Epistemic Logics for Authentication and Secrecy in Protocols

V Workshop Científico LogIA - PenCogLin

UFC

Mario Benevides

Federal University of Rio de Janeiro - Brazil

November-2015

Join work with Anna Oliveira, Luiz Fernandez and Ivan Varzinczak

Overview

- Motivation

Overview

- Motivation
- Algebraic and Logic Approaches for Autheticity and Secrecy

Overview

- Motivation
- Algebraic and Logic Approaches for Autheticity and Secrecy
- Epistemic Logic for Autheticity and Secrecy

Overview

- Motivation
- Algebraic and Logic Approaches for Autheticity and Secrecy
- Epistemic Logic for Autheticity and Secrecy
- Logics for Authorization

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

Properties

- Autheticity:

  The content of a message encrypted for agent A can only be read by A

Properties

- Autheticity:

  The content of a message encrypted for agent A can only be read by A

- Secrecy:

  No agents can read the content of any message which was not encrypted for him

Por que Lógicas Modais?

- Expressivas e Boa Complexidade

Por que Lógicas Modais?

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

Por que Lógicas Modais?

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

Por que Lógicas Modais?

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

Por que Lógicas Modais?

- Expressivas e Boa Complexidade
- **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**

Por que Lógicas Modais?

- Expressivas e Boa Complexidade
- **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)

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 mês

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 mês
- **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 mês
- **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 mês
- **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 mês
- **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 mês
- **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!"

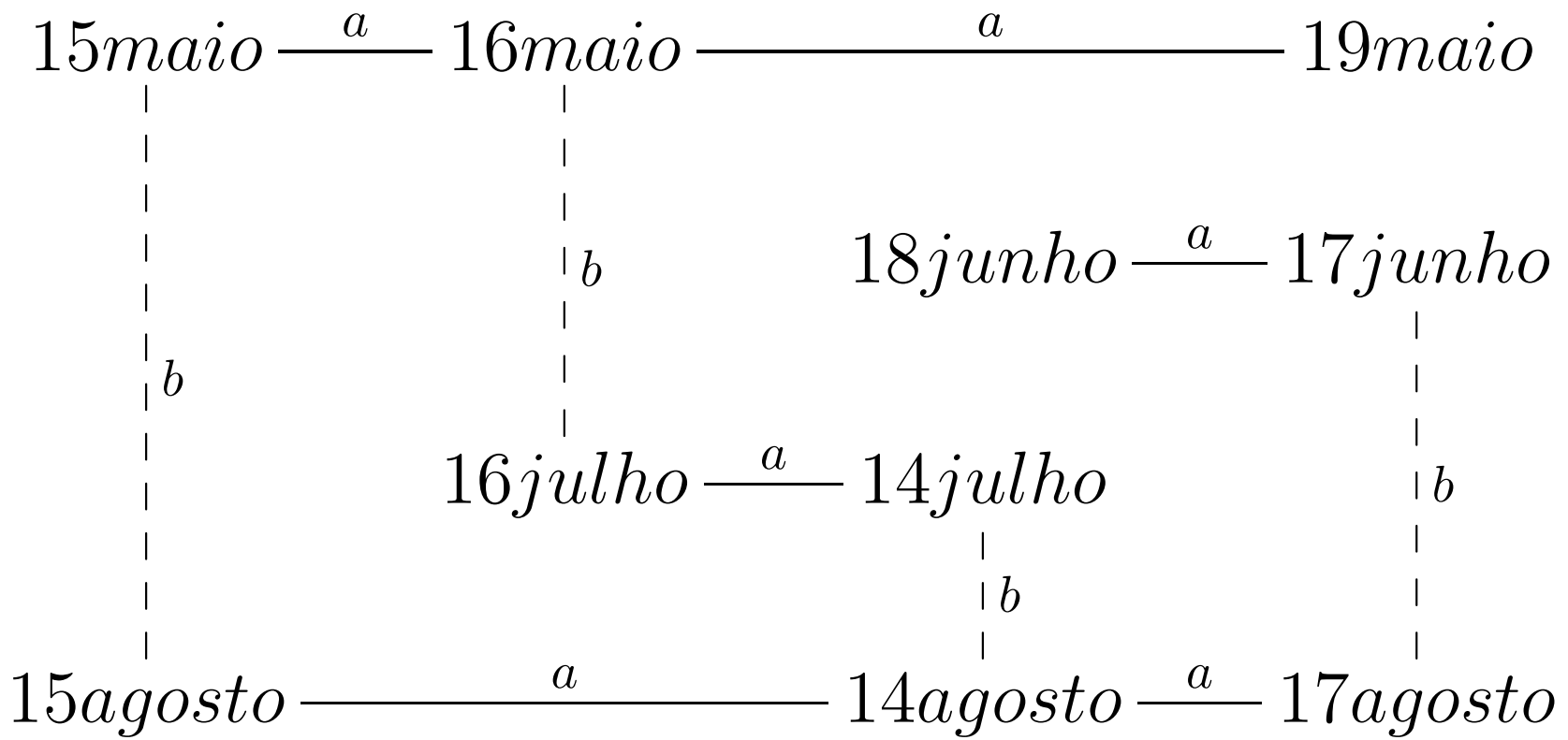
Paradoxo dos Aniversários

- **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



Quebra Cab. Lógicos Beberrões

- **Agentes**: **A**na, **B**eto e **C**arla

Quebra Cab. Lógicos Beberrões

- **Agentes**: **A**na, **B**eto e **C**arla
- Entram num bar.

Quebra Cab. Lógicos Beberrões

- **Agentes**: **A**na, **B**eto e **C**arla
- Entram num bar.
- **Garçon**: Todos vão querer beber cerveja?

Quebra Cab. Lógicos Beberrões

- **Agentes**: **A**na, **B**eto e **C**arla
- Entram num bar.
- **Garçon**: Todos vão querer beber cerveja?
- **A**na diz: Eu não sei!

Quebra Cab. Lógicos Beberrões

- **Agentes**: **A**na, **B**eto e **C**arla
- Entram num bar.
- **Garçon**: Todos vão querer beber cerveja?
- **A**na diz: Eu não sei!
- **B**eto diz: Eu não sei!

Quebra Cab. Lógicos Beberrões

- **Agentes**: **A**na, **B**eto e **C**arla
- Entram num bar.
- **Garçon**: Todos vão querer beber cerveja?
- **A**na diz: Eu não sei!
- **B**eto diz: Eu não sei!
- **C**arla diz: Sim, todos queremos cerveja!

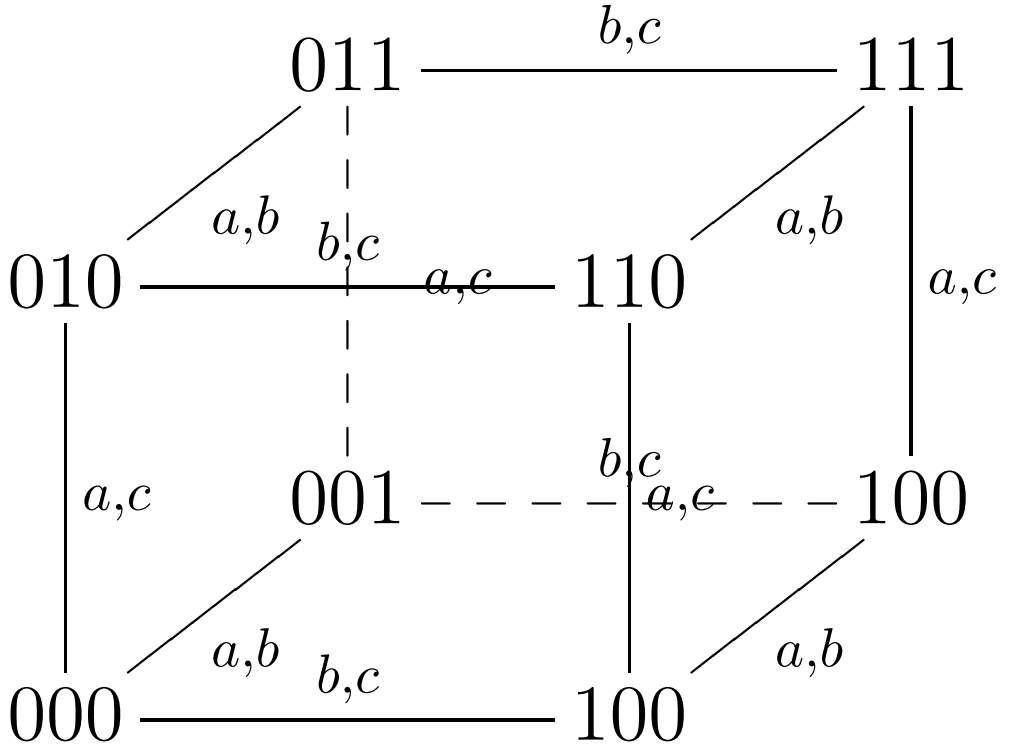
Quebra Cab. Lógicos Beberrões

- **Agentes**: **A**na, **B**eto e **C**arla
- Entram num bar.
- **Garçon**: Todos vão querer beber cerveja?
- **A**na diz: Eu não sei!
- **B**eto diz: Eu não sei!
- **C**arla diz: Sim, todos queremos cerveja!

- Como Carla soube???

Quebra Cab. Lógicos Beberrões

- Estado: notação posicional: **a b c**
- Ex.: **011 - a**na quer beber, **b**eto não e **c**arla não;



Procolo

- 3 Agentes: A, B e Z (malicioso)
- **Modelo de Chave Públicas** (Public Key Protocol)

- função de encriptação: E_X (publica), $X = \{A, B, Z\}$
- função de decriptação: D_X (só X sabe), $X = \{A, B, Z\}$
- $D_X E_X(M) = M$

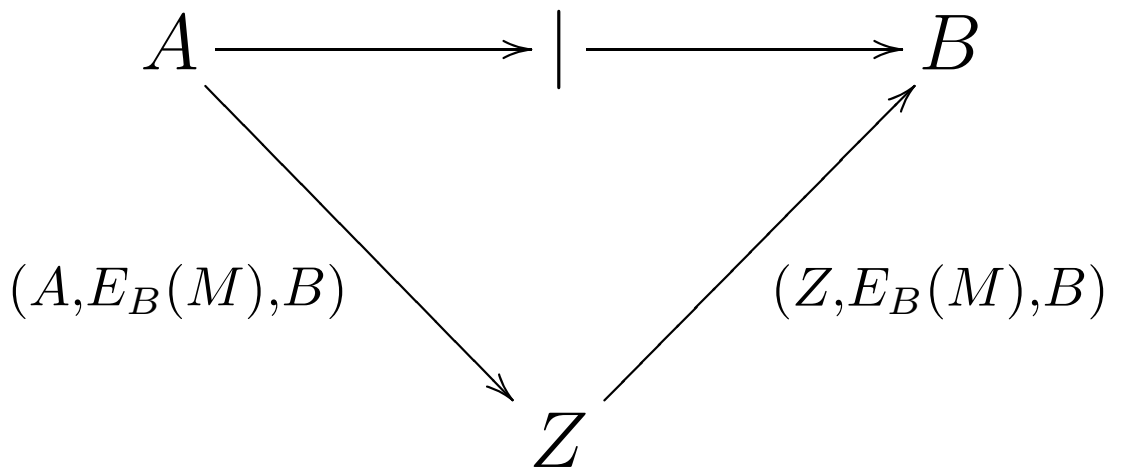
Protocolol

A manda msg M para B

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

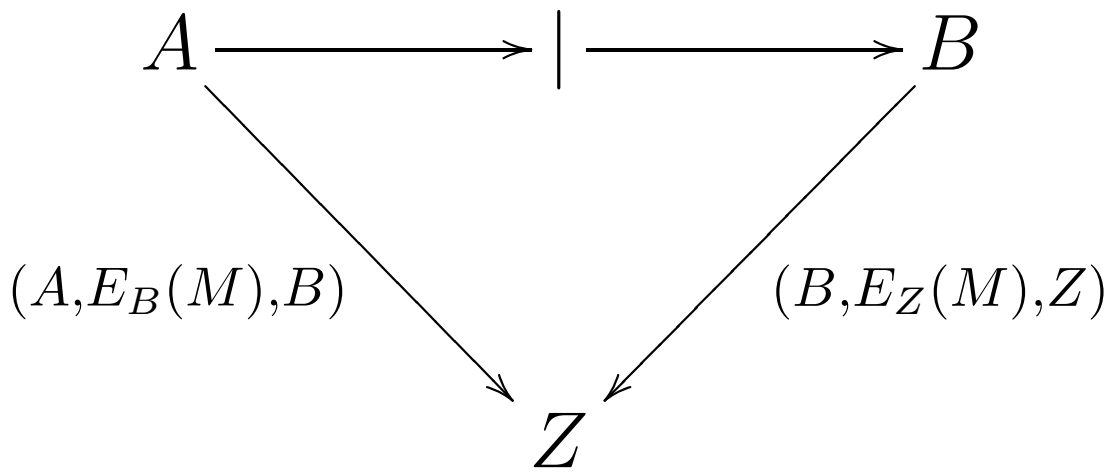
Z intercepta a msg de A para B

Z sends envia $(Z, E_B(M), B)$ para B



Protocolo

B envia msg $(B, E_Z(M), Z)$ para Z



Z decodifica $E_Z(M)$ e obtem M

Como Z raciocinou p/ quebrar o Protocolo?

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

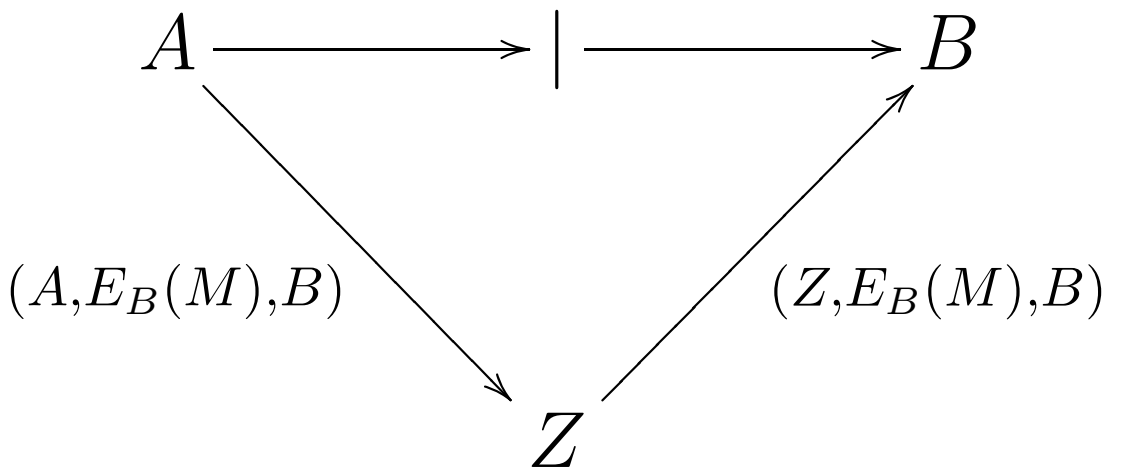
Example 1: Dolev & Yao Model

A sends msg M to B

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

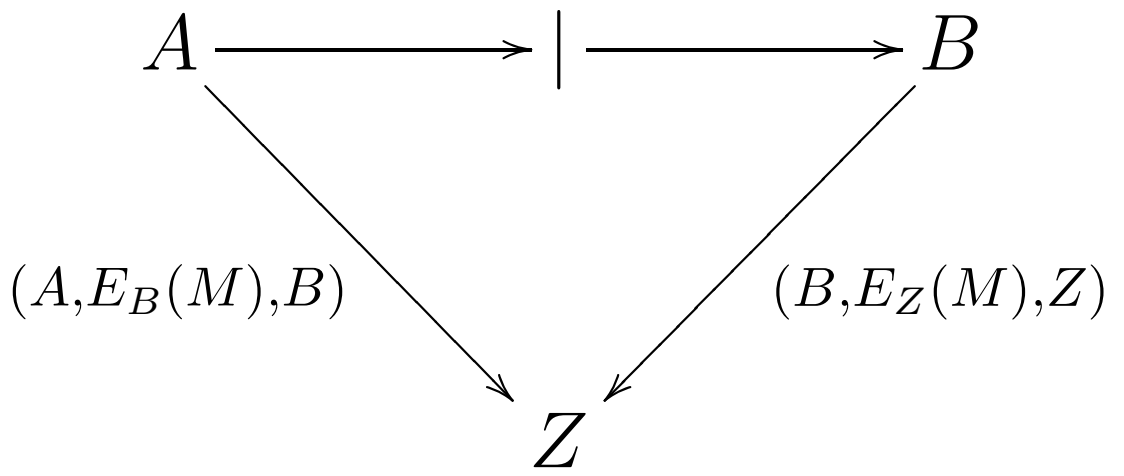
Intruder Z intercepts the message sent from A to B

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

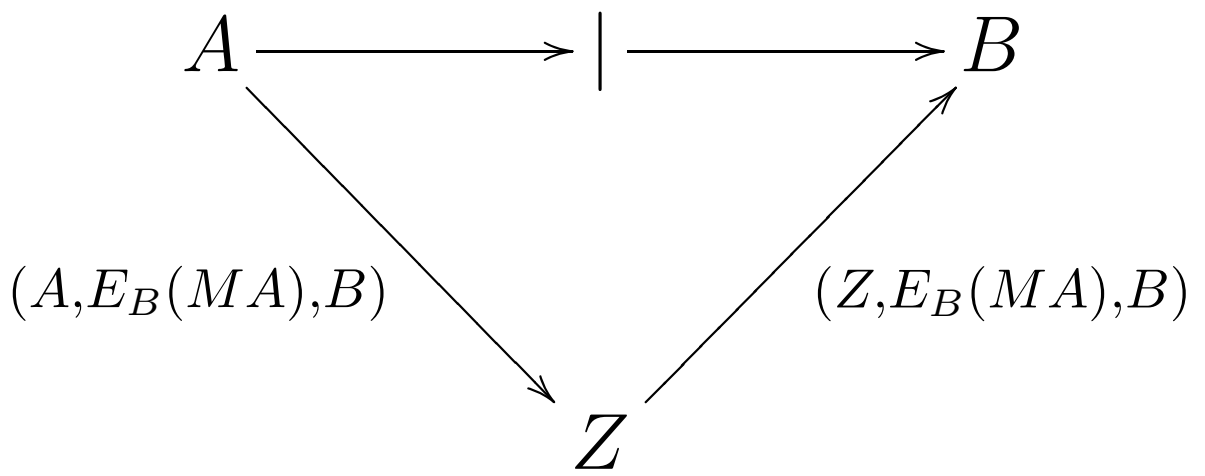
Example 2: Dolev & Yao Model

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

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

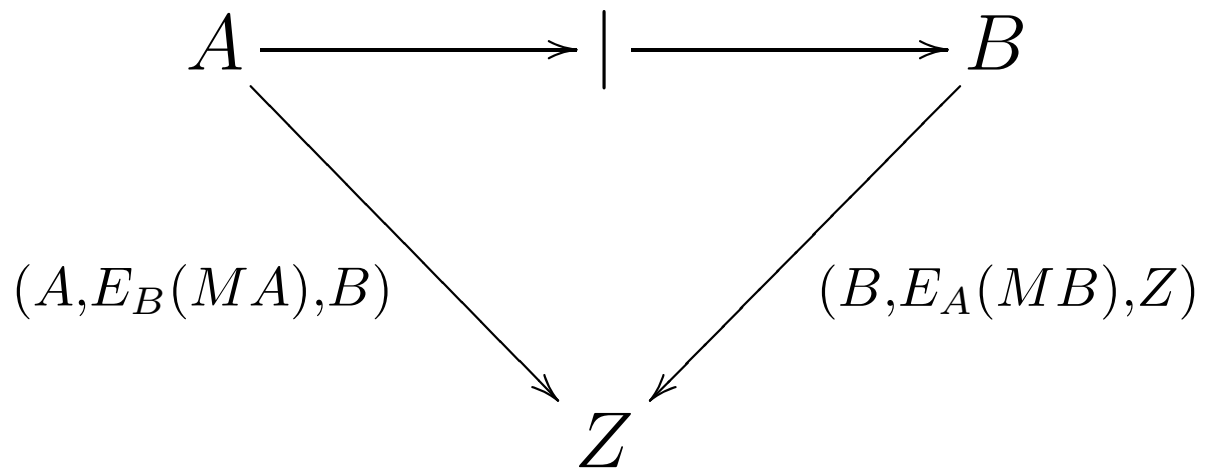
Intruder Z intercepts the message sent from A to B

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**

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

- 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

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

For public keys

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

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:

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

Logical Postulates - BAN Logic

Principal sees:

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

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

Logical Postulates - BAN Logic

Principal sees:

$$\frac{P \,\mathbf{believes}\, \overset{K}{\mapsto}\, P, \quad P \,\mathbf{sees}\, \{X\}_K}{P \,\mathbf{sees}\, X}$$

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

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

Quantifiers - BAN Logic

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

Quantifiers - BAN Logic

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

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

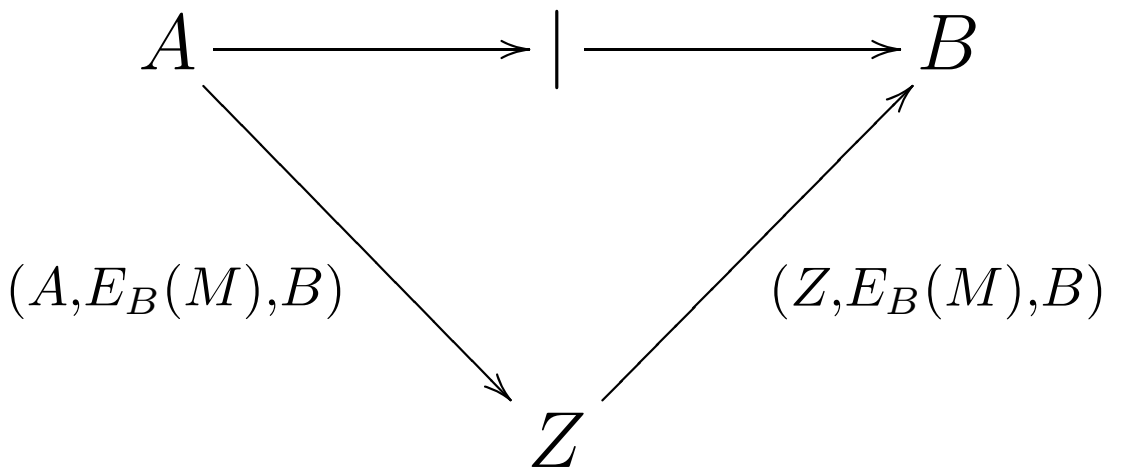
Example 1: Dolev & Yao Model

A sends msg M to B

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

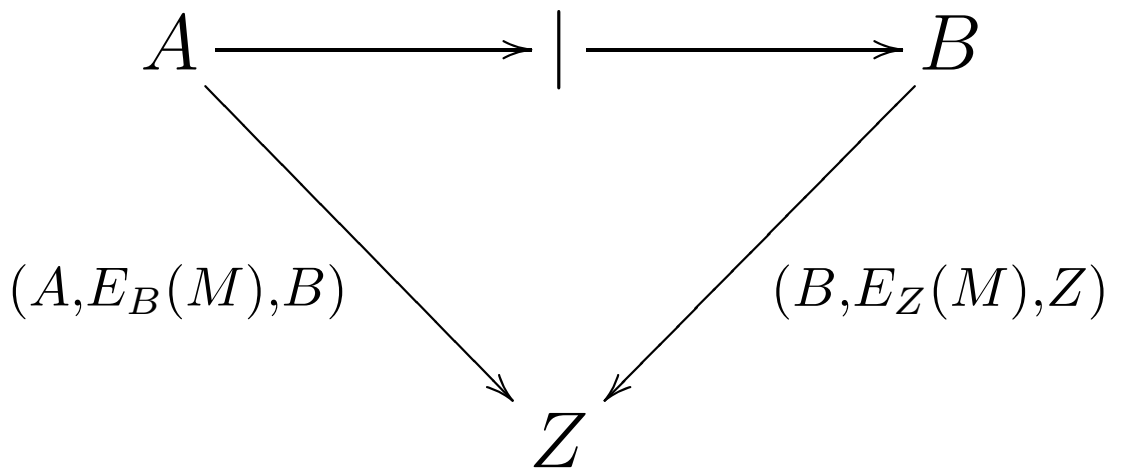
Intruder Z intercepts the message sent from A to B

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 - $\mathbf{S5_{DY}}$

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

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.

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

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 - $\mathbf{S5_{DY}}$

- 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,

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 - $\mathbf{S5_{DY}}$

- **Expressions**:

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

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

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

- **Expressions**:

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

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

- **Formulas**:

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

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

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

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

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

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

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

- **Frames**: $\mathbf{S5_{DY}}$ *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 - $\mathbf{S5_{DY}}$

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

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

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

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

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

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

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

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

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

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

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

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

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 - $\mathrm{S5_{DY}}$

- *All instantiations of propositional tautologies,*

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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 - $\mathbf{S5_{DY}}$

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

Properties

- Autheticity:

$$K_Z\{m\}_{k_X} \leftrightarrow \neg K_Z m, \quad \text{for all } X \neq Z$$

Properties

- Autheticity:

  $$K_Z\{m\}_{k_X} \leftrightarrow \neg K_Z m, \quad \text{for all } X \neq Z$$

- Secrecy:

  $$\neg K_Z k_A \wedge K_Z\{m\}_{k_A} \rightarrow \neg K_Z m, \quad \text{for all } A \neq Z$$

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

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

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

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

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

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

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

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

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

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

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

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

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

Complexity

- **To be donne!**

Complexity

- **To be donne!**
- Hints for $\mathbf{S5_{DY}}$

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 $\mathbf{S5_{DY}^{CK}}$
- $\mathbf{S5^{CK}}$
  - **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 $\mathbf{S5_{DY}^{CK}}$
- $\mathbf{S5^{CK}}$
  - **Model Checking**: Polynomial
  - **Validity**: EXPTIME-Complete

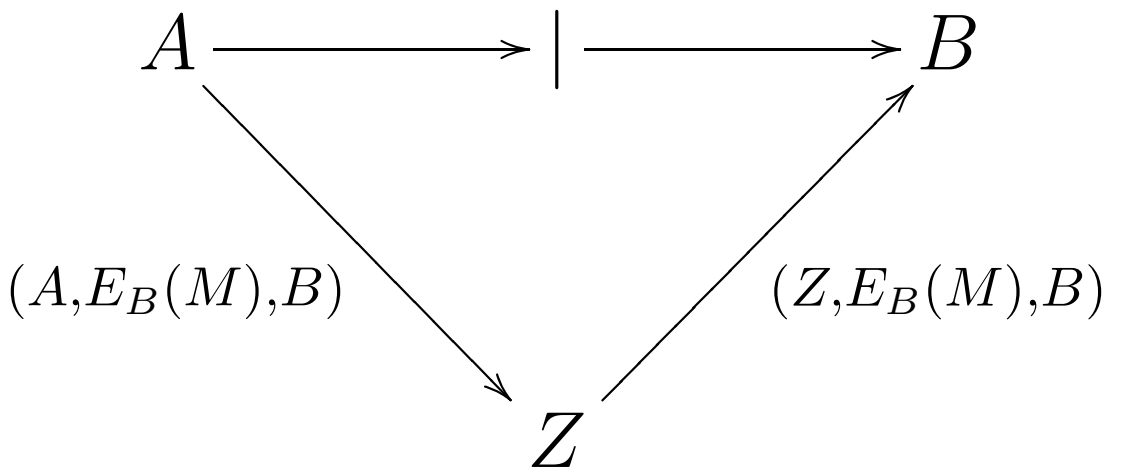
Example 1: Dolev & Yao - Cont.

A sends msg M to B

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

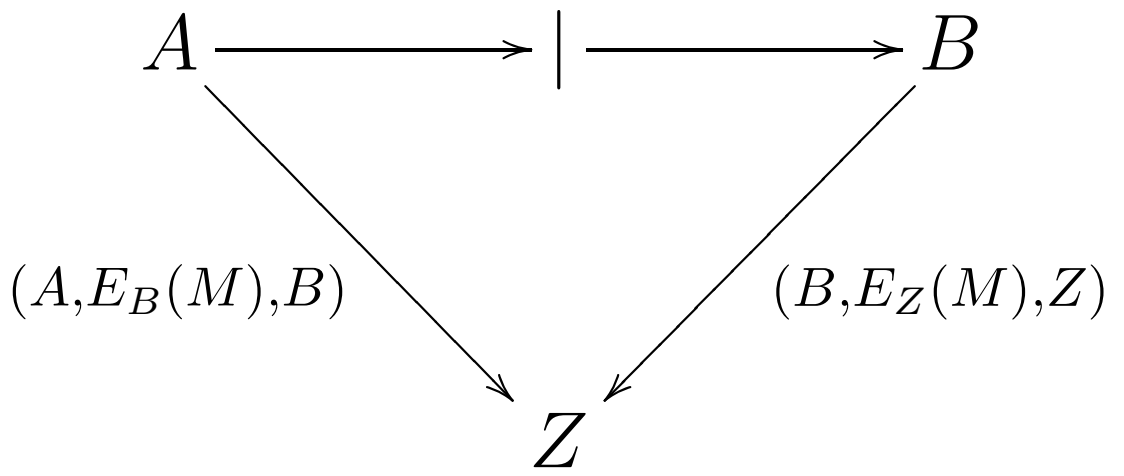
Intruder Z intercepts the message sent from A to B

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

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

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

$$- - -$$

$$\downarrow Z\ intecepts$$

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

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

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

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

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

Proving Example 1

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

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

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

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

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

$$K_Z m \quad ax.7$$

Intruder Z knows M

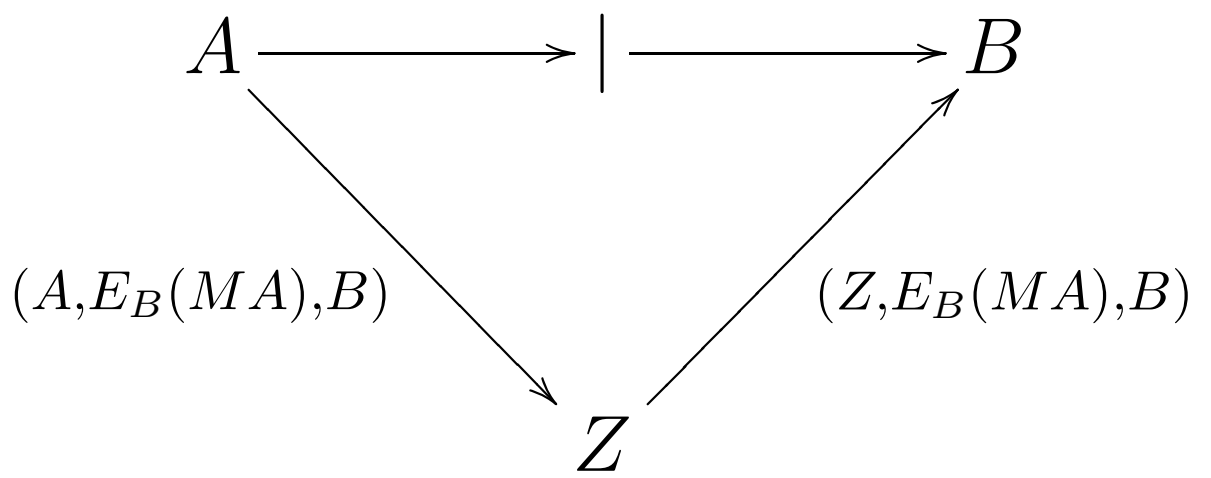
Example 2: Dolev & Yao Model

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

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

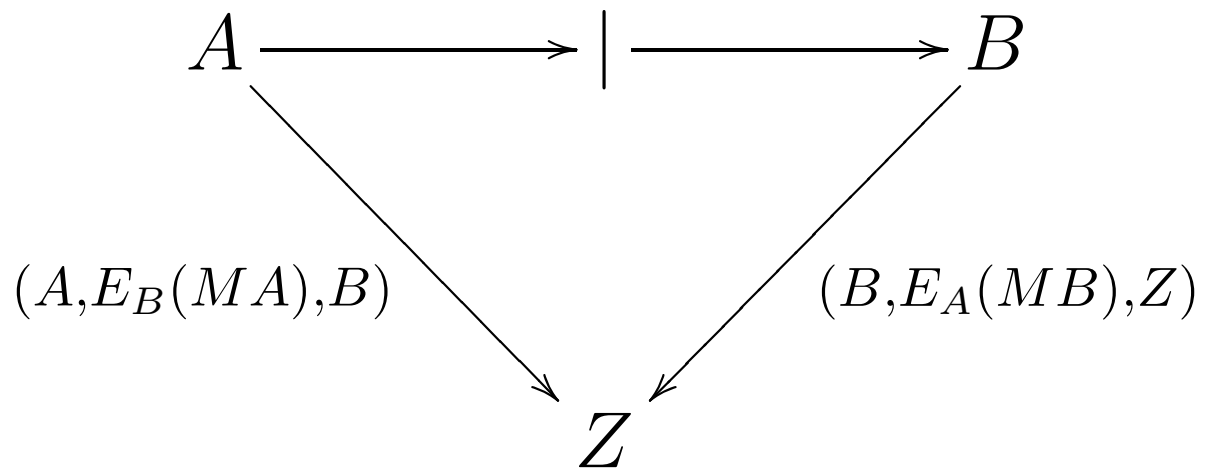
Intruder Z intercepts the message sent from A to B

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

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

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

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

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

$$- \ - \ -$$

$$\Big\downarrow {\scriptstyle Z\ intecepts}$$

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

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

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

Proving Example 2

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

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

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

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

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

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

$$KB_3 \nvdash K_Z m$$

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 $\mathbf{S5^{A}_{DY}}$

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

Knowledge De re and De Dicto

- What is the meaning of $K_a\{m\}_K$?

Knowledge De re and De Dicto

- What is the meaning of $K_a\{m\}_K$?
- If agent a does not know the key K?

Knowledge De re and De Dicto

- What is the meaning of $K_a\{m\}_K$?
- If agent a does not know the key K?
- It would be more appropriated to say "agent a has the peace of information $\{m\}_K$ ".

Knowledge De re and De Dicto

- What is the meaning of $K_a\{m\}_K$?
- If agent a does not know the key K?
- It would be more appropriated to say "agent a has the peace of information $\{m\}_K$ ".
- If agent a does know the key K?

Knowledge De re and De Dicto

- What is the meaning of $K_a\{m\}_K$?
- If agent a does not know the key K?
- It would be more appropriated to say "agent a has the peace of information $\{m\}_K$ ".
- If agent a does know the key K?
- Is there any difference of knowing something and knowing the content of something?

An Extension of the $S5_{DY}$ Logic

- **Message language**:

$$M \ ::= a \mid k \mid (M, M) \mid \{M\}_k$$

An Extension of the $S5_{DY}$ Logic

- **Message language**:

$$M \ ::= a \mid k \mid (M, M) \mid \{M\}_k$$

- we allow an agent's name a in the definition of a message so that it can be appended to a message M to obtain the (signed) message (M, a), as it is used in some of the examples.

An Extension of the $S5_{DY}$ Logic

- **Message language**:

$$M \ ::= a \mid k \mid (M, M) \mid \{M\}_k$$

- we allow an agent's name a in the definition of a message so that it can be appended to a message M to obtain the (signed) message (M, a), as it is used in some of the examples.

- **Language**:

$$\alpha \ ::= p \mid \neg\alpha \mid \alpha \wedge \alpha \mid K_a\alpha \mid a\mathsf{k}M$$

An Extension of the $S5_{DY}$ Logic

- **Language**:

$$\alpha \ ::= p \mid \neg\alpha \mid \alpha \wedge \alpha \mid K_a\alpha \mid a\mathsf{k}M$$

An Extension of the $S5_{DY}$ Logic

- **Language**:

$$\alpha \ ::= p \mid \neg\alpha \mid \alpha \wedge \alpha \mid K_a\alpha \mid a\mathsf{k}M$$

- The intuition of a proposition $a\mathsf{k}M$ is that it captures the notion of knowledge *de re* that a has of M i.e., $a\mathsf{k}M$ denotes the fact that agent a *knows the content of* M.

An Extension of the $S5_{DY}$ Logic

- **Language**:

$$\alpha \ ::= p \mid \neg\alpha \mid \alpha \wedge \alpha \mid K_a\alpha \mid a\mathsf{k}M$$

- The intuition of a proposition $a\mathsf{k}M$ is that it captures the notion of knowledge *de re* that a has of M i.e., $a\mathsf{k}M$ denotes the fact that agent a *knows the content of* M.

- The modal operator K_a is meant to capture the standard notion of knowledge *de dicto* that a has about a propositional sentence.

An Extension of the $S5_{DY}$ Logic

- **Language**:

$$\alpha \ ::= p \mid \neg\alpha \mid \alpha \wedge \alpha \mid K_a\alpha \mid a\mathsf{k}M$$

- The intuition of a proposition $a\mathsf{k}M$ is that it captures the notion of knowledge *de re* that a has of M i.e., $a\mathsf{k}M$ denotes the fact that agent a *knows the content of* M.

- The modal operator K_a is meant to capture the standard notion of knowledge *de dicto* that a has about a propositional sentence.

Axiomatization

- 2.$a\mathrm{k}a$ (every agent knows its own name)

Axiomatization

- 2.$a\mathsf{k}a$ (every agent knows its own name)
- 3. $a\mathsf{k}M \wedge a\mathsf{k}M' \leftrightarrow a\mathsf{k}(M, M')$ (pairing)

Axiomatization

- 2.$a\mathsf{k}a$ (every agent knows its own name)
- 3. $a\mathsf{k}M \wedge a\mathsf{k}M' \leftrightarrow a\mathsf{k}(M, M')$ (pairing)
- 4. $a\mathsf{k}M \wedge a\mathsf{k}k \rightarrow a\mathsf{k}\{M\}_k$ (encryption)

Axiomatization

- 2.$a\mathsf{k}a$ (every agent knows its own name)
- 3. $a\mathsf{k}M \wedge a\mathsf{k}M' \leftrightarrow a\mathsf{k}(M, M')$ (pairing)
- 4. $a\mathsf{k}M \wedge a\mathsf{k}k \rightarrow a\mathsf{k}\{M\}_k$ (encryption)
- 5. $a\mathsf{k}\{M\}_k \wedge a\mathsf{k}k \rightarrow a\mathsf{k}M$ (decryption)

Axiomatization

- 2.$a\mathsf{k}a$ (every agent knows its own name)
- 3. $a\mathsf{k}M \wedge a\mathsf{k}M' \leftrightarrow a\mathsf{k}(M, M')$ (pairing)
- 4. $a\mathsf{k}M \wedge a\mathsf{k}k \rightarrow a\mathsf{k}\{M\}_k$ (encryption)
- 5. $a\mathsf{k}\{M\}_k \wedge a\mathsf{k}k \rightarrow a\mathsf{k}M$ (decryption)
- 6. $K_a\alpha \wedge K_a(\alpha \rightarrow \beta) \rightarrow K_a\beta$

Axiomatization

- 2.$a\mathsf{k}a$ (every agent knows its own name)
- 3. $a\mathsf{k}M \wedge a\mathsf{k}M' \leftrightarrow a\mathsf{k}(M, M')$ (pairing)
- 4. $a\mathsf{k}M \wedge a\mathsf{k}k \rightarrow a\mathsf{k}\{M\}_k$ (encryption)
- 5. $a\mathsf{k}\{M\}_k \wedge a\mathsf{k}k \rightarrow a\mathsf{k}M$ (decryption)
- 6. $K_a\alpha \wedge K_a(\alpha \rightarrow \beta) \rightarrow K_a\beta$
- 7. $K_a\alpha \rightarrow \alpha$

Axiomatization

- 2.$a\mathsf{k}a$ (every agent knows its own name)
- 3. $a\mathsf{k}M \wedge a\mathsf{k}M' \leftrightarrow a\mathsf{k}(M, M')$ (pairing)
- 4. $a\mathsf{k}M \wedge a\mathsf{k}k \rightarrow a\mathsf{k}\{M\}_k$ (encryption)
- 5. $a\mathsf{k}\{M\}_k \wedge a\mathsf{k}k \rightarrow a\mathsf{k}M$ (decryption)
- 6. $K_a\alpha \wedge K_a(\alpha \rightarrow \beta) \rightarrow K_a\beta$
- 7. $K_a\alpha \rightarrow \alpha$
- 8.$K_a\alpha \rightarrow K_aK_a\alpha$

Axiomatization

- 2.$a\mathsf{k}a$ (every agent knows its own name)
- 3. $a\mathsf{k}M \wedge a\mathsf{k}M' \leftrightarrow a\mathsf{k}(M, M')$ (pairing)
- 4. $a\mathsf{k}M \wedge a\mathsf{k}k \rightarrow a\mathsf{k}\{M\}_k$ (encryption)
- 5. $a\mathsf{k}\{M\}_k \wedge a\mathsf{k}k \rightarrow a\mathsf{k}M$ (decryption)
- 6. $K_a\alpha \wedge K_a(\alpha \rightarrow \beta) \rightarrow K_a\beta$
- 7. $K_a\alpha \rightarrow \alpha$
- 8.$K_a\alpha \rightarrow K_aK_a\alpha$
- 9.$\neg K_a\alpha \rightarrow K_a\neg K_a\alpha$

Axiomatization

- 2.$a\mathsf{k}a$ (every agent knows its own name)
- 3. $a\mathsf{k}M \wedge a\mathsf{k}M' \leftrightarrow a\mathsf{k}(M, M')$ (pairing)
- 4. $a\mathsf{k}M \wedge a\mathsf{k}k \rightarrow a\mathsf{k}\{M\}_k$ (encryption)
- 5. $a\mathsf{k}\{M\}_k \wedge a\mathsf{k}k \rightarrow a\mathsf{k}M$ (decryption)
- 6. $K_a\alpha \wedge K_a(\alpha \rightarrow \beta) \rightarrow K_a\beta$
- 7. $K_a\alpha \rightarrow \alpha$
- 8.$K_a\alpha \rightarrow K_aK_a\alpha$
- 9.$\neg K_a\alpha \rightarrow K_a\neg K_a\alpha$
- 10. $a\mathsf{k}M \rightarrow K_aa\mathsf{k}M$ (+ *de re* introspection)

Axiomatization

- 2.$a\mathsf{k}a$ (every agent knows its own name)
- 3. $a\mathsf{k}M \wedge a\mathsf{k}M' \leftrightarrow a\mathsf{k}(M, M')$ (pairing)
- 4. $a\mathsf{k}M \wedge a\mathsf{k}k \rightarrow a\mathsf{k}\{M\}_k$ (encryption)
- 5. $a\mathsf{k}\{M\}_k \wedge a\mathsf{k}k \rightarrow a\mathsf{k}M$ (decryption)
- 6. $K_a\alpha \wedge K_a(\alpha \rightarrow \beta) \rightarrow K_a\beta$
- 7. $K_a\alpha \rightarrow \alpha$
- 8.$K_a\alpha \rightarrow K_aK_a\alpha$
- 9.$\neg K_a\alpha \rightarrow K_a\neg K_a\alpha$
- 10. $a\mathsf{k}M \rightarrow K_aa\mathsf{k}M$ (+ *de re* introspection)
- 11. $\neg(a\mathsf{k}M) \rightarrow K_a\neg(a\mathsf{k}M)$ (- *de re* introsp.)

Axiomatization

- 2.$a\mathsf{k}a$ (every agent knows its own name)
- 3. $a\mathsf{k}M \wedge a\mathsf{k}M' \leftrightarrow a\mathsf{k}(M, M')$ (pairing)
- 4. $a\mathsf{k}M \wedge a\mathsf{k}k \rightarrow a\mathsf{k}\{M\}_k$ (encryption)
- 5. $a\mathsf{k}\{M\}_k \wedge a\mathsf{k}k \rightarrow a\mathsf{k}M$ (decryption)
- 6. $K_a\alpha \wedge K_a(\alpha \rightarrow \beta) \rightarrow K_a\beta$
- 7. $K_a\alpha \rightarrow \alpha$
- 8.$K_a\alpha \rightarrow K_aK_a\alpha$
- 9.$\neg K_a\alpha \rightarrow K_a\neg K_a\alpha$
- 10. $a\mathsf{k}M \rightarrow K_aa\mathsf{k}M$ (+ *de re* introspection)
- 11. $\neg(a\mathsf{k}M) \rightarrow K_a\neg(a\mathsf{k}M)$ (- *de re* introsp.)
- 12. Maybe more?

Axiomatization

- 2.$a\mathsf{k}a$ (every agent knows its own name)
- 3. $a\mathsf{k}M \wedge a\mathsf{k}M' \leftrightarrow a\mathsf{k}(M, M')$ (pairing)
- 4. $a\mathsf{k}M \wedge a\mathsf{k}k \rightarrow a\mathsf{k}\{M\}_k$ (encryption)
- 5. $a\mathsf{k}\{M\}_k \wedge a\mathsf{k}k \rightarrow a\mathsf{k}M$ (decryption)
- 6. $K_a\alpha \wedge K_a(\alpha \rightarrow \beta) \rightarrow K_a\beta$
- 7. $K_a\alpha \rightarrow \alpha$
- 8.$K_a\alpha \rightarrow K_aK_a\alpha$
- 9.$\neg K_a\alpha \rightarrow K_a\neg K_a\alpha$
- 10. $a\mathsf{k}M \rightarrow K_aa\mathsf{k}M$ (+ *de re* introspection)
- 11. $\neg(a\mathsf{k}M) \rightarrow K_a\neg(a\mathsf{k}M)$ (- *de re* introsp.)
- 12. Maybe more?

Semantics

- should be the standard semantics of epistemic logic enriched with a treatment of $a\mathsf{k}M$ propositions.

Semantics

- should be the standard semantics of epistemic logic enriched with a treatment of $a\mathsf{k}M$ propositions.

- How???

Authorization

- Communication actions: says, speaks for, controls, sees,...

Authorization

- Communication actions: says, speaks for, controls, sees,...

- Example:

    1. If admin says that file1 should be deleted, then this must be the case.
    2. admin trusts Bob to decide whether file1 should be deleted.
    3. Bob wants to delete file1.

Authorization

- Communication actions: says, speaks for, controls, sees,...

- Example:

  1. If admin says that file1 should be deleted, then this must be the case.

  2. admin trusts Bob to decide whether file1 should be deleted.

  3. Bob wants to delete file1.

- Modal Operators $S_a\varphi$: agent a says φ

Authorization

- Communication actions: says, speaks for, controls, sees,...

- Example:
  1. If admin says that file1 should be deleted, then this must be the case.
  2. admin trusts Bob to decide whether file1 should be deleted.
  3. Bob wants to delete file1.

- Modal Operators $S_a\varphi$: agent a says φ
- Example:
  1. $S_{adm}del.file.1 \rightarrow del.file.1$
  2. $S_{adm}(S_{bob}del.file.1) \rightarrow del.file.1$
  3. $S_{bob}del.file.1$

Authorization

- Communication actions: says, speaks for, controls, sees,...

- Example:
  1. If admin says that file1 should be deleted, then this must be the case.
  2. admin trusts Bob to decide whether file1 should be deleted.
  3. Bob wants to delete file1.

- Modal Operators $S_a\varphi$: agent a says φ
- Example:
  1. $S_{adm}del.file.1 \rightarrow del.file.1$
  2. $S_{adm}(S_{bob}del.file.1) \rightarrow del.file.1$
  3. $S_{bob}del.file.1$

Properties of says

- $\varphi \rightarrow S_a \varphi$

Properties of says

- $\varphi \rightarrow S_a \varphi$
- $S_a \varphi \rightarrow \varphi$ (???)

Properties of says

- $\varphi \rightarrow S_a\varphi$
- $S_a\varphi \rightarrow \varphi$ (???)
- $K_a\varphi \rightarrow S_a\varphi$

Properties of says

- $\varphi \rightarrow S_a\varphi$
- $S_a\varphi \rightarrow \varphi$ (???)
- $K_a\varphi \rightarrow S_a\varphi$
- $S_a\varphi \rightarrow K_a\varphi$ (???)

Properties of says

- $\varphi \rightarrow S_a\varphi$
- $S_a\varphi \rightarrow \varphi$ (???)
- $K_a\varphi \rightarrow S_a\varphi$
- $S_a\varphi \rightarrow K_a\varphi$ (???)
- $S_aS_a\varphi \rightarrow S_a\varphi$

Properties of says

- $\varphi \rightarrow S_a\varphi$
- $S_a\varphi \rightarrow \varphi$ (???)
- $K_a\varphi \rightarrow S_a\varphi$
- $S_a\varphi \rightarrow K_a\varphi$ (???)
- $S_aS_a\varphi \rightarrow S_a\varphi$
- $(\varphi \rightarrow S_a\psi) \rightarrow (S_a\varphi \rightarrow S_a\psi)$

Properties of says

- $\varphi \rightarrow S_a\varphi$
- $S_a\varphi \rightarrow \varphi$ (???)
- $K_a\varphi \rightarrow S_a\varphi$
- $S_a\varphi \rightarrow K_a\varphi$ (???)
- $S_aS_a\varphi \rightarrow S_a\varphi$
- $(\varphi \rightarrow S_a\psi) \rightarrow (S_a\varphi \rightarrow S_a\psi)$
- $S_a\varphi \rightarrow B_b\varphi$, for all b???

Properties of says

- $\varphi \rightarrow S_a\varphi$
- $S_a\varphi \rightarrow \varphi$ (???)
- $K_a\varphi \rightarrow S_a\varphi$
- $S_a\varphi \rightarrow K_a\varphi$ (???)
- $S_aS_a\varphi \rightarrow S_a\varphi$
- $(\varphi \rightarrow S_a\psi) \rightarrow (S_a\varphi \rightarrow S_a\psi)$
- $S_a\varphi \rightarrow B_b\varphi$, for all b???
- $\vdots$

Properties of says

- $\varphi \rightarrow S_a\varphi$
- $S_a\varphi \rightarrow \varphi$ (???)
- $K_a\varphi \rightarrow S_a\varphi$
- $S_a\varphi \rightarrow K_a\varphi$ (???)
- $S_aS_a\varphi \rightarrow S_a\varphi$
- $(\varphi \rightarrow S_a\psi) \rightarrow (S_a\varphi \rightarrow S_a\psi)$
- $S_a\varphi \rightarrow B_b\varphi$, for all b???
- $\vdots$
- There are some logics to express **says**. Most of them are extensions of Intuicionistic Logic.

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