{K_{SB}}\rangle \\
&B = c_{SB}(x).case\ x\ of\ \{y\}_{K_{SB}}\ in \\
&c_{AB}(z).case\ z\ of\ \{w\}_{y}\ in\ F(w) \\
&Inst(M) = (\nu K_{AS})(\nu K_{SB})(A(M) \mid S \mid B)
\end{aligned}
$$

- Since all communication is protected by encryption, communication can take place through public channels: c_{AS}, c_{SB} and c_{AB}

Secrecy - Spi Calculus

- **Secrecy**: $Inst(M) \simeq Inst(M')$, if $F(M) \simeq F(M')$ for all M and M'

Secrecy - Spi Calculus

- **Secrecy**: $Inst(M) \simeq Inst(M')$, if $F(M) \simeq F(M')$ for all M and M'

- **Secrecy**: The message M cannot be read in transit from A to B: if F does not reveal M , then the whole protocol does not reveal M.

Secrecy - Spi Calculus

- **Secrecy**: $Inst(M) \simeq Inst(M')$, if $F(M) \simeq F(M')$ for all M and M'

- **Secrecy**: The message M cannot be read in transit from A to B: if F does not reveal M , then the whole protocol does not reveal M.
- The secrecy property can be stated in terms of equivalences: if $F(M) \simeq F(M'$ for all M and M' , then $Inst(M) \simeq Inst(M')$. This means that if $F(M)$ is indistinguishable from $F(M')$, then the protocol with message M is indistinguishable from the protocol with message M' .

Authenticity - Spi Calculus

- **Authenticity**: $Inst(M) \simeq Inst_{spec}(M)$, for all M

Authenticity - Spi Calculus

- **Authenticity**: $Inst(M) \simeq Inst_{spec}(M)$, for all M

- **Authenticity**:

Authenticity - Spi Calculus

- **Authenticity**: $Inst(M) \simeq Inst_{spec}(M)$, for all M

- **Authenticity**:
- $B = c_{SB}(x).case\ x\ of\ \{y\}_{K_{SB}}\ in\ c_{AB}(z).case\ z\ of\ \{w\}_y\ in\ F(w)$

Authenticity - Spi Calculus

- **Authenticity**: $Inst(M) \simeq Inst_{spec}(M)$, for all M

- **Authenticity**:
- $B = c_{SB}(x).case\ x\ of\ \{y\}_{K_{SB}}\ in\ c_{AB}(z).case\ z\ of\ \{w\}_y\ in\ F(w)$
- $B_{spec} = c_{SB}(x).case\ x\ of\ \{y\}_{K_{SB}}\ in\ c_{AB}(z).case\ z\ of\ \{w\}_y\ in\ F(M)$

Authenticity - Spi Calculus

- **Authenticity**: $Inst(M) \simeq Inst_{spec}(M)$, for all M

- **Authenticity**:
- $B = c_{SB}(x).case\ x\ of\ \{y\}_{K_{SB}}\ in\ c_{AB}(z).case\ z\ of\ \{w\}_y\ in\ F(w)$
- $B_{spec} = c_{SB}(x).case\ x\ of\ \{y\}_{K_{SB}}\ in\ c_{AB}(z).case\ z\ of\ \{w\}_y\ in\ F(M)$
- $Inst_{espc}$ is $Inst$ substituting B by B_{espec}

Authenticity - Spi Calculus

- **Authenticity**: $Inst(M) \simeq Inst_{spec}(M)$, for all M
- **Authenticity**:
- $B = c_{SB}(x).case\ x\ of\ \{y\}_{K_{SB}}\ in$ $c_{AB}(z).case\ z\ of\ \{w\}_y\ in\ F(w)$
- $B_{spec} = c_{SB}(x).case\ x\ of\ \{y\}_{K_{SB}}\ in$ $c_{AB}(z).case\ z\ of\ \{w\}_y\ in\ F(M)$
- $Inst_{espc}$ is $Inst$ substituting B by B_{espec}
- **Authenticity**: The run of the protocol is not affected by any message that an intruder can send to B,

Summary - Spi Calculus

- Equivalence between processes: Bisimulation $\simeq$

Summary - Spi Calculus

- Equivalence between processes: Bisimulation $\simeq$
- **Secrecy**: $Inst(M) \simeq Inst(M')$, if $F(M) \simeq F(M')$ for all M and M'

Summary - Spi Calculus

- Equivalence between processes: Bisimulation $\simeq$
- **Secrecy**: $Inst(M) \simeq Inst(M')$, if $F(M) \simeq F(M')$ for all M and M'
- **Secrecy**: The message M cannot be read in transit from A to B

Summary - Spi Calculus

- Equivalence between processes: Bisimulation $\simeq$
- **Secrecy**: $Inst(M) \simeq Inst(M')$, if $F(M) \simeq F(M')$ for all M and M'
- **Secrecy**: The message M cannot be read in transit from A to B
- **Authenticity**: $Inst(M) \simeq Inst_{spec}(M)$, for all M

Summary - Spi Calculus

- Equivalence between processes: Bisimulation $\simeq$
- **Secrecy**: $Inst(M) \simeq Inst(M')$, if $F(M) \simeq F(M')$ for all M and M'
- **Secrecy**: The message M cannot be read in transit from A to B
- **Authenticity**: $Inst(M) \simeq Inst_{spec}(M)$, for all M
- **Authenticity**: The run of the protocol is not affected by any message that an intruder can send to B

BAN Logic

- **A Logic of Authentication - BAN Logic**
  M. Burrows, M. Abadi, and R. Needham. ACM Transactions on Computer Systems, 8:18–36, 1990.

BAN Logic

- **A Logic of Authentication - BAN Logic**
  M. Burrows, M. Abadi, and R. Needham. ACM Transactions on Computer Systems, 8:18–36, 1990.

- It is hard to call it a **logic**

BAN Logic

- **A Logic of Authentication - BAN Logic**
  M. Burrows, M. Abadi, and R. Needham. ACM Transactions on Computer Systems, 8:18–36, 1990.

- It is hard to call it a **logic**
- It look likes Hoare Logic/Rules

BAN Logic

- **A Logic of Authentication - BAN Logic**
  M. Burrows, M. Abadi, and R. Needham. ACM Transactions on Computer Systems, 8:18–36, 1990.

- It is hard to call it a **logic**
- It look likes Hoare Logic/Rules
- Set of Rules to manipulate assertions

Notation - BAN Logic

- **Notation**

Notation - BAN Logic

- **Notation**
- A, B and S, denote specific principals;

Notation - BAN Logic

- **Notation**
- A, B and S, denote specific principals;
- K_{ab}, K_{as} and K_{bs}, denote specific shared keys;

Notation - BAN Logic

- **Notation**
- A, B and S, denote specific principals;
- K_{ab}, K_{as} and K_{bs}, denote specific shared keys;
- K_a, K_b and K_s, denote specific public keys;

Notation - BAN Logic

- **Notation**
- A, B and S, denote specific principals;
- K_{ab}, K_{as} and K_{bs}, denote specific shared keys;
- K_a, K_b and K_s, denote specific public keys;
- K_a^{-1}, K_b^{-1} and K_s^{-1}, denote the corresponding secret key;

Notation - BAN Logic

- **Notation**
- A, B and S, denote specific principals;
- K_{ab}, K_{as} and K_{bs}, denote specific shared keys;
- K_a, K_b and K_s, denote specific public keys;
- K_a^{-1}, K_b^{-1} and K_s^{-1}, denote the corresponding secret key;
- N_a, N_b and N_s, denote specific statements;

Notation - BAN Logic

- **Notation**
- A, B and S, denote specific principals;
- K_{ab}, K_{as} and K_{bs}, denote specific shared keys;
- K_a, K_b and K_s, denote specific public keys;
- K_a^{-1}, K_b^{-1} and K_s^{-1}, denote the corresponding secret key;
- N_a, N_b and N_s, denote specific statements;
- P, Q and R, range over principals;

Notation - BAN Logic

- **Notation**
- A, B and S, denote specific principals;
- K_{ab}, K_{as} and K_{bs}, denote specific shared keys;
- K_a, K_b and K_s, denote specific public keys;
- K_a^{-1}, K_b^{-1} and K_s^{-1}, denote the corresponding secret key;
- N_a, N_b and N_s, denote specific statements;
- P, Q and R, range over principals;
- X and Y, range over statements;

Notation - BAN Logic

- **Notation**
- A, B and S, denote specific principals;
- K_{ab}, K_{as} and K_{bs}, denote specific shared keys;
- K_a, K_b and K_s, denote specific public keys;
- K_a^{-1}, K_b^{-1} and K_s^{-1}, denote the corresponding secret key;
- N_a, N_b and N_s, denote specific statements;
- P, Q and R, range over principals;
- X and Y, range over statements;
- K, ranges over encryption keys.

Sintaxe- BAN Logic

- The only propositional connective is conjunction, denoted by a comma.

Sintaxe- BAN Logic

- The only propositional connective is conjunction, denoted by a comma.
- Conjunctions as sets: properties such as associativity and commutativity.

Sintaxe- BAN Logic

- The only propositional connective is conjunction, denoted by a comma.
- Conjunctions as sets: properties such as associativity and commutativity.
- P **believes** X;

Sintaxe- BAN Logic

- The only propositional connective is conjunction, denoted by a comma.
- Conjunctions as sets: properties such as associativity and commutativity.
- P **believes** X;
- P **sees** X;

Sintaxe- BAN Logic

- The only propositional connective is conjunction, denoted by a comma.
- Conjunctions as sets: properties such as associativity and commutativity.
- P **believes** X;
- P **sees** X;
- P **said** X;

Sintaxe- BAN Logic

- The only propositional connective is conjunction, denoted by a comma.
- Conjunctions as sets: properties such as associativity and commutativity.
- P **believes** X;
- P **sees** X;
- P **said** X;
- P **controls** X: P has jurisdiction over X;

Sintaxe- BAN Logic

- The only propositional connective is conjunction, denoted by a comma.
- Conjunctions as sets: properties such as associativity and commutativity.
- P **believes** X;
- P **sees** X;
- P **said** X;
- P **controls** X: P has jurisdiction over X;
- **fresh**(X): The formula X is fresh;

Sintaxe - BAN Logic

- $P \overset{K}{\leftrightarrow} Q$: P and Q may use the shared key K to communicate;

Sintaxe - BAN Logic

- $P \overset{K}{\leftrightarrow} Q$: P and Q may use the shared key K to communicate;
- $\overset{K}{\mapsto} P$: P has K as a public key;

Sintaxe - BAN Logic

- $P \overset{K}{\leftrightarrow} Q$: P and Q may use the shared key K to communicate;
- $\overset{K}{\mapsto} P$: P has K as a public key;
- $P \overset{X}{\rightleftharpoons} Q$: The formula X is a secret known only to P and Q;

Sintaxe - BAN Logic

- $P \overset{K}{\leftrightarrow} Q$: P and Q may use the shared key K to communicate;
- $\overset{K}{\mapsto} P$: P has K as a public key;
- $P \overset{X}{\rightleftharpoons} Q$: The formula X is a secret known only to P and Q;
- $\{X\}_K$: This represent the formula X encrypted under the key K.

Sintaxe - BAN Logic

- $P \overset{K}{\leftrightarrow} Q$: P and Q may use the shared key K to communicate;
- $\overset{K}{\mapsto} P$: P has K as a public key;
- $P \overset{X}{\rightleftharpoons} Q$: The formula X is a secret known only to P and Q;
- $\{X\}_K$: This represent the formula X encrypted under the key K.
- $\langle X \rangle_Y$: This represent X combined with the formula Y. In implementations, X is simply concatenated with the password Y.

Logical Postulates - BAN Logic

Message-meaning: rules for interpretation of messages.

$For\, shared\, keys$

$$\frac{P\,\textbf{believes}\,Q\overset{K}{\leftrightarrow}P, \quad P\,\textbf{sees}\,\{X\}_K}{P\,\textbf{believes}\,Q\,\textbf{said}\,X}$$

$For\, public\, keys$

$$\frac{P\,\textbf{believes}\,\overset{K}{\mapsto}Q, \quad P\,\textbf{sees}\,\{X\}_{K^{-1}}}{P\,\textbf{believes}\,Q\,\textbf{said}\,X}$$

Logical Postulates - BAN Logic

Message-meaning: rules for interpretation of messages.

$For\,shared\,secrets$

$$\frac{P\,\mathbf{believes}\,Q \overset{Y}{\rightleftharpoons} P, \quad P\,\mathbf{sees}\,\langle X\rangle_Y}{P\,\mathbf{believes}\,Q\,\mathbf{said}\,X}$$

Logical Postulates - BAN Logic

Jurisdiction:

$$\frac{P \,\mathbf{believes}\, Q \,\mathbf{controls}\, X, \quad P \,\mathbf{believes}\, Q \,\mathbf{believes}\, X}{P \,\mathbf{believes}\, X}$$

Logical Postulates - BAN Logic

Principal sees:

$$\frac{P \,\textbf{sees}\, (X, Y)}{P \,\textbf{sees}\, X} \qquad \frac{P \,\textbf{sees}\, \langle X \rangle_Y}{P \,\textbf{sees}\, X}$$

$$\frac{P \,\textbf{believes}\, Q \overset{K}{\leftrightarrow} P, \quad P \,\textbf{sees}\, \{X\}_K}{P \,\textbf{sees}\, X}$$

Logical Postulates - BAN Logic

Principal sees:

$$\frac{P\,\textbf{believes}\ \stackrel{K}{\mapsto} P, \quad P\,\textbf{sees}\,\{X\}_K}{P\,\textbf{sees}\,X}$$

$$\frac{P\,\textbf{believes}\ \stackrel{K}{\mapsto} Q, \quad P\,\textbf{sees}\,\{X\}_{K^{-1}}}{P\,\textbf{sees}\,X}$$

Logical Postulates - BAN Logic

Fresh:

$$\frac{P\,\textbf{believes}\quad \textbf{fresh}(X)}{P\,\textbf{believes}\quad \textbf{fresh}\,(X, Y)}$$

Quantifiers - BAN Logic

- Quantifiers in Delegations

Quantifiers - BAN Logic

- Quantifiers in Delegations
- A **believes** S **controls** $A \overset{K}{\leftrightarrow} B$

Quantifiers - BAN Logic

- Quantifiers in Delegations
- A **believes** S **controls** $A \stackrel{K}{\leftrightarrow} B$
- $A \,\textbf{believes}\, \forall K.(S \,\textbf{controls}\, A \stackrel{K}{\leftrightarrow} B)$

Quantifiers - BAN Logic

- Quantifiers in Delegations

- A **believes** S **controls** $A \stackrel{K}{\leftrightarrow} B$
- $A\,\mathbf{believes}\,\forall K.(S\,\mathbf{controls}\,A \stackrel{K}{\leftrightarrow} B)$

- $A\,\mathbf{believes}\,\forall K.(S\,\mathbf{controls}\,B\,\mathbf{controls}\,A \stackrel{K}{\leftrightarrow} B)$

Quantifiers - BAN Logic

- Quantifiers in Delegations
- A **believes** S **controls** $A \stackrel{K}{\leftrightarrow} B$
- $A \,\textbf{believes}\, \forall K.(S \,\textbf{controls}\, A \stackrel{K}{\leftrightarrow} B)$
- $A \,\textbf{believes}\, \forall K.(S \,\textbf{controls}\, B \,\textbf{controls}\, A \stackrel{K}{\leftrightarrow} B)$

$$\frac{P \,\textbf{believes}\, \forall V_1 \;\ldots\; V_n.(Q \,\textbf{controls}\, X)}{P \,\textbf{believes}\, Q' \,\textbf{controls}\, X'}$$

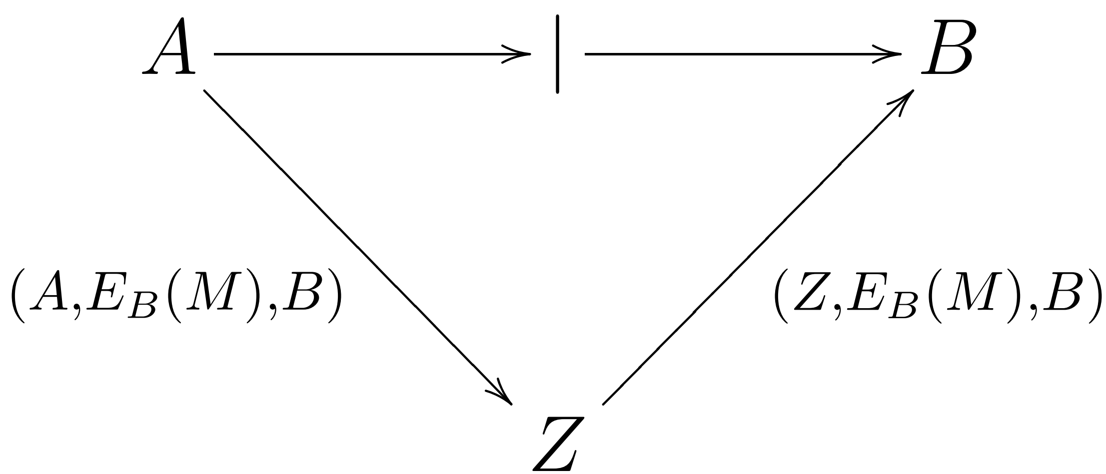
Example 1: Dolev & Yao Model

A sends msg M to B

$$A \longrightarrow (A, E_B(M), B) \longrightarrow B$$

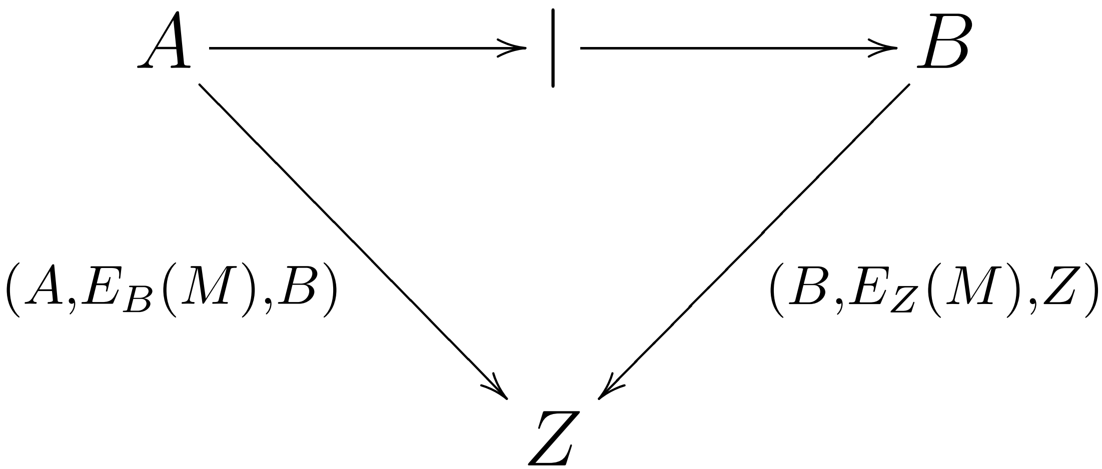
Intruder Z intercepts the message sent from A to B

Intruder Z sends message $(Z, E_B(M), B)$ to B



Example 1: Dolev & Yao Model

B sends message $(B, E_Z(M), Z)$ to Z



Intruder Z decodes $E_Z(M)$ and obtains M

Example - BAN Logic

- $m_1 : A \longrightarrow B : \{m\}_{K_B}$
- $m_2 : Z \longrightarrow B : \{m\}_{K_B}$
- $m_3 : B \longrightarrow Z : \{m\}_{K_Z}$
- B believes $A \overset{K_B}{\leftrightarrow} B$
- Z believes $B \overset{K_Z}{\leftrightarrow} Z$
- $m_1 : Z$ sees $\{m\}_{K_B}$
- $m_2 : B$ sees $\{m\}_{K_B}$
- B sees m $\quad rule\ principal\ sees$
- $m_3 : Z$ sees $\{m\}_{K_Z}$
- Z sees m $\quad rule\ principal\ sees$

Dolev/Yao Epistemic Logic

- Dolev/Yao Multi-Agent Epistemic Logic $\mathbf{S5_{DY}}$

Dolev/Yao Epistemic Logic

- Dolev/Yao Multi-Agent Epistemic Logic $\mathbf{S5_{DY}}$
- Reasoning about Knowledge in Protocols

Dolev/Yao Epistemic Logic

- Dolev/Yao Multi-Agent Epistemic Logic $\mathbf{S5_{DY}}$
- Reasoning about Knowledge in Protocols
- What kind of knowledge?

Dolev/Yao Epistemic Logic

- Dolev/Yao Multi-Agent Epistemic Logic $\mathbf{S5_{DY}}$
- Reasoning about Knowledge in Protocols
- What kind of knowledge?
- Knowledge about

Dolev/Yao Epistemic Logic

- Dolev/Yao Multi-Agent Epistemic Logic $\mathbf{S5_{DY}}$
- Reasoning about Knowledge in Protocols
- What kind of knowledge?
- Knowledge about
- **Keys**,

Dolev/Yao Epistemic Logic

- Dolev/Yao Multi-Agent Epistemic Logic $\mathbf{S5_{DY}}$
- Reasoning about Knowledge in Protocols
- What kind of knowledge?
- Knowledge about
- **Keys**,
- **Messages**,

Dolev/Yao Epistemic Logic

- Dolev/Yao Multi-Agent Epistemic Logic $\mathbf{S5_{DY}}$
- Reasoning about Knowledge in Protocols
- What kind of knowledge?
- Knowledge about
- **Keys**,
- **Messages**,
- **Encription/Decription**,

Dolev/Yao Epistemic Logic

- Dolev/Yao Multi-Agent Epistemic Logic $\mathbf{S5_{DY}}$
- Reasoning about Knowledge in Protocols
- What kind of knowledge?
- Knowledge about
- **Keys**,
- **Messages**,
- **Encription/Decription**,
- **Concatenation**

Dolev/Yao Epistemic Logic

- Dolev/Yao Multi-Agent Epistemic Logic $\mathbf{S5_{DY}}$
- Reasoning about Knowledge in Protocols
- What kind of knowledge?
- Knowledge about
- **Keys**,
- **Messages**,
- **Encription/Decription**,
- **Concatenation**
- **Agents and Groups**, and so on

Language - $S5_{DY}$

- Formulas are built from expressions and not only from proposition symbols.

Language - $S5_{DY}$

- Formulas are built from expressions and not only from proposition symbols.
- An expression is any peace of information that can be encrypted, decrypted or concatenated in order to be communicated.

Language - $\mathbf{S5}_{\mathbf{DY}}$

- Formulas are built from expressions and not only from proposition symbols.
- An expression is any peace of information that can be encrypted, decrypted or concatenated in order to be communicated.
- $\mathbf{S5}_{\mathbf{DY}}$ Alphabet

Language - $\mathbf{S5_{DY}}$

- Formulas are built from expressions and not only from proposition symbols.
- An expression is any peace of information that can be encrypted, decrypted or concatenated in order to be communicated.
- $\mathbf{S5_{DY}}$ Alphabet

- a set Φ of countably many proposition symbols,

Language - $\mathbf{S5}_{\mathrm{DY}}$

- Formulas are built from expressions and not only from proposition symbols.
- An expression is any peace of information that can be encrypted, decrypted or concatenated in order to be communicated.
- $\mathbf{S5}_{\mathbf{DY}}$ Alphabet

- a set Φ of countably many proposition symbols,
- a finite set $\mathcal{A}$ of agents,

Language - $\mathbf{S5_{DY}}$

- Formulas are built from expressions and not only from proposition symbols.
- An expression is any peace of information that can be encrypted, decrypted or concatenated in order to be communicated.
- $\mathbf{S5_{DY}}$ Alphabet
- a set Φ of countably many proposition symbols,
- a finite set $\mathcal{A}$ of agents,
- a set of keys $\mathcal{K} = \{k_1, \cdots\}$,

Language - $\mathbf{S5_{DY}}$

- Formulas are built from expressions and not only from proposition symbols.
- An expression is any peace of information that can be encrypted, decrypted or concatenated in order to be communicated.
- $\mathbf{S5_{DY}}$ Alphabet

- a set Φ of countably many proposition symbols,
- a finite set $\mathcal{A}$ of agents,
- a set of keys $\mathcal{K} = \{k_1, \cdots\}$,
- the boolean connectives $\neg$ and $\wedge$,

Language - $\mathbf{S5_{DY}}$

- Formulas are built from expressions and not only from proposition symbols.
- An expression is any peace of information that can be encrypted, decrypted or concatenated in order to be communicated.
- $\mathbf{S5_{DY}}$ Alphabet

- a set Φ of countably many proposition symbols,
- a finite set $\mathcal{A}$ of agents,
- a set of keys $\mathcal{K} = \{k_1, \cdots\}$,
- the boolean connectives $\neg$ and $\wedge$,
- modalities K_a for each agent a.

Language - $S5_{DY}$

- **Expressions**:

$$E ::= p \mid k \mid (E_1, E_2) \mid \{E\}_k$$

where $k \in \{K_1, \cdots\}$

Language - $S5_{DY}$

- **Expressions**:

$$E ::= p \mid k \mid (E_1, E_2) \mid \{E\}_k$$

where $k \in \{K_1, \cdots\}$

- **Formulas**:

$$\varphi ::= e \mid \top \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid K_a\varphi$$

where $e \in E$, $a \in \mathcal{A}$

Semantics - $\mathrm{S5_{DY}}$

- **Frames**: $\mathbf{S5_{DY}}$ *frame* is a tuple $\mathcal{F} = (W, \sim_a)$

Semantics - $\mathrm{S5_{DY}}$

- **Frames**: $\mathbf{S5_{DY}}$ *frame* is a tuple $\mathcal{F} = (W, \sim_a)$
- W is a non-empty set of states;

Semantics - $\mathrm{S5}_{\mathrm{DY}}$

- **Frames**: $\mathbf{S5}_{\mathbf{DY}}$ *frame* is a tuple $\mathcal{F} = (W, \sim_a)$
- W is a non-empty set of states;
- $\sim_a \subseteq W \times W$ is a reflexive, transitive and symmetric binary relation over W, for each agent $a \in \mathcal{A}$;

Semantics - $\mathrm{S5_{DY}}$

- **Models**: $\mathbf{S5_{DY}}$ *model* is a pair $\mathcal{M} = (\mathcal{F}, \mathbf{V})$, where

Semantics - $\mathrm{S5_{DY}}$

- **Models**: $\mathbf{S5_{DY}}$ *model* is a pair $\mathcal{M} = (\mathcal{F}, \mathbf{V})$, where
- $\mathcal{F}$ is a frame and

Semantics - $\mathbf{S5}_{\mathrm{DY}}$

- **Models**: $\mathbf{S5_{DY}}$ *model* is a pair $\mathcal{M} = (\mathcal{F}, \mathbf{V})$, where
- $\mathcal{F}$ is a frame and
- $\mathbf{V}$ is a valuation function $\mathbf{V} : E \to 2^W$ satisfying the following conditions.

Semantics - $\mathrm{S5_{DY}}$

- **Models**: $\mathbf{S5_{DY}}$ *model* is a pair $\mathcal{M} = (\mathcal{F}, \mathbf{V})$, where
- $\mathcal{F}$ is a frame and
- $\mathbf{V}$ is a valuation function $\mathbf{V} : E \rightarrow 2^W$ satisfying the following conditions.
- $V(m) \cap V(k) \subseteq V(\{m\}_k)$

Semantics - S5$_{\mathrm{DY}}$

- **Models**: $\mathbf{S5_{DY}}$ *model* is a pair $\mathcal{M} = (\mathcal{F}, \mathbf{V})$, where
- $\mathcal{F}$ is a frame and
- $\mathbf{V}$ is a valuation function $\mathbf{V} : E \to 2^W$ satisfying the following conditions.

- $V(m) \cap V(k) \subseteq V(\{m\}_k)$
- $V(\{m\}_k) \cap V(k) \subseteq V(m)$

Semantics - $\mathrm{S5}_{\mathrm{DY}}$

- **Models**: $\mathbf{S5}_{\mathbf{DY}}$ *model* is a pair $\mathcal{M} = (\mathcal{F}, \mathbf{V})$, where
- $\mathcal{F}$ is a frame and
- $\mathbf{V}$ is a valuation function $\mathbf{V} : E \to 2^W$ satisfying the following conditions.

- $V(m) \cap V(k) \subseteq V(\{m\}_k)$
- $V(\{m\}_k) \cap V(k) \subseteq V(m)$
- $V(m) \cap V(n) = V((m, n))$

Semantics - $\mathrm{S5_{DY}}$

- **Models**: $\mathbf{S5_{DY}}$ *model* is a pair $\mathcal{M} = (\mathcal{F}, \mathbf{V})$, where
- $\mathcal{F}$ is a frame and
- $\mathbf{V}$ is a valuation function $\mathbf{V} : E \to 2^W$ satisfying the following conditions.
- $V(m) \cap V(k) \subseteq V(\{m\}_k)$
- $V(\{m\}_k) \cap V(k) \subseteq V(m)$
- $V(m) \cap V(n) = V((m, n))$
- We call a rooted multi-agent epistemic model $(\mathcal{M}, s)$ an epistemic state.

Semantics - $S5_{DY}$

Satisfaction: $\mathcal{M}, s \models \varphi$

- $\mathcal{M}, s \models e\ iff\ s \in V(e)$
- $\mathcal{M}, s \models \neg\phi\ iff\ \mathcal{M}, s \not\models \phi$
- $\mathcal{M}, s \models \phi \wedge \psi\ iff\ \mathcal{M}, s \models \phi$ and $\mathcal{M}, s \models \psi$
- $\mathcal{M}, s \models K_a\phi\ iff$ for all $s' \in S : s \sim_a s' \Rightarrow \mathcal{M}, s' \models \phi$

Axiomatization - $S5_{DY}$

- *All instantiations of propositional tautologies,*

Axiomatization - $\mathrm{S5}_{\mathrm{DY}}$

- *All instantiations of propositional tautologies,*
- $K_a(\varphi \to \psi) \to (K_a\varphi \to K_a\psi)$,

Axiomatization - $\mathrm{S5}_{\mathrm{DY}}$

- *All instantiations of propositional tautologies,*
- $K_a(\varphi \to \psi) \to (K_a\varphi \to K_a\psi)$,
- $K_a\varphi \to \varphi$,

Axiomatization - $S5_{DY}$

- *All instantiations of propositional tautologies,*
- $K_a(\varphi \rightarrow \psi) \rightarrow (K_a\varphi \rightarrow K_a\psi)$,
- $K_a\varphi \rightarrow \varphi$,
- $K_a\varphi \rightarrow K_aK_a\varphi \qquad (+\ introspection)$,

Axiomatization - $S5_{DY}$

- *All instantiations of propositional tautologies,*
- $K_a(\varphi \rightarrow \psi) \rightarrow (K_a\varphi \rightarrow K_a\psi)$,
- $K_a\varphi \rightarrow \varphi$,
- $K_a\varphi \rightarrow K_aK_a\varphi \qquad (+\ introspection)$,
- $\neg K_a\varphi \rightarrow K_a\neg K_a\varphi \qquad (-\ introspection)$,

Axiomatization - $\mathrm{S5}_{\mathrm{DY}}$

- *All instantiations of propositional tautologies,*
- $K_a(\varphi \to \psi) \to (K_a\varphi \to K_a\psi)$,
- $K_a\varphi \to \varphi$,
- $K_a\varphi \to K_aK_a\varphi \qquad (+\ introspection)$,
- $\neg K_a\varphi \to K_a\neg K_a\varphi \qquad (-\ introspection)$,

- $K_am \wedge K_ak \to K_a\{m\}_k \quad (encryption)$

Axiomatization - $S5_{DY}$

- *All instantiations of propositional tautologies,*
- $K_a(\varphi \to \psi) \to (K_a\varphi \to K_a\psi)$,
- $K_a\varphi \to \varphi$,
- $K_a\varphi \to K_aK_a\varphi \qquad (+\ introspection)$,
- $\neg K_a\varphi \to K_a\neg K_a\varphi \qquad (-\ introspection)$,

- $K_am \wedge K_ak \to K_a\{m\}_k \quad (encryption)$
- $K_a\{m\}_k \wedge K_ak \to K_am \quad (decryption)$

Axiomatization - $S5_{DY}$

- *All instantiations of propositional tautologies,*
- $K_a(\varphi \to \psi) \to (K_a\varphi \to K_a\psi)$,
- $K_a\varphi \to \varphi$,
- $K_a\varphi \to K_aK_a\varphi \qquad (+\ introspection)$,
- $\neg K_a\varphi \to K_a\neg K_a\varphi \qquad (-\ introspection)$,

- $K_am \wedge K_ak \to K_a\{m\}_k \quad (encryption)$
- $K_a\{m\}_k \wedge K_ak \to K_am \quad (decryption)$
- $K_am \wedge K_an \leftrightarrow K_a(m, n)\ (pair\ comp./decomp.)$

Axiomatization - $\mathrm{S5_{DY}}$

- *All instantiations of propositional tautologies,*
- $K_a(\varphi \to \psi) \to (K_a\varphi \to K_a\psi)$,
- $K_a\varphi \to \varphi$,
- $K_a\varphi \to K_aK_a\varphi \qquad (+\ introspection)$,
- $\neg K_a\varphi \to K_a\neg K_a\varphi \qquad (-\ introspection)$,
- $K_am \wedge K_ak \to K_a\{m\}_k \quad (encryption)$
- $K_a\{m\}_k \wedge K_ak \to K_am \quad (decryption)$
- $K_am \wedge K_an \leftrightarrow K_a(m,n)\ (pair\ comp./decomp.)$

Inference Rules

M.P. $\varphi, \varphi \to \psi / \psi$ U.G. $\varphi / K_a\varphi$

Soundness - $\mathrm{S5}_{\mathrm{DY}}$

Theorem **Soundness**: The following axioms are sound.

- $K_a m \wedge K_a k \rightarrow K_a\{m\}_k \quad (encryption)$
- $K_a\{m\}_k \wedge K_a k \rightarrow K_a m \quad (decryption)$
- $K_a m \wedge K_a n \leftrightarrow K_a(m, n)\ (pair\ comp./decomp.)$

Completeness - $\mathbf{S5_{DY}}$

- Theorem **Completeness**: $\mathbf{S5_{DY}}$ is complete w.r.t. the class of $\mathbf{S5_{DY}}$ models.

Completeness - $\mathbf{S5_{DY}}$

- Theorem **Completeness**: $\mathbf{S5_{DY}}$ is complete w.r.t. the class of $\mathbf{S5_{DY}}$ models.

- Prove it by canonical models

Completeness - $\mathbf{S5_{DY}}$

- Theorem **Completeness**: $\mathbf{S5_{DY}}$ is complete w.r.t. the class of $\mathbf{S5_{DY}}$ models.

- Prove it by canonical models

- Fisher/Ladner construction $\Rightarrow$ Finite model

Completeness - $\mathbf{S5_{DY}}$

- Theorem **Completeness**: $\mathbf{S5_{DY}}$ is complete w.r.t. the class of $\mathbf{S5_{DY}}$ models.

- Prove it by canonical models

- Fisher/Ladner construction $\Rightarrow$ Finite model

- Finite Model Property

Common Knowledge - $\mathrm{S5_{DY}}$

- $\mathbf{S5^{CK}_{DY}} = \mathbf{S5_{DY}} +$ **Common Knowledge**

Common Knowledge - $\mathrm{S5_{DY}}$

- $\mathbf{S5^{CK}_{DY}} = \mathbf{S5_{DY}} +$ **Common Knowledge**
- **Modal Operator**: $C_G\varphi$ - φ is common knowledge for agents in group G

Common Knowledge - $\mathrm{S5_{DY}}$

- $\mathbf{S5^{CK}_{DY}} = \mathbf{S5_{DY}} +$ **Common Knowledge**
- **Modal Operator**: $C_G\varphi$ - φ is common knowledge for agents in group G
- Axioms and rules of $\mathbf{S5_{DY}}$

Common Knowledge - $\mathrm{S5_{DY}}$

- $\mathbf{S5_{DY}^{CK}} = \mathbf{S5_{DY}} +$ **Common Knowledge**
- **Modal Operator**: $C_G\varphi$ - φ is common knowledge for agents in group G
- Axioms and rules of $\mathbf{S5_{DY}}$
- $E_G\varphi \leftrightarrow \bigwedge_{a\in G} K_a\varphi$

Common Knowledge - $\mathrm{S5_{DY}}$

- $\mathbf{S5^{CK}_{DY}} = \mathbf{S5_{DY}} +$ **Common Knowledge**

- **Modal Operator**: $C_G\varphi$ - φ is common knowledge for agents in group G

- Axioms and rules of $\mathbf{S5_{DY}}$
- $E_G\varphi \leftrightarrow \bigwedge_{a\in G} K_a\varphi$
- $C_G(\varphi \to \psi) \to (C_G\varphi \to C_G\psi)$,

Common Knowledge - $\mathrm{S5_{DY}}$

- $\mathbf{S5_{DY}^{CK}} = \mathbf{S5_{DY}} +$ **Common Knowledge**
- **Modal Operator**: $C_G\varphi$ - φ is common knowledge for agents in group G
- Axioms and rules of $\mathbf{S5_{DY}}$
- $E_G\varphi \leftrightarrow \bigwedge_{a\in G} K_a\varphi$
- $C_G(\varphi \to \psi) \to (C_G\varphi \to C_G\psi)$,
- $C_G\varphi \to (\varphi \wedge E_G C_G\varphi)$,

Common Knowledge - $\mathrm{S5_{DY}}$

- $\mathbf{S5^{CK}_{DY}} = \mathbf{S5_{DY}} +$ **Common Knowledge**
- **Modal Operator**: $C_G\varphi$ - φ is common knowledge for agents in group G
- Axioms and rules of $\mathbf{S5_{DY}}$
- $E_G\varphi \leftrightarrow \bigwedge_{a\in G} K_a\varphi$
- $C_G(\varphi \to \psi) \to (C_G\varphi \to C_G\psi)$,
- $C_G\varphi \to (\varphi \wedge E_G C_G\varphi)$,
- $C_G(\varphi \to E_G\varphi) \to (\varphi \to C_G\varphi)\ \ (+\ induction)$,

Common Knowledge - $\mathbf{S5_{DY}}$

- $\mathbf{S5_{DY}^{CK}} = \mathbf{S5_{DY}} + $ **Common Knowledge**
- **Modal Operator**: $C_G\varphi$ - φ is common knowledge for agents in group G
- Axioms and rules of $\mathbf{S5_{DY}}$
- $E_G\varphi \leftrightarrow \bigwedge_{a\in G} K_a\varphi$
- $C_G(\varphi \to \psi) \to (C_G\varphi \to C_G\psi)$,
- $C_G\varphi \to (\varphi \wedge E_G C_G\varphi)$,
- $C_G(\varphi \to E_G\varphi) \to (\varphi \to C_G\varphi)\ \ (+\ induction)$,
- Inference Rules: U.G. $\varphi / C_G\varphi$

Complexity

- **To be donne!**

Complexity

- **To be donne!**
- Hints for $\mathbf{S5}_{\mathrm{DY}}$

Complexity

- **To be donne!**
- Hints for $\mathbf{S5_{DY}}$
- $\mathbf{S5}$
  - **Model Checking**: Polynomial
  - **Validity**: NP-Complete

Complexity

- **To be donne!**

- Hints for $\mathbf{S5_{DY}}$
- $\mathbf{S5}$
  - **Model Checking**: Polynomial
  - **Validity**: NP-Complete

- Hints for $\mathbf{S5_{DY}^{CK}}$

Complexity

- **To be donne!**

- Hints for $\mathbf{S5_{DY}}$
- $\mathbf{S5}$
  - **Model Checking**: Polynomial
  - **Validity**: NP-Complete

- Hints for $\mathbf{S5^{CK}_{DY}}$
- $\mathbf{S5^{CK}}$
  - **Model Checking**: Polynomial
  - **Validity**: EXPTIME-Complete

Complexity

- **To be donne!**

- Hints for $\mathbf{S5_{DY}}$
- $\mathbf{S5}$
  - **Model Checking**: Polynomial
  - **Validity**: NP-Complete

- Hints for $\mathbf{S5^{CK}_{DY}}$
- $\mathbf{S5^{CK}}$
  - **Model Checking**: Polynomial
  - **Validity**: EXPTIME-Complete

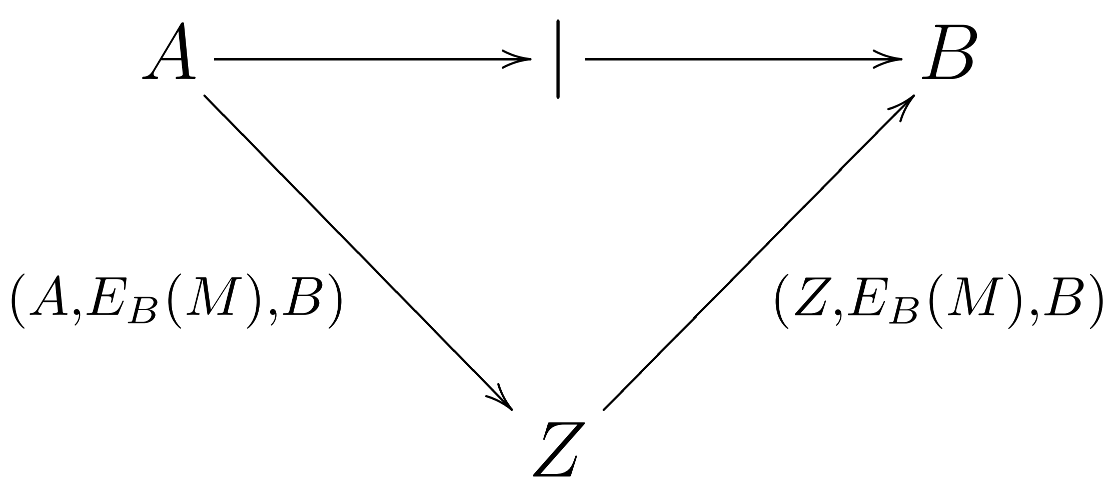
Example 1: Dolev & Yao - Cont.

A sends msg M to B

$$A \longrightarrow (A, E_B(M), B) \longrightarrow B$$

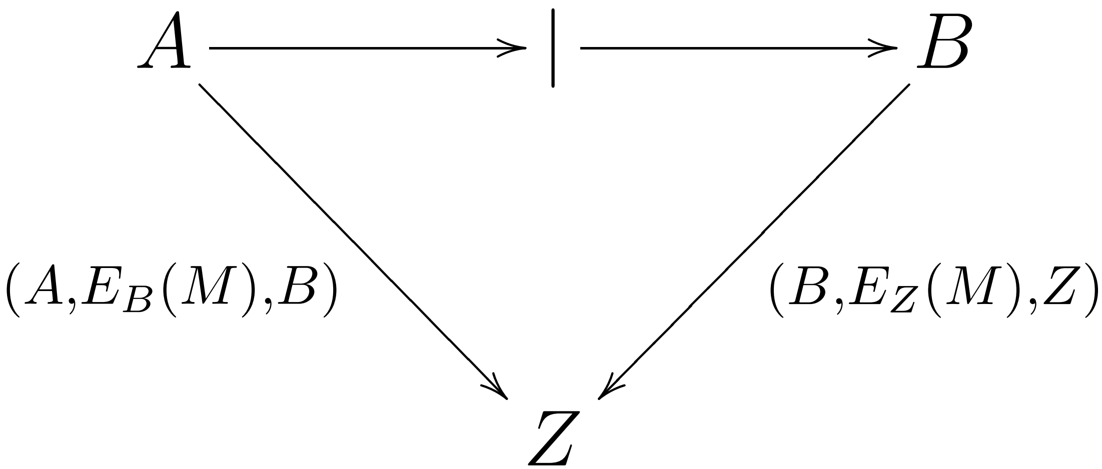
Intruder Z intercepts the message sent from A to B

Intruder Z sends message $(Z, E_B(M), B)$ to B



Example 1: Dolev & Yao - Cont.

B sends message $(B, E_Z(M), Z)$ to Z



Intruder Z decodes $E_Z(M)$ and obtains M

Proving Example 1

- Proving example 1 Dolev & Yao in $\mathbf{S5_{DY}}$

Proving Example 1

- Proving example 1 Dolev & Yao in $\mathbf{S5_{DY}}$
- Three agents A, B and Z.

Proving Example 1

- Proving example 1 Dolev & Yao in $\mathbf{S5_{DY}}$

- Three agents A, B and Z.
- $K_{XY} = K_{YX}$ for every agent X and Y.

Proving Example 1

- Proving example 1 Dolev & Yao in $\mathbf{S5_{DY}}$

- Three agents A, B and Z.
- $K_{XY} = K_{YX}$ for every agent X and Y.

- **Initial Knowledge**:

$$KB_0 = \{K_A k_{AB},\ K_B k_{AB},\ K_B k_{BZ},\ K_Z k_{BZ},\ K_A m\}$$

Proving Example 1

$$KB_0 = \{K_A k_{AB},\ K_B k_{AB},\ K_B k_{BZ},\ K_Z k_{BZ},\ K_A m\}$$

$$\Big\downarrow send_{AB}(\{m\}_{k_{AB}})$$

$$- - -$$

$$\Big\downarrow Z\ intecepts$$

$$KB_1 := KB_0 \cup K_Z\{m\}_{k_{AB}}$$

$$\Big\downarrow send_{ZB}(\{m\}_{k_{AB}})$$

$$KB_2 := KB_1 \cup K_B\{m\}_{k_{AB}}$$

$$K_B m \quad ax.\,7.$$

$$K_B\{m\}_{k_{ZB}} \quad ax.\,6.$$

Proving Example 1

$$KB_2 := KB_1 \cup K_B\{m\}_{k_{AB}}$$

$$K_B m \quad ax.\, 7.$$

$$K_B\{m\}_{k_{ZB}} \quad ax.\, 6.$$

$$\Big\downarrow send_{BZ}(\{m\}_{k_{BZ}})$$

$$KB_3 := KB_2 \cup K_Z\{m\}_{k_{BZ}}$$

$$K_Z m \quad ax. 7$$

Intruder Z knows M

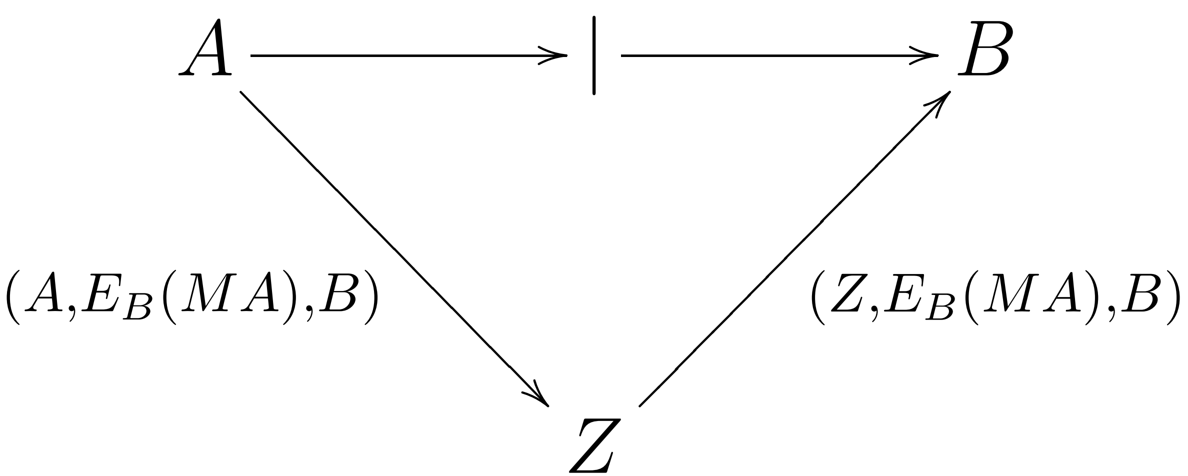
Example 2: Dolev & Yao Model

A sends msg MA to B and B replies to the user that is encrypted with the message M and not to the sender

$$A \longrightarrow (A, E_B(MA), B) \longrightarrow B$$

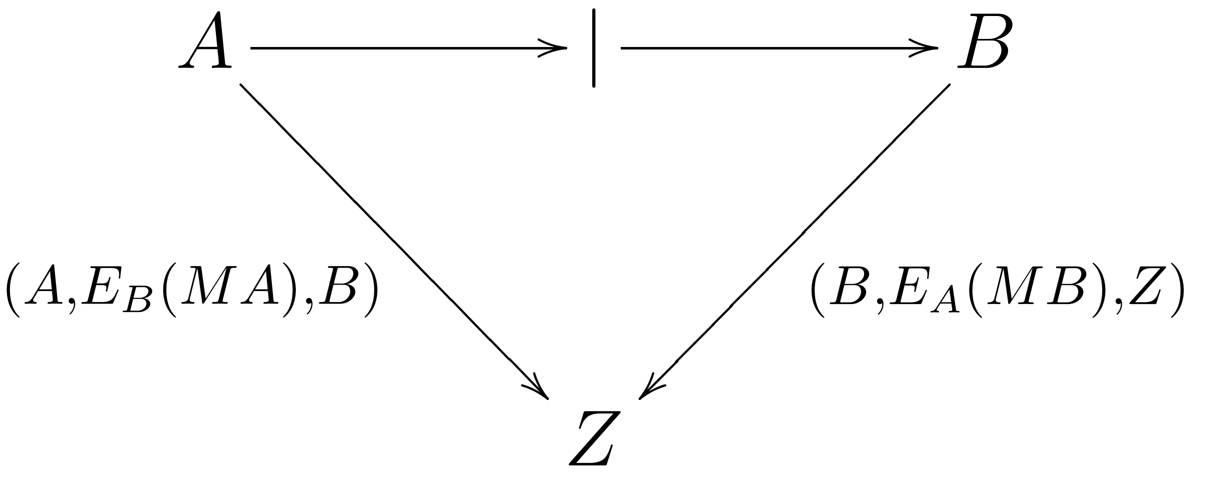
Intruder Z intercepts the message sent from A to B

Intruder Z sends message $(Z, E_B(MA), B)$ to B



Example 2: Dolev & Yao Model

B sends message $(B, E_A(MB), Z)$ to Z



Intruder Z **cannot** decode $E_A(MB)$ to obtain M

It can be proved that this protocol in secure against arbitrary behaviour of the intruder.

Proving Example 2

- Proving example 2 Dolev & Yao in $\mathbf{S5_{DY}}$

Proving Example 2

- Proving example 2 Dolev & Yao in $\mathbf{S5_{DY}}$
- Three agents A, B and Z.

Proving Example 2

- Proving example 2 Dolev & Yao in $\mathbf{S5_{DY}}$

- Three agents A, B and Z.
- $K_{XY} = K_{YX}$ for every agent X and Y.

Proving Example 2

- Proving example 2 Dolev & Yao in $\mathbf{S5_{DY}}$

- Three agents A, B and Z.
- $K_{XY} = K_{YX}$ for every agent X and Y.

- **Initial Knowledge**:

$$KB_0 = \{K_A k_{AB},\ K_B k_{AB},\ K_B k_{BZ},\ K_Z k_{BZ},\ K_A m\}$$

Proving Example 2

$$KB_0 = \{K_A k_{AB},\ K_B k_{AB},\ K_B k_{BZ},\ K_Z k_{BZ},\ K_A m\}$$

$$KB_0 \vdash K_A(k_{AB}, m)$$

$$KB_0 \vdash K_A\{(k_{AB}, m)\}_{k_{AB}} \qquad ax.\ 6$$

$$\Big\downarrow send_{AB}(\{(k_{AB},m)\}_{k_{AB}})$$

$$- - -$$

$$\Big\downarrow Z\ intecepts$$

$$KB_1 := KB_0 \cup K_Z\{(k_{AB}, m)\}_{k_{AB}}$$

$$\Big\downarrow send_{ZB}(\{(k_{AB},m)\}_{k_{AB}})$$

$$KB_2 := KB_1 \cup K_B\{(k_{AB}, m)\}_{k_{AB}}$$

Proving Example 2

$$KB_2 := KB_1 \cup K_B\{(k_{AB}, m)\}_{k_{AB}}$$

$$K_B(k_{AB}, m) \quad ax.\,7.$$

$$K_B m \quad ax.\,8.$$

$$K_B\{(k_{AB}, m)\}_{k_{AB}} \quad ax.\,6.$$

$$\Big\downarrow {\scriptstyle send_{BZ}(\{(k_{AB},m)\}_{k_{AB}})}$$

$$KB_3 := KB_2 \cup K_Z\{(k_{AB}, m)\}_{k_{AB}}$$

$$KB_3 \not\vdash K_Z m$$

More Examples

- Third example of the original article of Dolev & Yao

More Examples

- Third example of the original article of Dolev & Yao
- Kerberos Protocol

More Examples

- Third example of the original article of Dolev & Yao
- Kerberos Protocol
- Andrew Secure RPC Handshake Protocol

Adding Actions

- Adding Actions to $\mathbf{S5_{DY}}$

Adding Actions

- Adding Actions to $\mathbf{S5_{DY}}$
- In all protocols - Actions are executed in the Meta-level

Adding Actions

- Adding Actions to $\mathbf{S5_{DY}}$
- In all protocols - Actions are executed in the Meta-level
- Internalizing Actions to $\mathbf{S5_{DY}}$

Adding Actions

- Adding Actions to $\mathbf{S5_{DY}}$
- In all protocols - Actions are executed in the Meta-level
- Internalizing Actions to $\mathbf{S5_{DY}}$
- Action Dolev/Yao Multi-Agent Epistemic Logic $\mathbf{S5_{DY}^{A}}$

axiom: $K_A m \rightarrow [send_{AB}(M)]K_B m$???????

Future Works

- Adding Common Knowledge to $\mathbf{S5_{DY}}$

Future Works

- Adding Common Knowledge to $\mathbf{S5_{DY}}$
- Adding Actions to $\mathbf{S5_{DY}}$

Future Works

- Adding Common Knowledge to $\mathbf{S5_{DY}}$
- Adding Actions to $\mathbf{S5_{DY}}$
- Computational Complexity

Future Works

- Adding Common Knowledge to $\mathbf{S5_{DY}}$
- Adding Actions to $\mathbf{S5_{DY}}$
- Computational Complexity
- Model Checking Algorithms