UNIVERSITÄT PASSAU

Fakultät für Informatik und Mathematik

6090: Security of Computer and Embedded Systems

Week 10: Formal Analysis of Security Protocols

Elif Bilge Kavun

elif.kavun@uni-passau.de

December 21, 2021

This Week's Outline

- Formal Methods
- Syntax and Semantics
  - Alice and Bob
  - Roles
- The Dolev-Yao-Style Intruder

What Have We Seen So Far?

In the last lectures, we have seen that…

Designing correct security protocols is difficult!

What Are Formal Methods?

What Are Formal Methods?

- A language is formal
  - If it has a well-defined syntax and semantics
  - There is also often a deductive system for determining the truth of statements
- A model (or construction) is formal
  - If it is specified in a formal language

What Are Formal Methods?

- A language is formal
  - If it has a well-defined syntax and semantics
  - There is also often a deductive system for determining the truth of statements
- A model (or construction) is formal
  - If it is specified in a formal language

- Standard protocol notation is **not** formal
  - This session is about how to formalize such notations

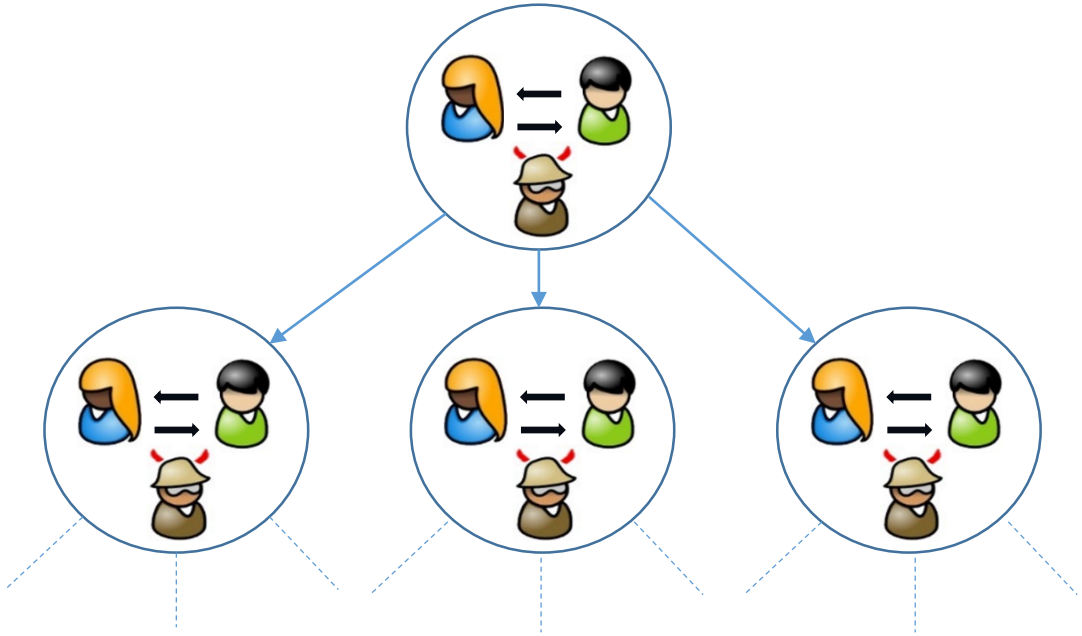
Formal Modeling and Analysis of Protocols

Goal

- Formally model protocols & their properties and provide a mathematically sound means for reasoning about these models

Basis

- Suitable abstraction of protocols



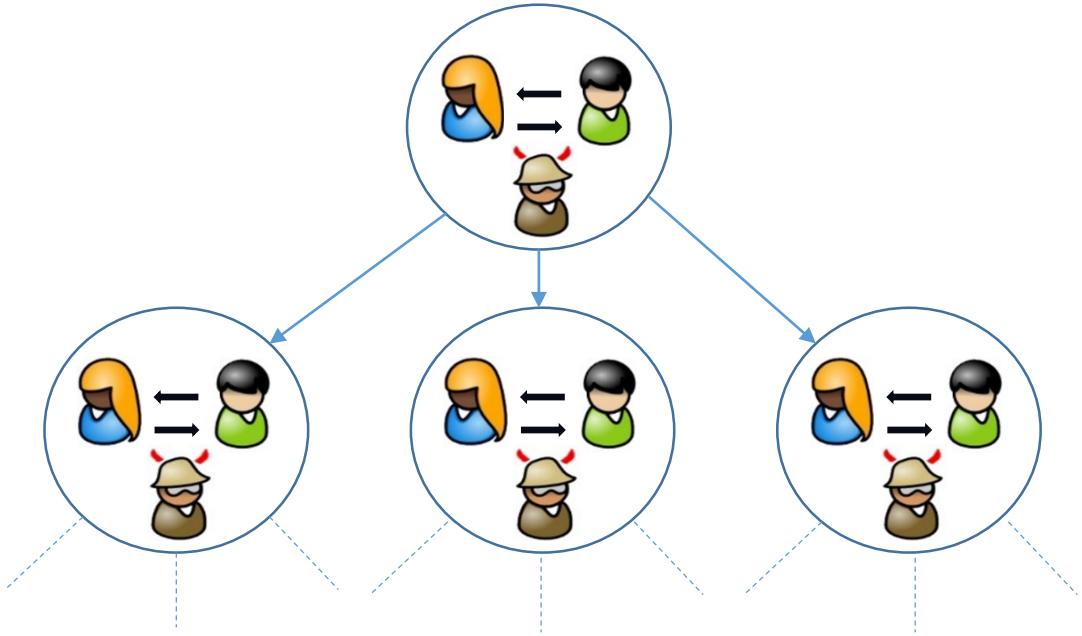
Formal Modeling and Analysis of Protocols

Goal

- Formally model protocols & their properties and provide a mathematically sound means for reasoning about these models

Basis

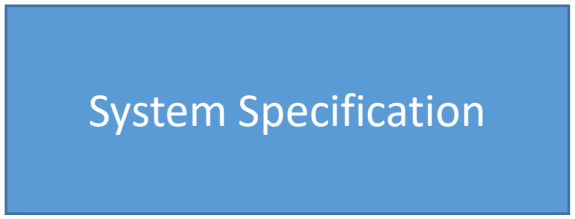
- Suitable abstraction of protocols



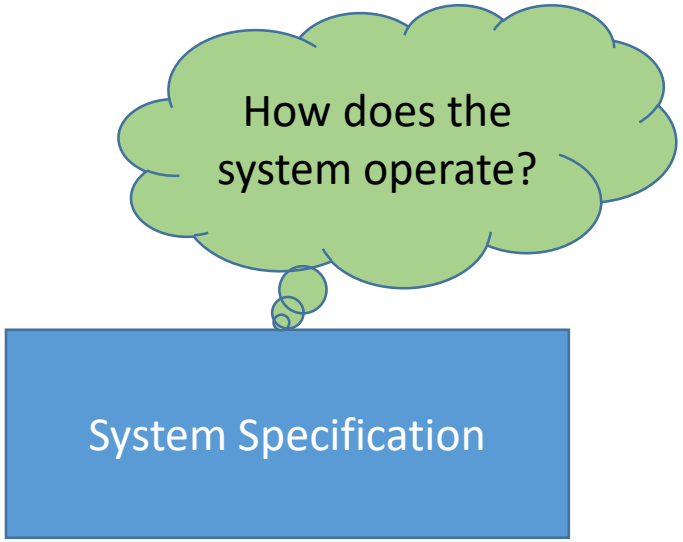
Analysis

- Proceeds with formal methods based on mathematics and logic

Formal Methods



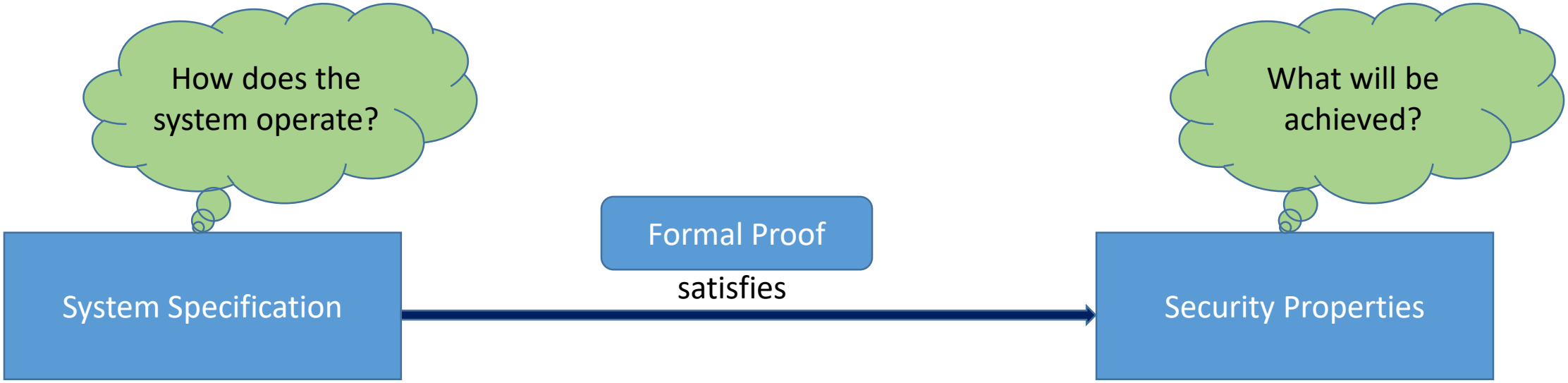
Formal Methods



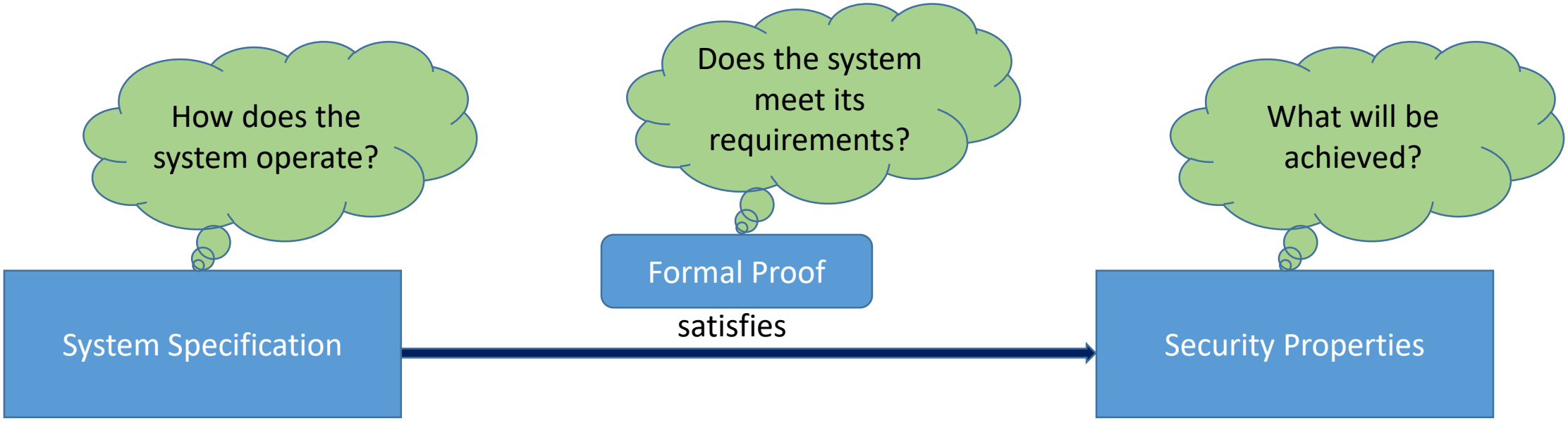
Formal Methods

Formal Methods

Formal Methods

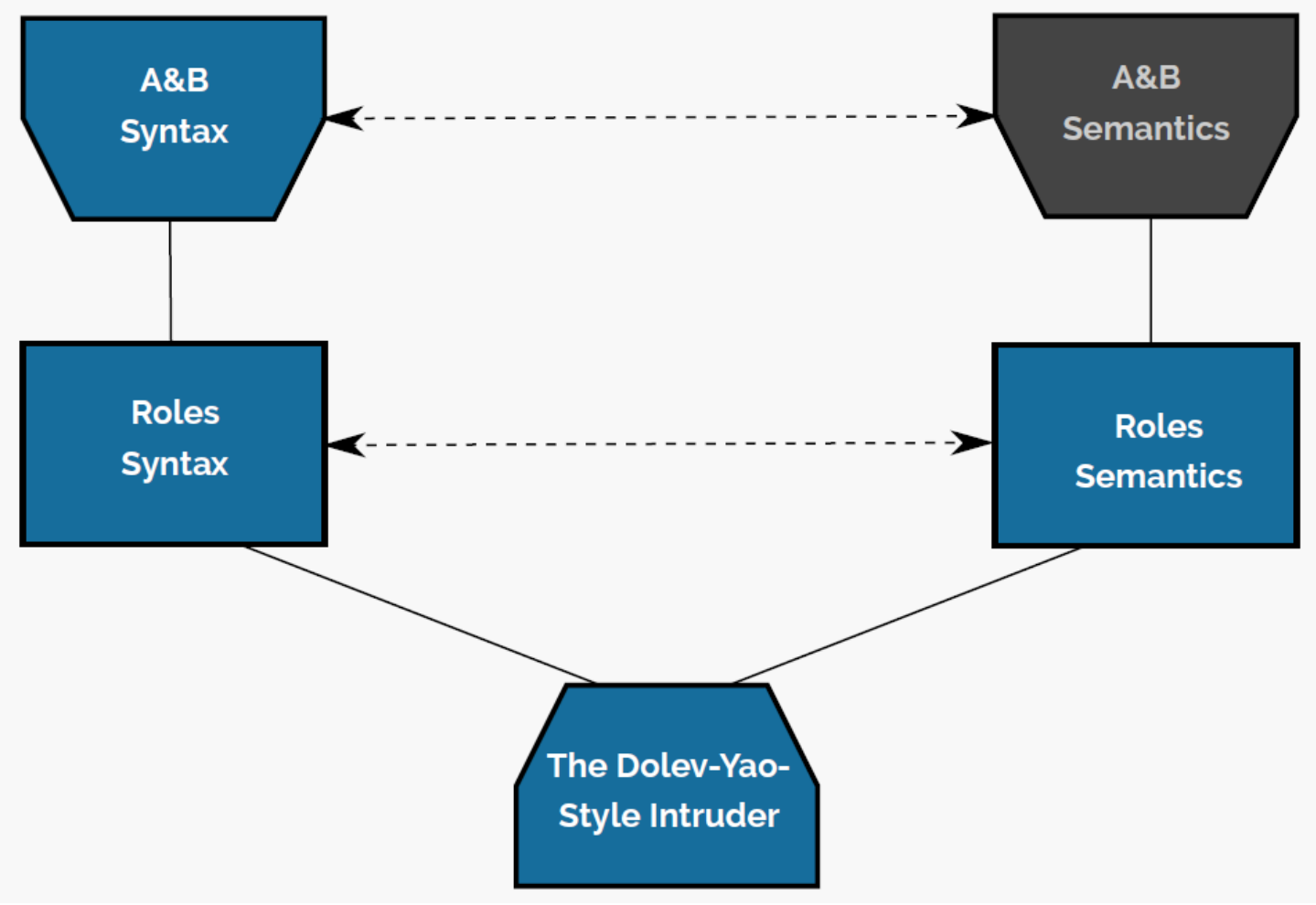


Formal Methods



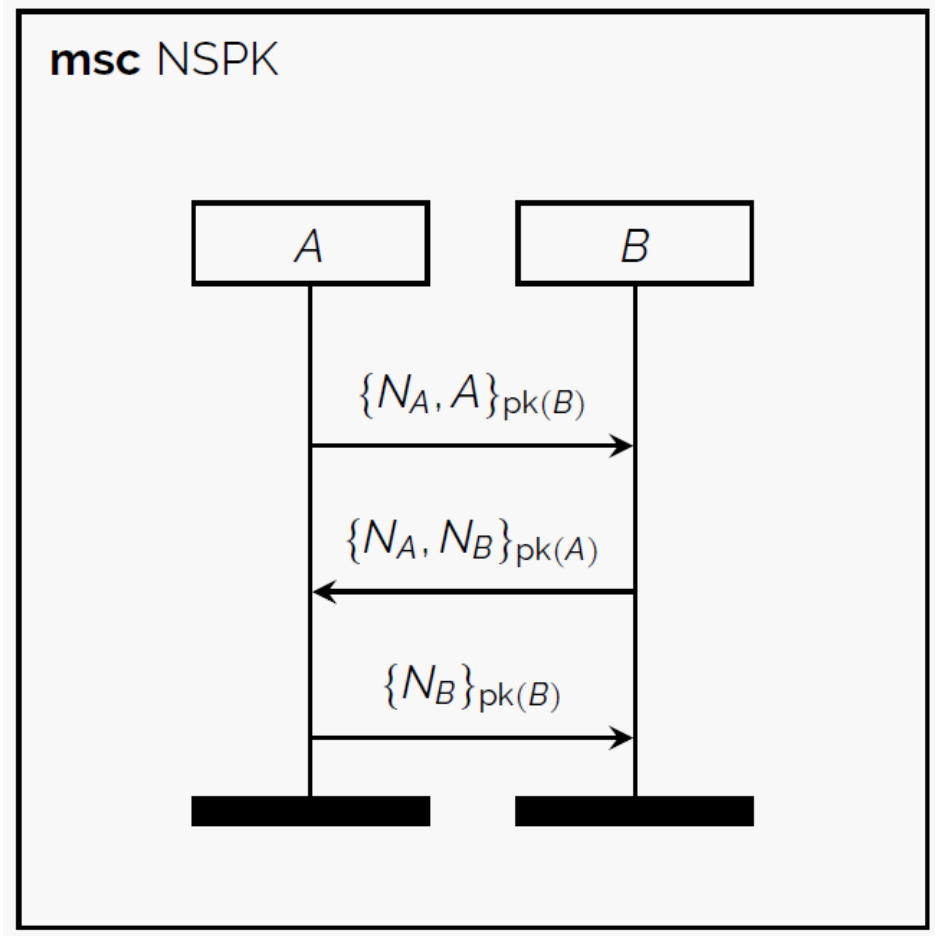
Alice & Bob, Roles, Intruder

Overview



Alice and Bob: Syntax

- Alice & Bob (AnB) Notation: Message Sequence Charts (msc)



A Formal Alice & Bob Language: Syntax

Actions:

$$A \longrightarrow B : \{\mathrm{N}_A, A\}_{\mathrm{pk}(B)}$$
$$B \longrightarrow A : \{\mathrm{N}_A, \mathrm{N}_B\}_{\mathrm{pk}(A)}$$
$$A \longrightarrow B : \{\mathrm{N}_B\}_{\mathrm{pk}(B)}$$

A Formal Alice & Bob Language: Syntax

Actions: $A \longrightarrow B : \{\mathrm{N}_A, A\}_{\mathrm{pk}(B)}$ $B \longrightarrow A : \{\mathrm{N}_A, \mathrm{N}_B\}_{\mathrm{pk}(A)}$ $A \longrightarrow B : \{\mathrm{N}_B\}_{\mathrm{pk}(B)}$	Actions: The exchanged messages

A Formal Alice & Bob Language: Syntax

Protocol: NSPK	
Actions: $A \longrightarrow B : \{\mathrm{N}_A, A\}_{\mathrm{pk}(B)}$ $B \longrightarrow A : \{\mathrm{N}_A, \mathrm{N}_B\}_{\mathrm{pk}(A)}$ $A \longrightarrow B : \{\mathrm{N}_B\}_{\mathrm{pk}(B)}$	Actions: The exchanged messages

A Formal Alice & Bob Language: Syntax

Protocol: NSPK	Protocol: Name of the protocol
Actions: $A \longrightarrow B : \{\mathrm{N}_A, A\}_{\mathrm{pk}(B)}$ $B \longrightarrow A : \{\mathrm{N}_A, \mathrm{N}_B\}_{\mathrm{pk}(A)}$ $A \longrightarrow B : \{\mathrm{N}_B\}_{\mathrm{pk}(B)}$	Actions: The exchanged messages

A Formal Alice & Bob Language: Syntax

Protocol: NSPK	Protocol: Name of the protocol
Types: Agent: A, B; Number:$\mathrm{N}_A, \mathrm{N}_B$; Function: pk;	
Actions: $A \longrightarrow B : \{\mathrm{N}_A, A\}_{\mathrm{pk}(B)}$ $B \longrightarrow A : \{\mathrm{N}_A, \mathrm{N}_B\}_{\mathrm{pk}(A)}$ $A \longrightarrow B : \{\mathrm{N}_B\}_{\mathrm{pk}(B)}$	Actions: The exchanged messages

A Formal Alice & Bob Language: Syntax

Protocol: NSPK	Protocol: Name of the protocol
Types: Agent: A, B; Number: $\mathrm{N}_A, \mathrm{N}_B$; Function: pk;	Types: Types of all identifiers (may not be considered in the analysis)
Actions: $A \longrightarrow B : \{\mathrm{N}_A, A\}_{\mathrm{pk}(B)}$ $B \longrightarrow A : \{\mathrm{N}_A, \mathrm{N}_B\}_{\mathrm{pk}(A)}$ $A \longrightarrow B : \{\mathrm{N}_B\}_{\mathrm{pk}(B)}$	Actions: The exchanged messages

A Formal Alice & Bob Language: Syntax

Protocol: NSPK	Protocol: Name of the protocol
Types: Agent: A, B; Number: $\mathrm{N}_A, \mathrm{N}_B$; Function: pk;	Types: Types of all identifiers (may not be considered in the analysis)
Knowledge: A: $A, B, \mathrm{pk}, \mathrm{inv}(\mathrm{pk}(A))$; B: $B, \mathrm{pk}, \mathrm{inv}(\mathrm{pk}(B))$;	
Actions: $A \longrightarrow B : \{\mathrm{N}_A, A\}_{\mathrm{pk}(B)}$ $B \longrightarrow A : \{\mathrm{N}_A, \mathrm{N}_B\}_{\mathrm{pk}(A)}$ $A \longrightarrow B : \{\mathrm{N}_B\}_{\mathrm{pk}(B)}$	Actions: The exchanged messages

A Formal Alice & Bob Language: Syntax

Protocol: NSPK	Protocol: Name of the protocol
Types: Agent: A, B; Number: $\mathrm{N}_A, \mathrm{N}_B$; Function: pk;	Types: Types of all identifiers (may not be considered in the analysis)
Knowledge: A: $A, B, \mathrm{pk}, \mathrm{inv}(\mathrm{pk}(A))$; B: $B, \mathrm{pk}, \mathrm{inv}(\mathrm{pk}(B))$;	Knowledge: Initial knowledge of each role
Actions: $A \longrightarrow B : \{\mathrm{N}_A, A\}_{\mathrm{pk}(B)}$ $B \longrightarrow A : \{\mathrm{N}_A, \mathrm{N}_B\}_{\mathrm{pk}(A)}$ $A \longrightarrow B : \{\mathrm{N}_B\}_{\mathrm{pk}(B)}$	Actions: The exchanged messages

A Formal Alice & Bob Language: Syntax

Protocol: NSPK	Protocol: Name of the protocol
Types: Agent: A, B; Number: $\mathrm{N}_A, \mathrm{N}_B$; Function: pk;	Types: Types of all identifiers (may not be considered in the analysis)
Knowledge: A: $A, B, \mathrm{pk}, \mathrm{inv}(\mathrm{pk}(A))$; B: $B, \mathrm{pk}, \mathrm{inv}(\mathrm{pk}(B))$;	Knowledge: Initial knowledge of each role
Actions: $A \longrightarrow B : \{\mathrm{N}_A, A\}_{\mathrm{pk}(B)}$ $B \longrightarrow A : \{\mathrm{N}_A, \mathrm{N}_B\}_{\mathrm{pk}(A)}$ $A \longrightarrow B : \{\mathrm{N}_B\}_{\mathrm{pk}(B)}$	Actions: The exchanged messages
Goals: $A \cdot\!\!\longrightarrow\!\!\cdot B : \mathrm{N}_A$ $B \cdot\!\!\longrightarrow\!\!\cdot A : \mathrm{N}_B$	

A Formal Alice & Bob Language: Syntax

Protocol: NSPK	Protocol: Name of the protocol
Types: Agent: A, B; Number: $\mathrm{N}_A, \mathrm{N}_B$; Function: pk;	Types: Types of all identifiers (may not be considered in the analysis)
Knowledge: A: $A, B, \mathrm{pk}, \mathrm{inv}(\mathrm{pk}(A))$; B: $B, \mathrm{pk}, \mathrm{inv}(\mathrm{pk}(B))$;	Knowledge: Initial knowledge of each role
Actions: $A \longrightarrow B : \{\mathrm{N}_A, A\}_{\mathrm{pk}(B)}$ $B \longrightarrow A : \{\mathrm{N}_A, \mathrm{N}_B\}_{\mathrm{pk}(A)}$ $A \longrightarrow B : \{\mathrm{N}_B\}_{\mathrm{pk}(B)}$	Actions: The exchanged messages
Goals: $A \cdot\!\!\longrightarrow\!\!\cdot B : \mathrm{N}_A$ $B \cdot\!\!\longrightarrow\!\!\cdot A : \mathrm{N}_B$	Goals: The goals that we want to achieve $A \cdot\!\!\longrightarrow\!\!\cdot B : \mathrm{N}_A$ secure transmission of nonce N_A from A to B $A \cdot\!\!\longrightarrow B : \mathrm{N}_A$ authentic transmission of nonce N_A from A to B $A \longrightarrow\!\!\cdot B : \mathrm{N}_A$ confidential transmission of nonce N_A from A to B

A Formal Alice & Bob Language: Syntax

Protocol: NSPK	Protocol: Name of the protocol
Types: Agent: A, B; Number: $\mathrm{N}_A, \mathrm{N}_B$; Function: pk;	Types: Types of all identifiers (may not be considered in the analysis)
Knowledge: A: $A, B, \mathrm{pk}, \mathrm{inv}(\mathrm{pk}(A))$; B: $B, \mathrm{pk}, \mathrm{inv}(\mathrm{pk}(B))$;	Knowledge: Initial knowledge of each role
Actions: $A \longrightarrow B : \{\mathrm{N}_A, A\}_{\mathrm{pk}(B)}$ $B \longrightarrow A : \{\mathrm{N}_A, \mathrm{N}_B\}_{\mathrm{pk}(A)}$ $A \longrightarrow B : \{\mathrm{N}_B\}_{\mathrm{pk}(B)}$	Actions: The exchanged messages
Goals: $A \cdot\!\!\longrightarrow\!\!\cdot B : \mathrm{N}_A$ $B \cdot\!\!\longrightarrow\!\!\cdot A : \mathrm{N}_B$	Goals: The goals that we want to achieve $A \cdot\!\!\longrightarrow\!\!\cdot B : \mathrm{N}_A$ secure transmission of nonce N_A from A to B $A \cdot\!\!\longrightarrow B : \mathrm{N}_A$ authentic transmission of nonce N_A from A to B $A \longrightarrow\!\!\cdot B : \mathrm{N}_A$ confidential transmission of nonce N_A from A to B

A Formal Alice & Bob Language: Syntax

Protocol: NSPK	Protocol: Name of the protocol
Types: Agent: A, B; Number: $\mathrm{N}_A, \mathrm{N}_B$; Function: pk;	Types: Types of all identifiers (may not be considered in the analysis)
Knowledge: A: $A, B, \mathrm{pk}, \mathrm{inv}(\mathrm{pk}(A))$; B: $B, \mathrm{pk}, \mathrm{inv}(\mathrm{pk}(B))$;	Knowledge: Initial knowledge of each role
Actions: $A \longrightarrow B : \{\mathrm{N}_A, A\}_{\mathrm{pk}(B)}$ $B \longrightarrow A : \{\mathrm{N}_A, \mathrm{N}_B\}_{\mathrm{pk}(A)}$ $A \longrightarrow B : \{\mathrm{N}_B\}_{\mathrm{pk}(B)}$	Actions: The exchanged messages
Goals: $A \cdot\!\!\longrightarrow\!\!\cdot B : \mathrm{N}_A$ $B \cdot\!\!\longrightarrow\!\!\cdot A : \mathrm{N}_B$	Goals: The goals that we want to achieve $A \cdot\!\!\longrightarrow\!\!\cdot B : \mathrm{N}_A$ secure transmission of nonce N_A from A to B $A \cdot\!\!\longrightarrow B : \mathrm{N}_A$ authentic transmission of nonce N_A from A to B $A \longrightarrow\!\!\cdot B : \mathrm{N}_A$ confidential transmission of nonce N_A from A to B

- **Finally, a plaintext notation for tools**

A Formal Alice & Bob Language: Syntax

`Protocol: NSPK`	Protocol: Name of the protocol

`Types:`
`  Agent: A, B;`
`  Number: NA, NB;`
`  Function: pk;`
	Types:
	Types of all identifiers
	(may not be considered in the analysis)

`Knowledge:`
`  A: A, B, pk,inv(pk(A));`
`  B: B,pk,inv(pk(B));`
	Knowledge:
	Initial knowledge of each role

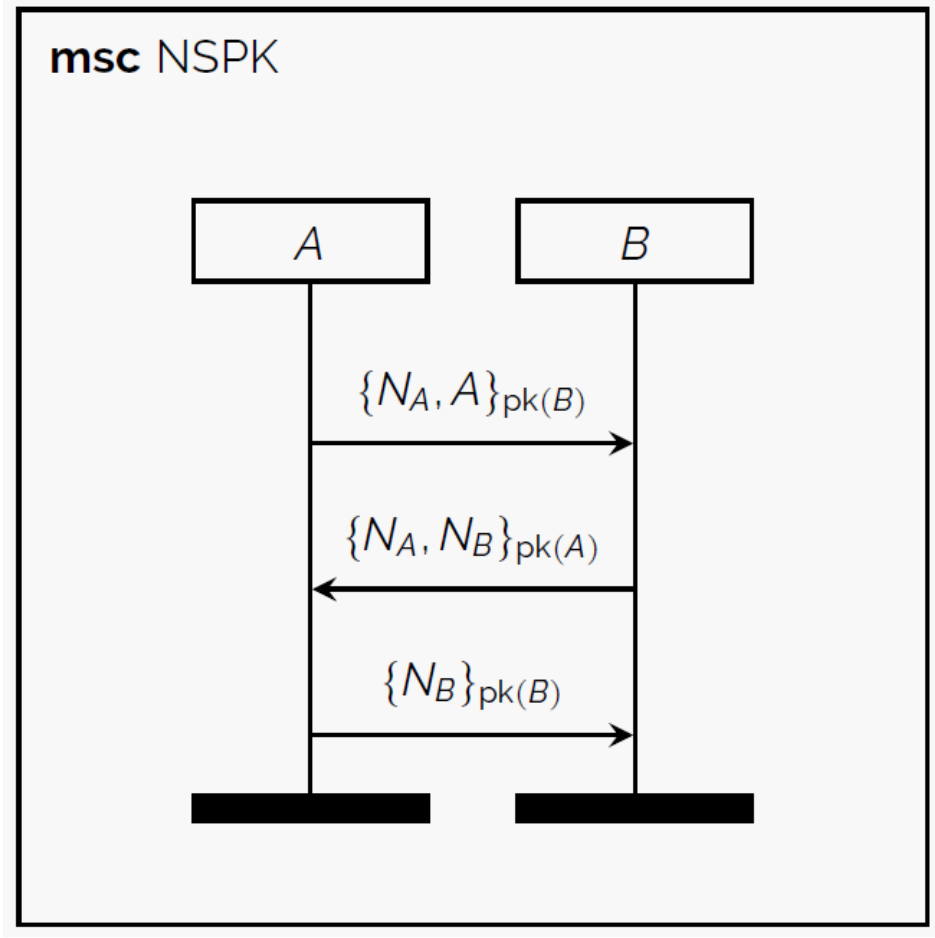
`Actions:`
`  A -> B: {NA, A}(pk(B))`
`  B -> A: {NA, NB}(pk(A))`
`  A -> B: {NB}(pk(B))`
	Actions:
	The exchanged messages

`Goals:`
`  A *->* B: NA`
`  B *->* A: NB`
	Goals:
	The goals that we want to achieve
	`A *->* B: NA` secure transmission of nonce `NA` from `A` to `B`
	`A *-> B: NA` authentic transmission of nonce `NA` from `A` to `B`
	`A ->* B: NA` confidential transmission of nonce `NA` from `A` to `B`

- **Finally, a plaintext notation for tools**

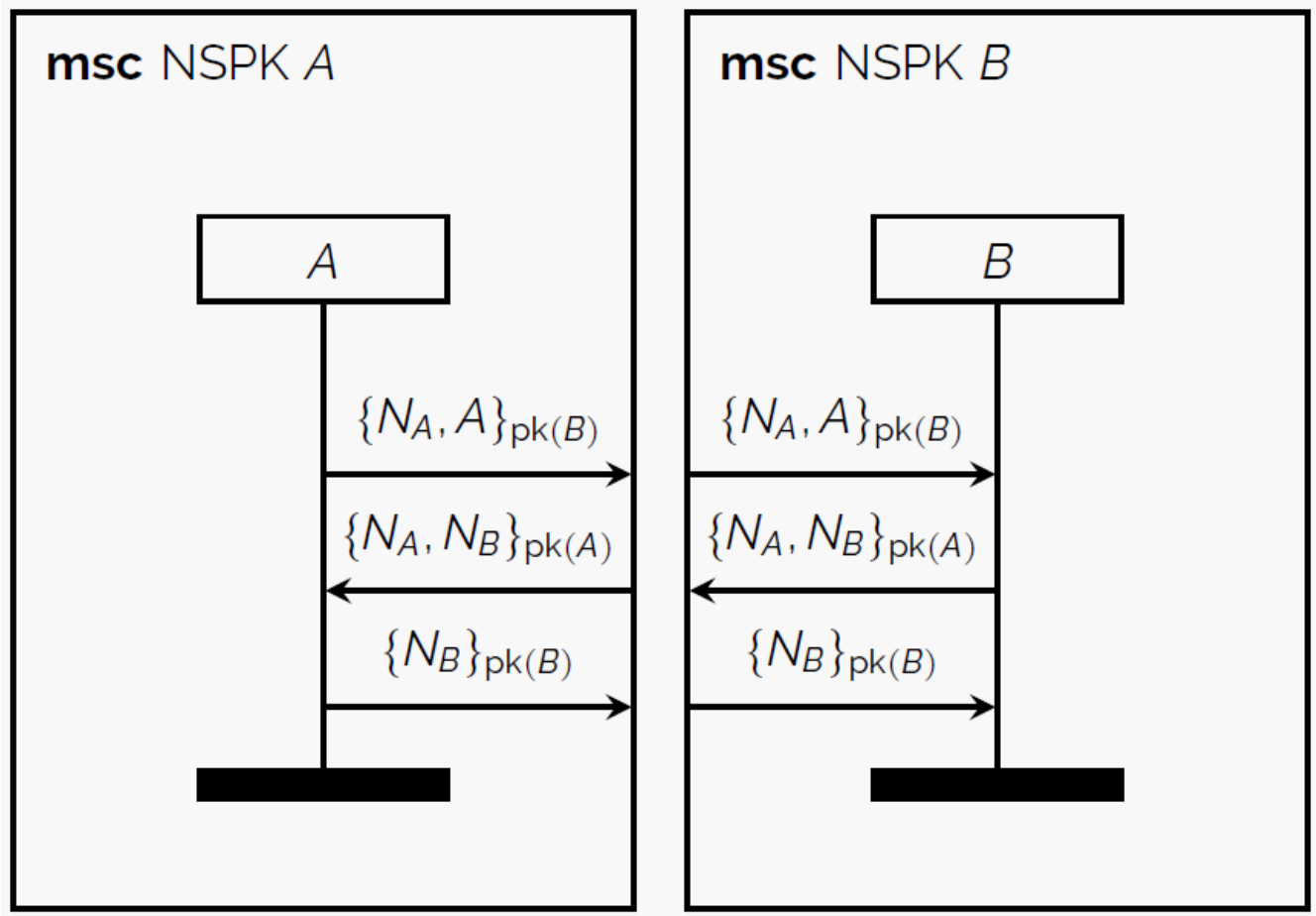
Towards A (Formal) Meaning of AnB Specifications (1/2)

- **Idea:** Split a message sequence chart into single roles
  - a.k.a. chords, symbolic strands, role scripts



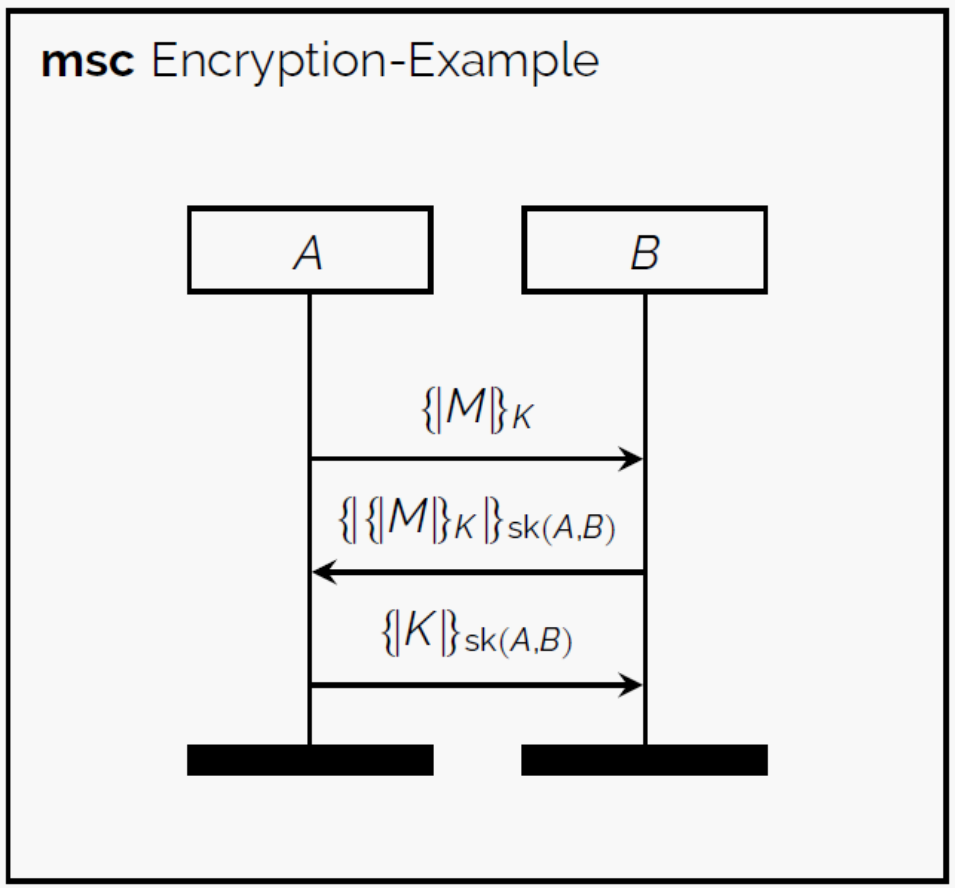
Towards A (Formal) Meaning of AnB Specifications (1/2)

- **Idea:** Split a message sequence chart into single roles
  - a.k.a. chords, symbolic strands, role scripts



Towards A (Formal) Meaning of AnB Specifications (2/2)

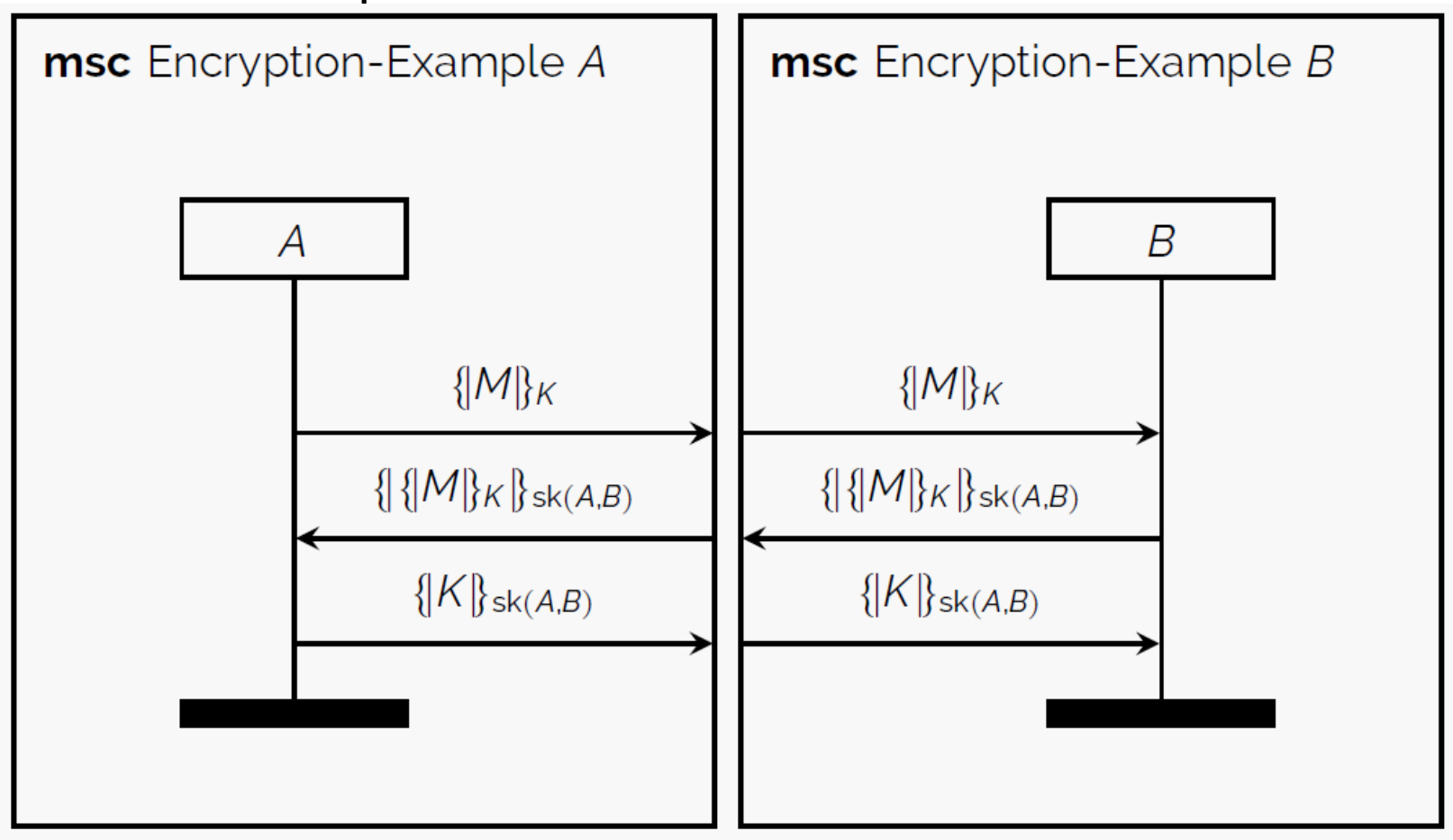
Note: Non-trivial for some protocols



where $sk(A, B)$ is shared key of A and B, and K is fresh

Towards A (Formal) Meaning of AnB Specifications (2/2)

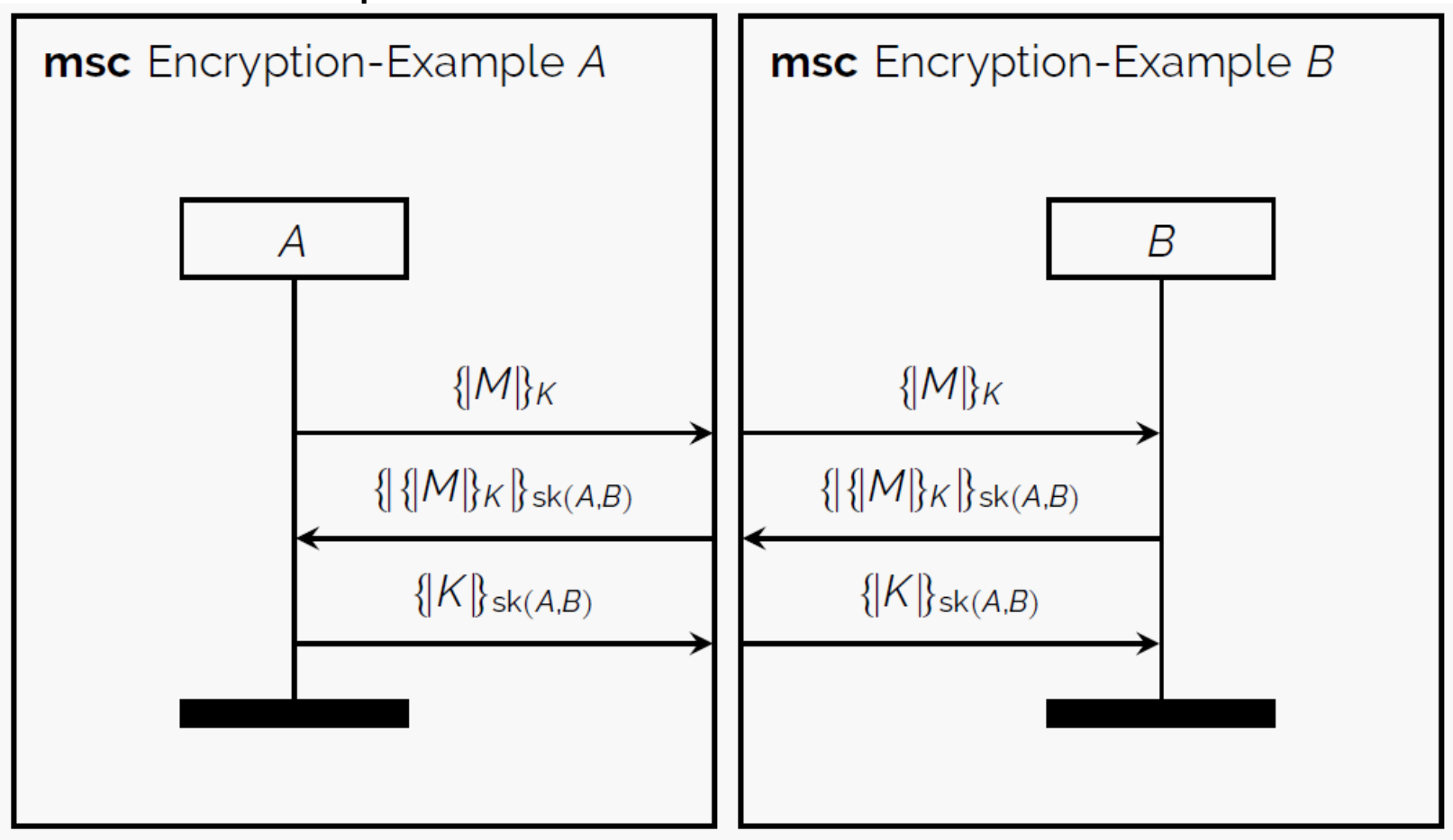
Note: Non-trivial for some protocols



where $sk(A, B)$ is shared key of A and B, and K is fresh

Towards A (Formal) Meaning of AnB Specifications (2/2)

Note: Non-trivial for some protocols



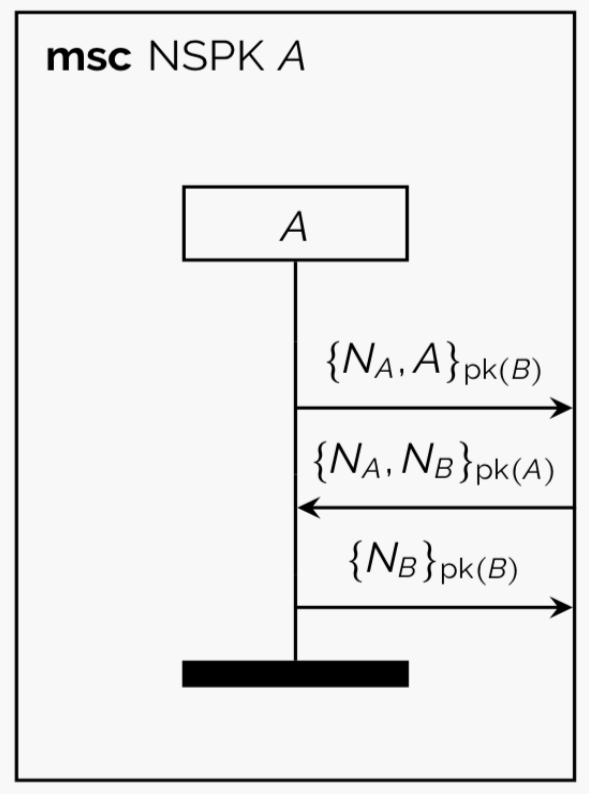
This is wrong!

B cannot check the format of the first message before receiving the third!

where $sk(A,B)$ is shared key of A and B, and K is fresh

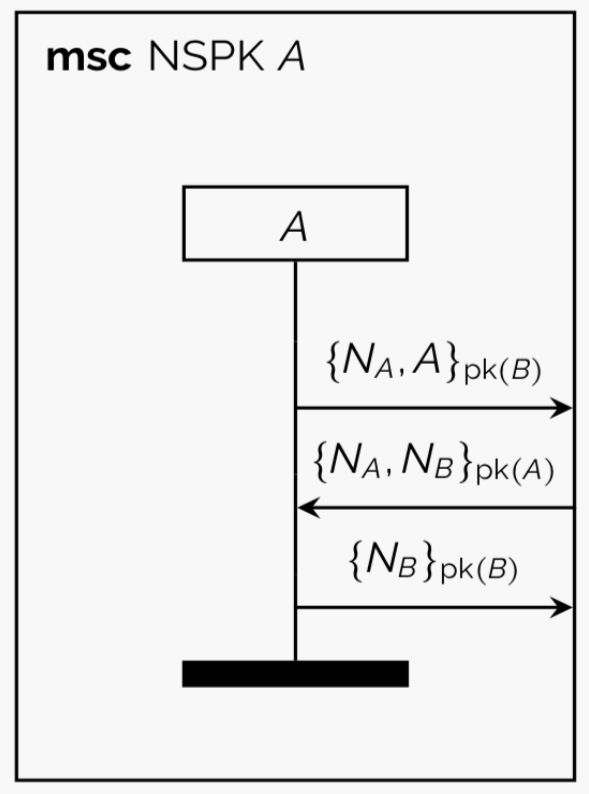
Roles: Syntax

Graphical



Roles: Syntax

Graphical



Textual

$$\mathrm{snd}\left(\{\mathrm{N}_A, A\}_{\mathrm{pk}(B)}\right).\mathrm{rcv}\left(\{\mathrm{N}_A, \mathrm{N}_B\}_{\mathrm{pk}(A)}\right).\mathrm{snd}\left(\{\mathrm{N}_B\}_{\mathrm{pk}(B)}\right)$$

- $Event = \mathrm{snd}\ (Term) \mid \mathrm{rcv}\ (Term) \mid \mathrm{sig}\ (SID, Term)$
- $Roles = Event\ *$
- $Protocols = Rolename \rightharpoonup Role$

Role scripts

- A protocol is described by a role script for each role name
- Role names are variables of type agent
- Signal (sig) events are used for property specification and verification

Roles: Semantics

- Free Variables
  - Variables of a chord are called free for which the first occurrence is not in a receive event

- Role-based Protocol Specifications (a.k.a. chords, symbolic strands, role scripts, etc.)
  - To execute a role, first instantiate all free variables
    - The name of the agent playing the role
    - The name of other agents that are free in the role
    - All the freshly generated values in the role
  - This yields a closed role description, i.e., one without free variables
  - The remaining bound variables are placeholders for parts of messages that are going to be received and for which the value is not yet determined

Roles: Semantics

- Operational Semantics: Definitions

  - $State = Trace \text{ X } IntruderKnowledge \text{ X } Threads$
  - $Trace = (TID \text{ X } Event) *$
  - $IntruderKnowledge = P(Term)$
  - $Threads = TID \rightharpoonup Role$

The TID are Thread IDs for each honest agent playing a role

The Dolev-Yao-Style Intruder

- Defined in

  “On the Security of Public Key Protocols”
  (IEEE Trans. Inf. Th., 1983)
  by Danny Dolev & Andrew C. Yao

On the Security of Public Key Protocols

DANNY DOLEV AND ANDREW C. YAO, MEMBER, IEEE

Abstract—**Recently the use of public key encryption to provide secure network communication has received considerable attention. Such public key systems are usually effective against passive eavesdroppers, who merely tap the lines and try to decipher the message. It has been pointed out, however, that an improperly designed protocol could be vulnerable to an active saboteur, one who may impersonate another user or alter the message being transmitted. Several models are formulated in which the security of protocols can be discussed precisely. Algorithms and characterizations that can be used to determine protocol security in these models are given.**

I. Introduction

THE USE of public key encryption [1], [11] to provide secure network communication has received considerable attention [2], [7], [8], [10]. Such public key systems are usually very effective against a “passive” eavesdropper, namely, one who merely taps the communication line and tries to decipher the intercepted message. However, as pointed out in Needham and Schroeder [8], an improperly designed protocol could be vulnerable to an “active” saboteur, one who may impersonate another user and may alter or replay the message. As a protocol might be compromised in a complex way, informal arguments that assert the security for a protocol are prone to errors. It is thus desirable to have a formal model in which the security issues can be discussed precisely. The models we introduce will enable us to study the security problem for families of protocols, with very few assumptions on the behavior of the saboteur.

We briefly recall the essence of public key encryption (see [1], [11] for more information). In a public key system, every user X has an *encryption function* E_x and a *decryption function* D_x, both are mappings from $\{0, 1\}^*$ (the set of all finite binary sequences) into $\{0, 1\}^*$. A secure public directory contains all the (X, E_x) pairs, while the decryption function D_x is known only to user X. The main requirements on E_x, D_x are:

1) $E_x D_x = D_x E_x = 1$, and
2) knowing $E_x(M)$ and the public directory does not reveal anything about the value M.

Thus everyone can send X a message $E_x(M)$, X will be able to decode it by forming $D_x(E_x(M)) = M$, but nobody other than X will be able to find M even if $E_x(M)$ is available to them.

We will be interested mainly in protocols for transmitting a secret plaintext M between two users. To give an idea of the way a saboteur may break a system, we consider a few examples. A message sent between parties in the network consists of three fields: the sender's name, the receiver's name, and the text. The text is the encrypted part of the message. We will write a message in the format: sender's name, text, receiver's name.

Example 1: Consider the following protocol for sending a plaintext M between A and B:

a) A sends B the message $(A, E_B(M), B)$.
b) B answers A with the message $(B, E_A(M), A)$.

Manuscript received July 15, 1981; revised August 8, 1982. This work was supported in part by ARPA under Grant MDA-903-80-C-102 and by National Science Foundation under Grant MCS-77-05313-A01. This paper was partially presented at the 22nd Annual IEEE Symposium on Foundations of Computer Science, Nashville, TN, October 28–30, 1981.

D. Dolev was with the Computer Science Department, Stanford University, Stanford, CA. He is now with the Institute of Mathematics and Computer Science, Hebrew University, Jerusalem, Israel.

A. C. Yao is with the Computer Science Department, Stanford University, Stanford, CA 94305.

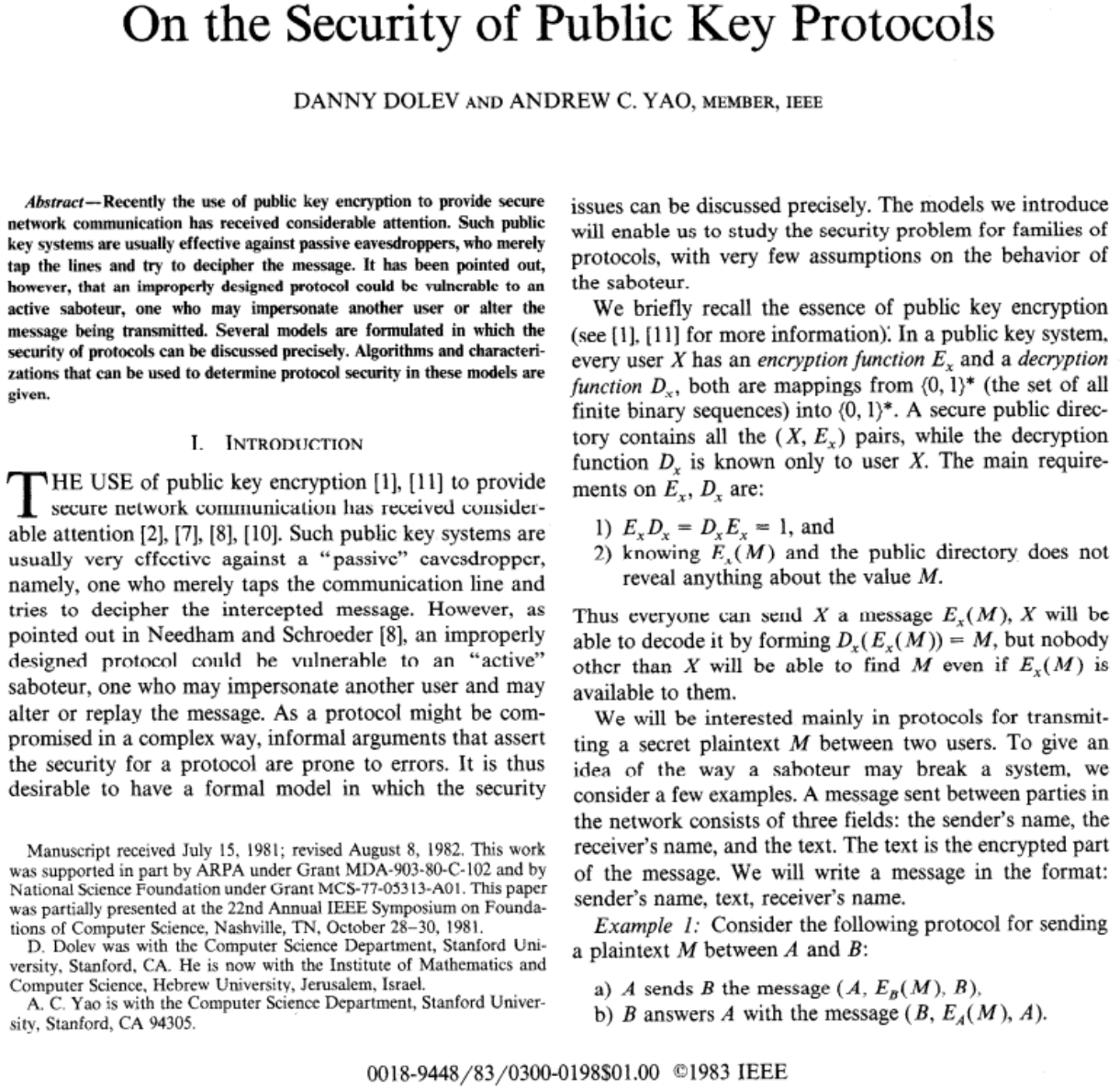
0018-9448/83/0300-0198$01.00

The Dolev-Yao-Style Intruder

- Defined in

  "On the Security of Public Key Protocols"
  (IEEE Trans. Inf. Th., 1983)
  by Danny Dolev & Andrew C. Yao

- Consider a public key system in which for every user X
  - There is a public encryption function E_X
    - Every user can apply this function
  - And a private decryption function D_X
    - Only X can apply this function



On the Security of Public Key Protocols

DANNY DOLEV AND ANDREW C. YAO, MEMBER, IEEE

Abstract—Recently the use of public key encryption to provide secure network communication has received considerable attention. Such public key systems are usually effective against passive eavesdroppers, who merely tap the lines and try to decipher the message. It has been pointed out, however, that an improperly designed protocol could be vulnerable to an active saboteur, one who may impersonate another user or alter the message being transmitted. Several models are formulated in which the security of protocols can be discussed precisely. Algorithms and characterizations that can be used to determine protocol security in these models are given.

I. Introduction

THE USE of public key encryption [1], [11] to provide secure network communication has received considerable attention [2], [7], [8], [10]. Such public key systems are usually very effective against a "passive" eavesdropper, namely, one who merely taps the communication line and tries to decipher the intercepted message. However, as pointed out in Needham and Schroeder [8], an improperly designed protocol could be vulnerable to an "active" saboteur, one who may impersonate another user and may alter or replay the message. As a protocol might be compromised in a complex way, informal arguments that assert the security for a protocol are prone to errors. It is thus desirable to have a formal model in which the security issues can be discussed precisely. The models we introduce will enable us to study the security problem for families of protocols, with very few assumptions on the behavior of the saboteur.

We briefly recall the essence of public key encryption (see [1], [11] for more information). In a public key system, every user X has an *encryption function* E_x and a *decryption function* D_x, both are mappings from $\{0, 1\}^*$ (the set of all finite binary sequences) into $\{0, 1\}^*$. A secure public directory contains all the (X, E_x) pairs, while the decryption function D_x is known only to user X. The main requirements on E_x, D_x are:

1) $E_x D_x = D_x E_x = 1$, and
2) knowing $E_x(M)$ and the public directory does not reveal anything about the value M.

Thus everyone can send X a message $E_x(M)$, X will be able to decode it by forming $D_x(E_x(M)) = M$, but nobody other than X will be able to find M even if $E_x(M)$ is available to them.

We will be interested mainly in protocols for transmitting a secret plaintext M between two users. To give an idea of the way a saboteur may break a system, we consider a few examples. A message sent between parties in the network consists of three fields: the sender's name, the receiver's name, and the text. The text is the encrypted part of the message. We will write a message in the format: sender's name, text, receiver's name.

Example 1: Consider the following protocol for sending a plaintext M between A and B:

a) A sends B the message $(A, E_B(M), B)$.
b) B answers A with the message $(B, E_A(M), A)$.

Manuscript received July 15, 1981; revised August 8, 1982. This work was supported in part by ARPA under Grant MDA-903-80-C-102 and by National Science Foundation under Grant MCS-77-05313-A01. This paper was partially presented at the 22nd Annual IEEE Symposium on Foundations of Computer Science, Nashville, TN, October 28–30, 1981.

D. Dolev was with the Computer Science Department, Stanford University, Stanford, CA. He is now with the Institute of Mathematics and Computer Science, Hebrew University, Jerusalem, Israel.

A. C. Yao is with the Computer Science Department, Stanford University, Stanford, CA 94305.

0018-9448/83/0300-0198$01.00

The Dolev-Yao-Style Intruder

- Defined in

  "On the Security of Public Key Protocols"
  (IEEE Trans. Inf. Th., 1983)
  by Danny Dolev & Andrew C. Yao

- Consider a public key system in which for every user X
  - There is a public encryption function E_X
    - Every user can apply this function
  - And a private decryption function D_X
    - Only X can apply this function
  - These functions have the property that

$$E_X\ D_X = D_X\ E_X = 1$$

On the Security of Public Key Protocols

DANNY DOLEV AND ANDREW C. YAO, MEMBER, IEEE

Abstract—**Recently the use of public key encryption to provide secure network communication has received considerable attention. Such public key systems are usually effective against passive eavesdroppers, who merely tap the lines and try to decipher the message. It has been pointed out, however, that an improperly designed protocol could be vulnerable to an active saboteur, one who may impersonate another user or alter the message being transmitted. Several models are formulated in which the security of protocols can be discussed precisely. Algorithms and characterizations that can be used to determine protocol security in these models are given.**

I. Introduction

THE USE of public key encryption [1], [11] to provide secure network communication has received considerable attention [2], [7], [8], [10]. Such public key systems are usually very effective against a "passive" eavesdropper, namely, one who merely taps the communication line and tries to decipher the intercepted message. However, as pointed out in Needham and Schroeder [8], an improperly designed protocol could be vulnerable to an "active" saboteur, one who may impersonate another user and may alter or replay the message. As a protocol might be compromised in a complex way, informal arguments that assert the security for a protocol are prone to errors. It is thus desirable to have a formal model in which the security issues can be discussed precisely. The models we introduce will enable us to study the security problem for families of protocols, with very few assumptions on the behavior of the saboteur.

We briefly recall the essence of public key encryption (see [1], [11] for more information). In a public key system, every user X has an *encryption function* E_x and a *decryption function* D_x, both are mappings from $\{0, 1\}^*$ (the set of all finite binary sequences) into $\{0, 1\}^*$. A secure public directory contains all the (X, E_x) pairs, while the decryption function D_x is known only to user X. The main requirements on E_x, D_x are:

1) $E_x D_x = D_x E_x = 1$, and
2) knowing $E_x(M)$ and the public directory does not reveal anything about the value M.

Thus everyone can send X a message $E_x(M)$, X will be able to decode it by forming $D_x(E_x(M)) = M$, but nobody other than X will be able to find M even if $E_x(M)$ is available to them.

We will be interested mainly in protocols for transmitting a secret plaintext M between two users. To give an idea of the way a saboteur may break a system, we consider a few examples. A message sent between parties in the network consists of three fields: the sender's name, the receiver's name, and the text. The text is the encrypted part of the message. We will write a message in the format: sender's name, text, receiver's name.

Example 1: Consider the following protocol for sending a plaintext M between A and B:

a) A sends B the message $(A, E_B(M), B)$.
b) B answers A with the message $(B, E_A(M), A)$.

Manuscript received July 15, 1981; revised August 8, 1982. This work was supported in part by ARPA under Grant MDA-903-80-C-102 and by National Science Foundation under Grant MCS-77-05313-A01. This paper was partially presented at the 22nd Annual IEEE Symposium on Foundations of Computer Science, Nashville, TN, October 28–30, 1981.

D. Dolev was with the Computer Science Department, Stanford University, Stanford, CA. He is now with the Institute of Mathematics and Computer Science, Hebrew University, Jerusalem, Israel.

A. C. Yao is with the Computer Science Department, Stanford University, Stanford, CA 94305.

0018-9448/83/0300-0198$01.00

The Dolev-Yao-Style Intruder

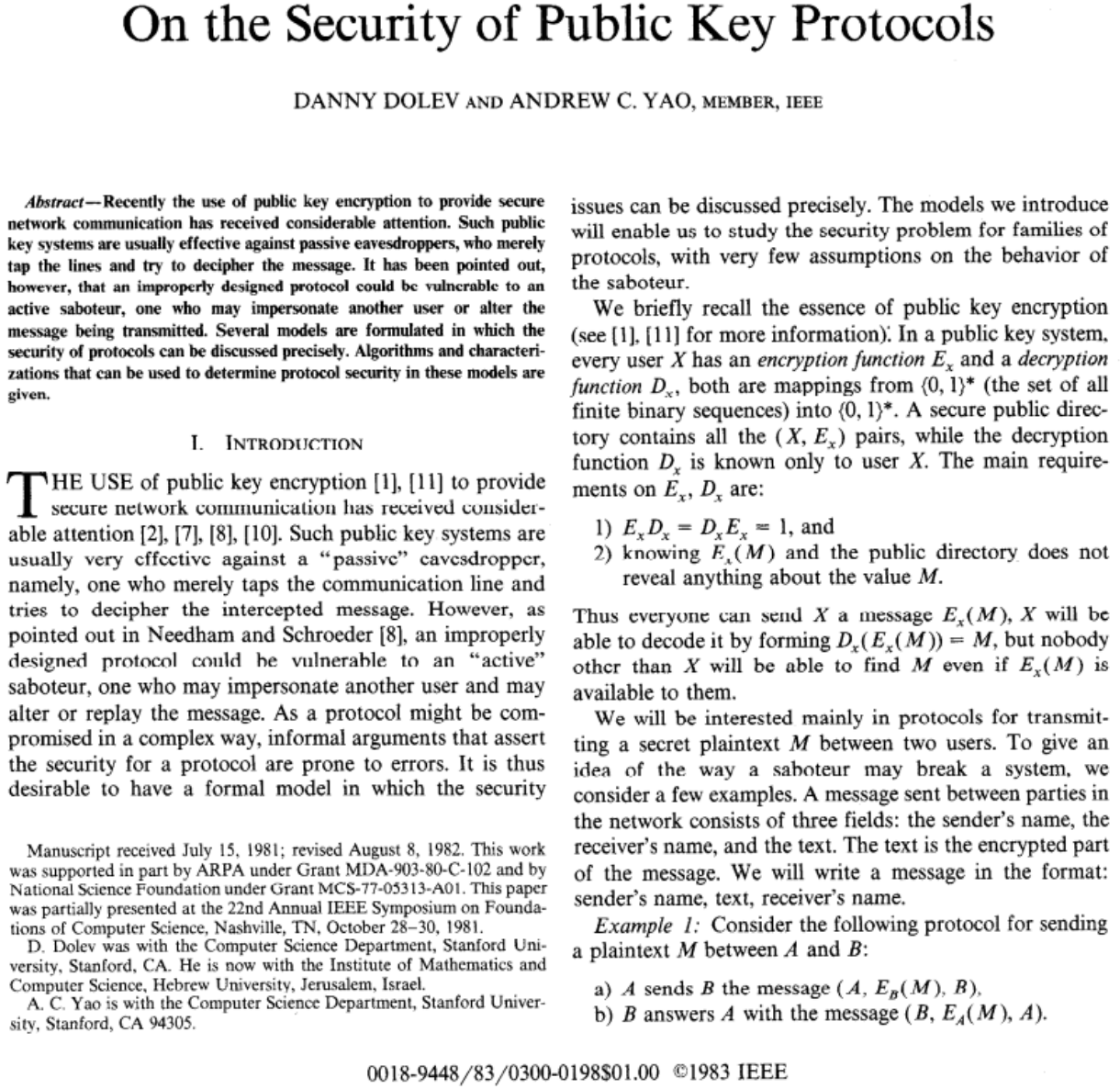
- Defined in

  "On the Security of Public Key Protocols"
  (IEEE Trans. Inf. Th., 1983)
  by Danny Dolev & Andrew C. Yao

- Consider a public key system in which for every user X
  - There is a public encryption function E_X
    - Every user can apply this function
  - And a private decryption function D_X
    - Only X can apply this function
  - These functions have the property that

$$E_X\ D_X = D_X\ E_X = 1$$

- The **Dolev-Yao Intruder**
  - Controls the network (read, intercept, send)
  - Is also a user, called Z
  - Can apply E_X for any X
  - Can apply D_Z



On the Security of Public Key Protocols

DANNY DOLEV AND ANDREW C. YAO, MEMBER, IEEE

Abstract—Recently the use of public key encryption to provide secure network communication has received considerable attention. Such public key systems are usually effective against passive eavesdroppers, who merely tap the lines and try to decipher the message. It has been pointed out, however, that an improperly designed protocol could be vulnerable to an active saboteur, one who may impersonate another user or alter the message being transmitted. Several models are formulated in which the security of protocols can be discussed precisely. Algorithms and characterizations that can be used to determine protocol security in these models are given.

I. Introduction

THE USE of public key encryption [1], [11] to provide secure network communication has received considerable attention [2], [7], [8], [10]. Such public key systems are usually very effective against a "passive" eavesdropper, namely, one who merely taps the communication line and tries to decipher the intercepted message. However, as pointed out in Needham and Schroeder [8], an improperly designed protocol could be vulnerable to an "active" saboteur, one who may impersonate another user and may alter or replay the message. As a protocol might be compromised in a complex way, informal arguments that assert the security for a protocol are prone to errors. It is thus desirable to have a formal model in which the security issues can be discussed precisely. The models we introduce will enable us to study the security problem for families of protocols, with very few assumptions on the behavior of the saboteur.

We briefly recall the essence of public key encryption (see [1], [11] for more information). In a public key system, every user X has an *encryption function* E_x and a *decryption function* D_x, both are mappings from $\{0, 1\}^*$ (the set of all finite binary sequences) into $\{0, 1\}^*$. A secure public directory contains all the (X, E_x) pairs, while the decryption function D_x is known only to user X. The main requirements on E_x, D_x are:

1) $E_x D_x = D_x E_x = 1$, and
2) knowing $E_x(M)$ and the public directory does not reveal anything about the value M.

Thus everyone can send X a message $E_x(M)$, X will be able to decode it by forming $D_x(E_x(M)) = M$, but nobody other than X will be able to find M even if $E_x(M)$ is available to them.

We will be interested mainly in protocols for transmitting a secret plaintext M between two users. To give an idea of the way a saboteur may break a system, we consider a few examples. A message sent between parties in the network consists of three fields: the sender's name, the receiver's name, and the text. The text is the encrypted part of the message. We will write a message in the format: sender's name, text, receiver's name.

Example 1: Consider the following protocol for sending a plaintext M between A and B:

a) A sends B the message $(A, E_B(M), B)$.
b) B answers A with the message $(B, E_A(M), A)$.

Manuscript received July 15, 1981; revised August 8, 1982. This work was supported in part by ARPA under Grant MDA-903-80-C-102 and by National Science Foundation under Grant MCS-77-05313-A01. This paper was partially presented at the 22nd Annual IEEE Symposium on Foundations of Computer Science, Nashville, TN, October 28–30, 1981.

D. Dolev was with the Computer Science Department, Stanford University, Stanford, CA. He is now with the Institute of Mathematics and Computer Science, Hebrew University, Jerusalem, Israel.

A. C. Yao is with the Computer Science Department, Stanford University, Stanford, CA 94305.

0018-9448/83/0300-0198$01.00

The Dolev-Yao-Style Intruder

- Defined in

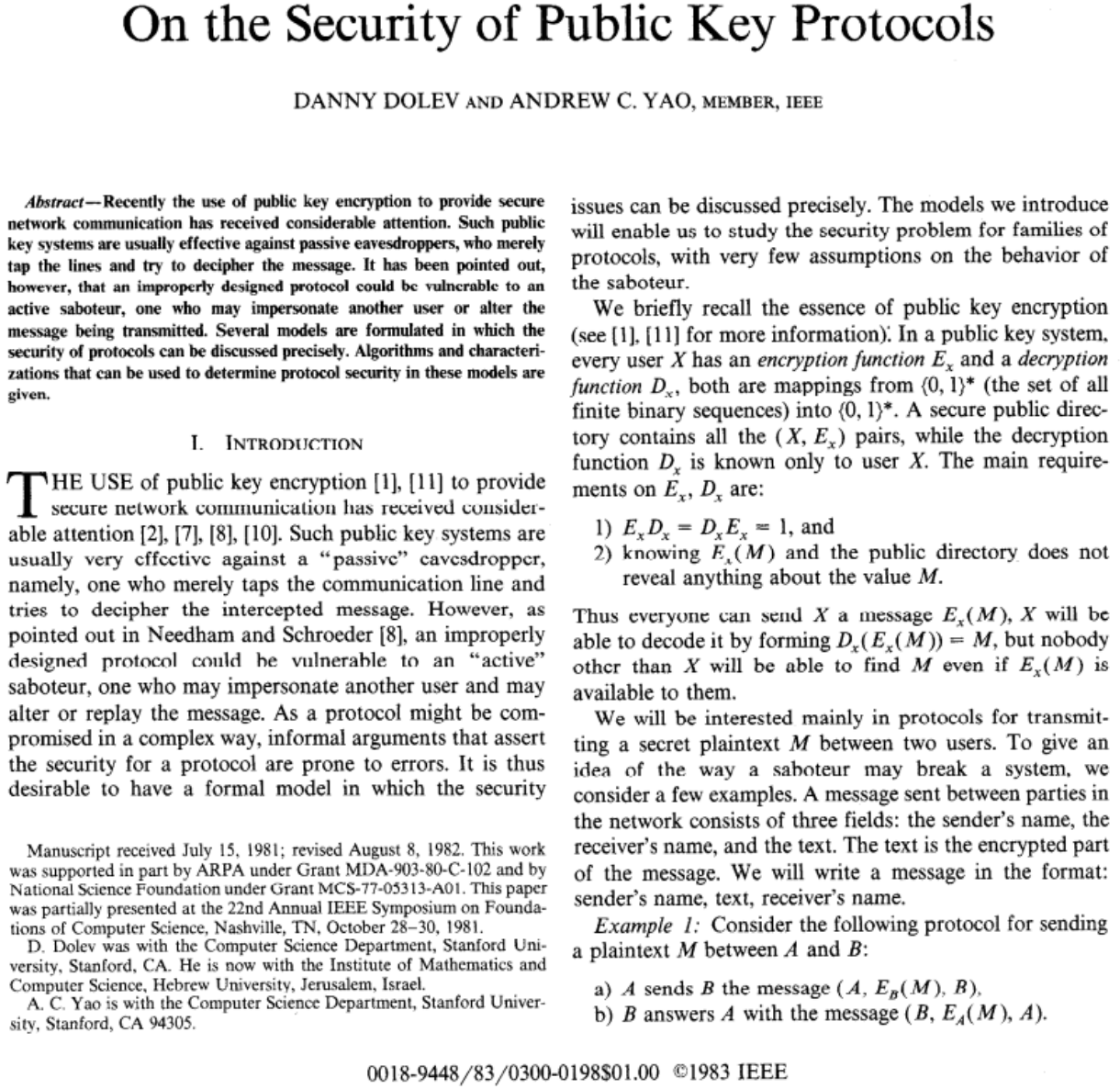
  "On the Security of Public Key Protocols"
  (IEEE Trans. Inf. Th., 1983)
  by Danny Dolev & Andrew C. Yao

- Consider a public key system in which for every user X
  - There is a public encryption function E_X
    - Every user can apply this function
  - And a private decryption function D_X
    - Only X can apply this function
  - These functions have the property that

$$E_X\ D_X = D_X\ E_X = 1$$

- The **Dolev-Yao Intruder**
  - Controls the network (read, intercept, send)
  - Is also a user, called Z
  - Can apply E_X for any X
  - Can apply D_Z

See the "Attacker" in your Problem Sheet 9, Exercise 2!



On the Security of Public Key Protocols

DANNY DOLEV AND ANDREW C. YAO, MEMBER, IEEE

Abstract—Recently the use of public key encryption to provide secure network communication has received considerable attention. Such public key systems are usually effective against passive eavesdroppers, who merely tap the lines and try to decipher the message. It has been pointed out, however, that an improperly designed protocol could be vulnerable to an active saboteur, one who may impersonate another user or alter the message being transmitted. Several models are formulated in which the security of protocols can be discussed precisely. Algorithms and characterizations that can be used to determine protocol security in these models are given.

I. INTRODUCTION

THE USE of public key encryption [1], [11] to provide secure network communication has received considerable attention [2], [7], [8], [10]. Such public key systems are usually very effective against a "passive" eavesdropper, namely, one who merely taps the communication line and tries to decipher the intercepted message. However, as pointed out in Needham and Schroeder [8], an improperly designed protocol could be vulnerable to an "active" saboteur, one who may impersonate another user and may alter or replay the message. As a protocol might be compromised in a complex way, informal arguments that assert the security for a protocol are prone to errors. It is thus desirable to have a formal model in which the security issues can be discussed precisely. The models we introduce will enable us to study the security problem for families of protocols, with very few assumptions on the behavior of the saboteur.

We briefly recall the essence of public key encryption (see [1], [11] for more information). In a public key system, every user X has an *encryption function* E_x and a *decryption function* D_x, both are mappings from $\{0, 1\}^*$ (the set of all finite binary sequences) into $\{0, 1\}^*$. A secure public directory contains all the (X, E_x) pairs, while the decryption function D_x is known only to user X. The main requirements on E_x, D_x are:

1) $E_x D_x = D_x E_x = 1$, and
2) knowing $E_x(M)$ and the public directory does not reveal anything about the value M.

Thus everyone can send X a message $E_x(M)$, X will be able to decode it by forming $D_x(E_x(M)) = M$, but nobody other than X will be able to find M even if $E_x(M)$ is available to them.

We will be interested mainly in protocols for transmitting a secret plaintext M between two users. To give an idea of the way a saboteur may break a system, we consider a few examples. A message sent between parties in the network consists of three fields: the sender's name, the receiver's name, and the text. The text is the encrypted part of the message. We will write a message in the format: sender's name, text, receiver's name.

Example 1: Consider the following protocol for sending a plaintext M between A and B:

a) A sends B the message $(A, E_B(M), B)$.
b) B answers A with the message $(B, E_A(M), A)$.

Manuscript received July 15, 1981; revised August 8, 1982. This work was supported in part by ARPA under Grant MDA-903-80-C-102 and by National Science Foundation under Grant MCS-77-05313-A01. This paper was partially presented at the 22nd Annual IEEE Symposium on Foundations of Computer Science, Nashville, TN, October 28–30, 1981.

D. Dolev was with the Computer Science Department, Stanford University, Stanford, CA. He is now with the Institute of Mathematics and Computer Science, Hebrew University, Jerusalem, Israel.

A. C. Yao is with the Computer Science Department, Stanford University, Stanford, CA 94305.

0018-9448/83/0300-0198$01.00

The Dolev-Yao-Style Intruder

- Follow the definitions, models, examples from the Dolev & Yao paper

- Follow notation from “Logic in Computer Science: Modelling and Reasoning About Systems” by M. Huth and M. D. Ryan
  - Terms and algebraic properties

- For this course
  - You should be able to work with
    - The (graphical and textual) AnB notation
    - The “Dolev-Yao Intruder” definition
  - You should be able to understand the proofs of simple properties
  - You should be able to focus on practical problems similar to the ones discussed in exercise sessions

Summary

- Security protocols are difficult to design
  - Lack of formal approaches lead to security problems
- Formal methods help to precisely define
  - Protocols
  - Security goals
- In turn
  - Analysis of protocol security is formalized
  - Proving the security of the protocol “easier” with a formal approach

Reading List

- D. Dolev and A. C. Yao. On the Security of Public Key Protocols. Symposium on Foundations of Computer Science, 0:350–357, 1981.
  - The paper is available at: https://www.cs.huji.ac.il/~dolev/pubs/dolev-yao-ieee-01056650.pdf
- M. Huth and M. D. Ryan. Logic in Computer Science: Modelling and Reasoning About Systems. Cambridge University Press, 2004.

Thanks for your attention!

- Any questions or remarks?