Network Security

Cryptographic Protocols

Dr. Heiko Niedermayer
Cornelius Diekmann

Lehrstuhl für Netzarchitekturen und Netzdienste
Institut für Informatik
Technische Universität München

Version: January 7, 2016

Acknowledgements

- This course is based to some extend on slides provided by Günter Schäfer, author of the book ”Netzsicherheit - Algorithmische Grundlagen und Protokolle”, available in German from dpunkt Verlag.
- The English version of the book is entitled “Security in Fixed and Wireless Networks: An Introduction to Securing Data Communications” and is published by Wiley is also available. We gratefully acknowledge his support.
- The slide set has been reworked by Heiko Niedermayer, Ali Fessi, Ralph Holz, Cornelius Diekmann, and Georg Carle.

 Explanation Pony and Exercises

- Slides called ”- Explanation” and usually marked with  are not for the lecture, but they contain further explanations for your learning at home.
- Parts called ”Exercise” are voluntary exercises for discussion in lecture as well as for your reworking of the slides and learning at home.

Agenda

1 Introduction

2 Learning Goals

3 Protocols

4 Authentication and Key Establishment Protocols

5 Attack Concepts against Cryptographic Protocols

6 Desirable Properties of Cryptographic Protocols

7 Final Protocol, Goals, and Notation

8 Example: Needham Schroeder Protocol
- Needham Schroeder Symmetric Key Protocol
- Needham Schroeder Public Key Protocol

9 Conclusions - What have we learned

Introduction

Introduction

- Communicate over distance using a network
- Do I speak with the right person?
- Who can read the content?
- How can cryptography be used for that?
- Which keys? From where and when?
- Protocols describe an exchange of messages for a certain purpose (e.g. security goals).

  $\rightarrow$ Cryptographic Protocols

Learning Goals

Learning Goals

- Basic understanding of cryptographic protocols
  - Know the terms and methods and apply them
- Get to know some elementary protocols $\rightarrow$ real-world protocols discussed in later chapters use the basics you learn here
  - Remember and explain them
- You will gain some first thoughts about how to break protocols and learn to think in a way of finding attacks
  - Apply them to find weaknesses
- You will gain some first ideas how to improve protocols
  - Apply them to remove a similar weakness

Stick to the protocol layer (Exercise, Exam)

- When we discuss cryptographic protocols, we assume the following
  - The cryptographic primitives are secure.
    $\rightarrow$ Insecure primitives or implementations can make a secure protocol insecure[1].
  - The computers, machines, ... are secure.
    $\rightarrow$ Insecure machines can make the use of a secure protocol insecure.
  - Our reasoning uses the Dolev-Yao attacker model.
    $\rightarrow$ Attacker = Network
  - Our reasoning focuses on the layer of the cryptographic protocol.
    $\rightarrow$ Security can be shown by formal methods (model checking of protocol, security proofs, etc.).

[1] Defended by security proofs, code review, formal code analysis, . . .

Stick to the protocol layer (Exercise, Exam)

- When we try to break a protocol, we do this on the layers of the protocol.
  - The reason is that we do not learn anything about weaknesses and security of protocols if we attack them by assuming to hack a computer and steal all data. So, whenever you are asked to analyze or attack a protocol in an exercise, attack on the layer of the protocol and attack its operation. Otherwise you do not learn to understand and evaluate protocols.
- The same is true if we consider mitigations. Fix the protocol by changing its operation, not by adding new requirements like super-secure primitives or machines.

Protocols

What do we know as of yet?

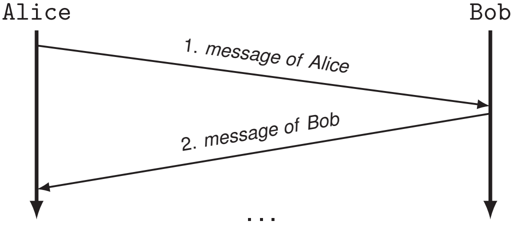
- What do we know?
  - Symmetric encryption and keys
  - Asymmetric encryption and keys
  - Cryptographic hash functions
  - Secure Channel
- To use a secure channel, Alice and Bob need a shared key.
  $\rightarrow$ A protocol to establish a secure channel needs to establish a shared key.

Protocols, Notation, . . .

- Cryptographic protocols contain:
  - General entities that are normal participants of the protocol. We call them Alice (A), Bob (B), . . .
  - Special-purpose entities that have a special role. Authentication Server (AS), . . .
  - Some synonyms: entity, principle, participant
- Alice-Bob notation: one way to describe cryptographic protocols
  - Protocol messages in sequence (numbering optional):
  - 1. $Alice \rightarrow Bob : message\ of\ Alice$
  - 2. $Bob \rightarrow Alice : message\ of\ Bob$
  - . . .

Protocols, Notation, ... 2

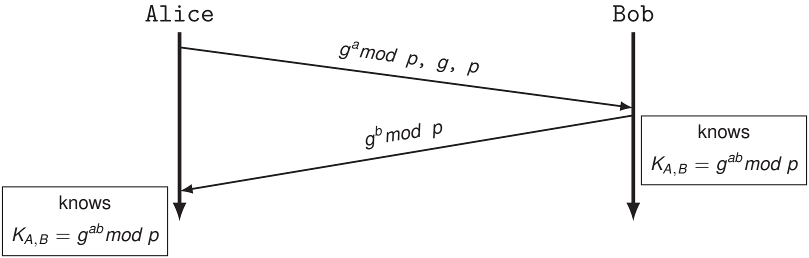
- Or sequence diagram:



- Some Notation:

Notation	Meaning
$A, B, \ldots$	Protocol principles
$K_{A,B}$	Key, here shared key of A and B
$\{m\}_K$	Plaintext *m* encrypted and integrity-protected with key *K*

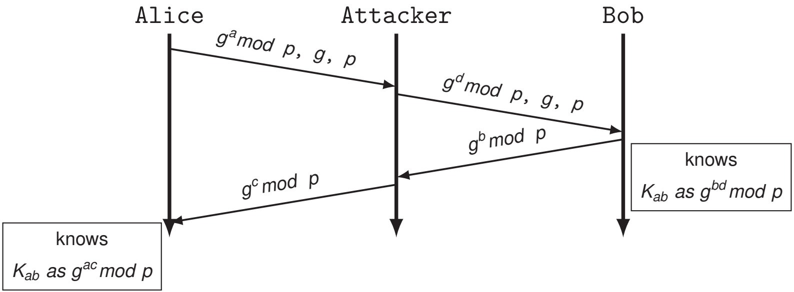
Protocol Try 1 (Textbook Diffie-Hellman)



- Alice and Bob have completed a Diffie-Hellman exhange and established a shared key at the end of the protocol.
- Are we done?

Protocol Try 1 - What goes wrong (Man-in-the-Middle)

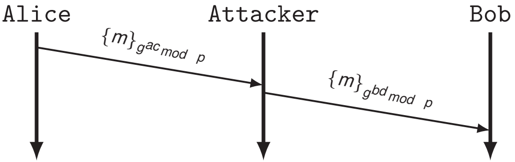
Repetition. This should already be known.



- Attacker now has a shared key K_{ac} with Alice and a shared key K_{bd} with Bob.
- The attacker is called Man-in-the-Middle attacker as it sits in-between any communication between Alice and Bob.

Protocol Try 1 - What goes wrong (Man-in-the-Middle) 2

When Alice uses the secure channel to send message m:

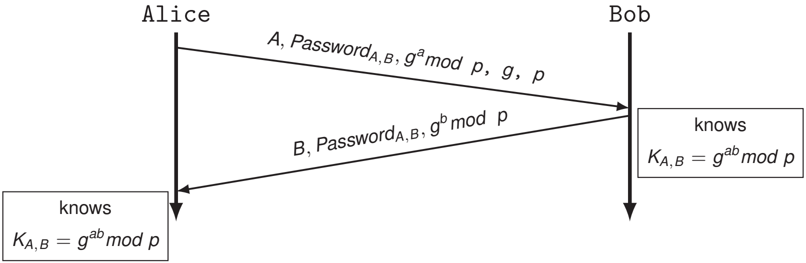


- Despite using the secure channel, the attacker can read, modify, or create message between Alice and Bob.

Protocol Try 1 - What goes wrong (No Authentication)

- The exchange does not contain any authentication.
- Thus, Alice has no way of identifying Bob.
- Bob has no way of identifying Alice.
- An attacker can impersonate whomever it likes.

Protocol Try 2 - Adding a password



- Try 2 still fails:
  - Man-in-the-middle still possible
  - Eavesdropper can read password, then impersonation possible
- Why do they already have a password? Lets discuss authentication.

Authentication and Key Establishment Protocols

Entity Authentication and Key Establishment

- Entity Authentication
  - Authenticity of an entity is shown
  - An authentication protocol is run and at the end, some protocol participants are ensured of the identity of other participants.
  - Mutual authentication: Authenticity of Alice and Bob is shown to each other
- Key Establishment
  - A key is established between some protocol participants
  - Key Transport: Some entity creates the key and sends it to other entities.
  - Key Agreement: Multiple entities contribute to the generation of the key.

Entity Authentication and Key Establishment

- Many authentication protocols – as a side effect of the authentication - do establish a shared session key $K_{A,B}$ for securing the session.
- Some opinions about the relationship between authentication and key establishment:
  - ”It is accepted that these topics should be considered jointly rather separately” [Diff92]
  - ”... authentication is rarely useful in the absence of an associated key distribution” [Bell95]
  - ”In our view there are situations when entity authentication by itself may useful, such as when using a physically secured communication channel.” [Boyd03]

Key Establishment without Entity Authentication

- Why our first try failed... After a protocol run, neither Alice nor Bob know with whom they actually have exchanged a key.
- Can Key Establishment without Authentication work?
  - If Alice and Bob already have an authenticated channel, then a key exchange over that channel may not need to authenticate.

Entity Authentication without Key Establishment

- Entity Authentication without Key Establishment?
  - In cyber-physical system: something happens in physical world upon authentication.
    - E.g. door opens for Alice. No session key needed.
  - Over the network?
    - If a shared key already exists, only the binding of key and identity (authentication) may be needed.

Entity Authentication and Key Establishment in WWW

- Alice wants to use the online banking service provided by her bank
- Authentication of the web server of the bank:
  - Web browser verifies the identity of the web server via HTTPS using asymmetric encryption
  - A shared session key $K_{A,B}$ is generated as part of the server authentication
  - A secure channel between web browser and web server is established
- Authentication of the client:
  - Uses the secure channel to the web server
  - The web server authenticates Alice based on her PIN number
  - No additional secret key is established

Operation of Cryptographic Protocols

- Initiator
  - The principle (entity) that starts the protocol by sending the first message.
- Responder
  - Principles that did not start the protocol.
- All principles
  - see messages
  - send messages
  - draw conclusions from observations

Authentication = Proof in Formal Logic

- Each principle has its knowledge and beliefs.
- In the operation of the cryptographic protocol it takes certain actions.
- In the operation of the cryptographic protocol it makes observations.
- Reasoning on actions and observation needs to establish the objectives of the protocol.
  - Example (Authenticity of Bob):
    - Sent fresh challenge to Bob.
    - Protected it with public key of Bob.
    - If anyone can read the challenge, then it has to have knowledge of Bob's private key.
    - Value from the challenge is seen again.
    - Thus, Bob participated in the protocol and used his private key.

Where do the keys come from?

- Alice and Bob can have a long-term shared key.
- Alice and Bob can have exchanged their public keys.
- Alice and Bob have exchanged keys with a Trusted Third Party (TTP). The TTP helps.
  - More scalable.
  - Typical names for the TTP: Authentication Server (AS), Certification Authority (CA), ...
- If no such pre-exchanged keys exist, cryptographic protocols cannot operate securely (Boyd's Theorem).
- More on the issue in a separate chapter on Identity and Public Key Infrastructures.

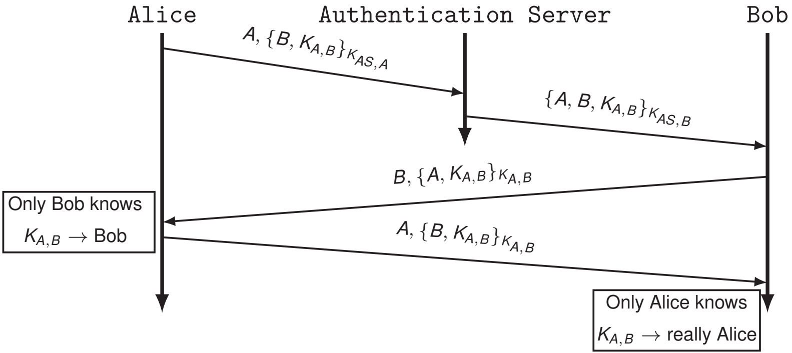
Authentication and Key Establishment Problem Statement - Version 1

Goals: Run a key exchange protocol such that at the end of the protocol:

- Alice and Bob have shared session key for a secure channel
- Alice (Bob) must be able to verify that Bob (Alice) participated in the protocol run (authentication)

Protocol Try 3 Shared Key with Server

- Using a TTP is more scalable, so lets use a server.
- Alice generates a fresh key and sends it to Bob via the server.
- Btw, when we encrypt, the receiver might need to know who sends the message, at least if it is not the server.



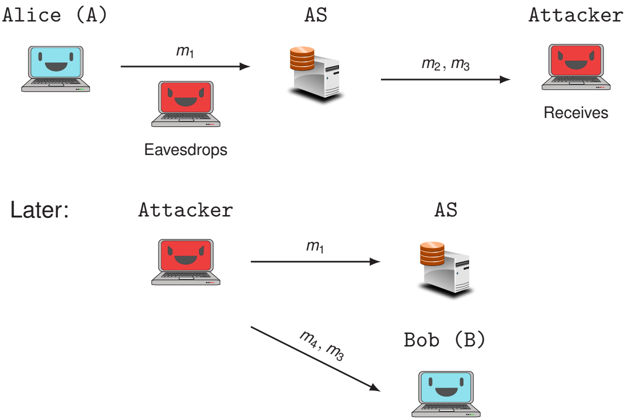
Attack Concepts against Cryptographic Protocols

Attacks

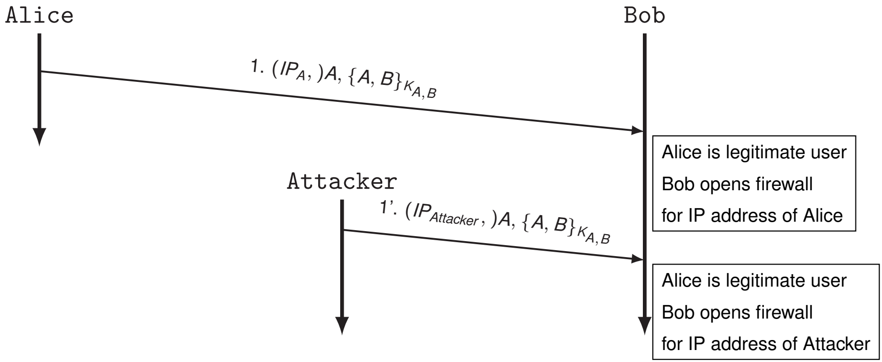
- Already known:
  - Eavesdropping
  - Man-in-the-Middle Attack
  - Cryptanalysis
- Attacker:
  - Can control parts or all of the network (see Dolev-Yao)
  - Eavesdrops and memorizes all it has seen
  - Can initiate protocol run
  - Can interfere with protocol runs
  - Can try to trick principles into running the protocol
  - For protocol analysis, it is usually not able to break crypto and hack the computers.

Replay Attack

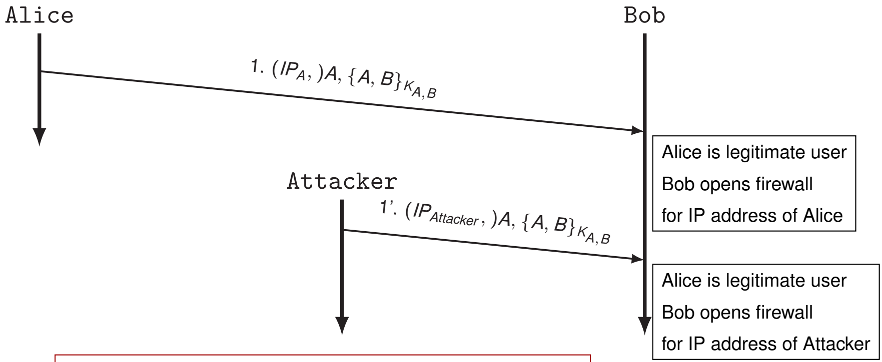
- Replay Attack: Receives and eavesdrops messages → later-on send message or part of message to some principle.



Replay Attack - Example



Replay Attack - Example

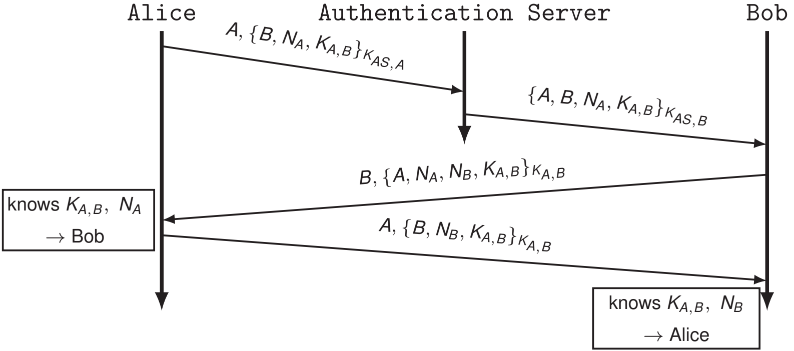


Bob cannot decide whether the message is fresh or not.

Reacting to an old message can result in security compromise!

Protocol Try 4 Replay Attack Defences

- An attacker can replay all messages of protocol try 3. None needs to be fresh.
- Better add a defense → Nonces $N_A, N_B, \ldots$



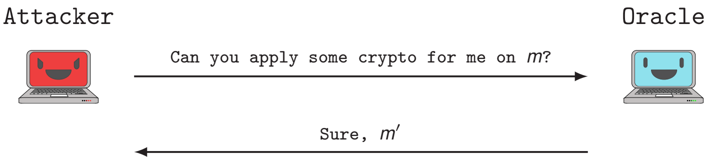
Authentication and Key Establishment Problem Statement - Version 2

Goals:

- Run a key exchange protocol such that at the end of the protocol:
- Alice and Bob have a shared session key for a secure channel
- Alice (Bob) must be able to verify that Bob (Alice) participated in the protocol run (authentication) and that he (she) is “alive” (freshness)

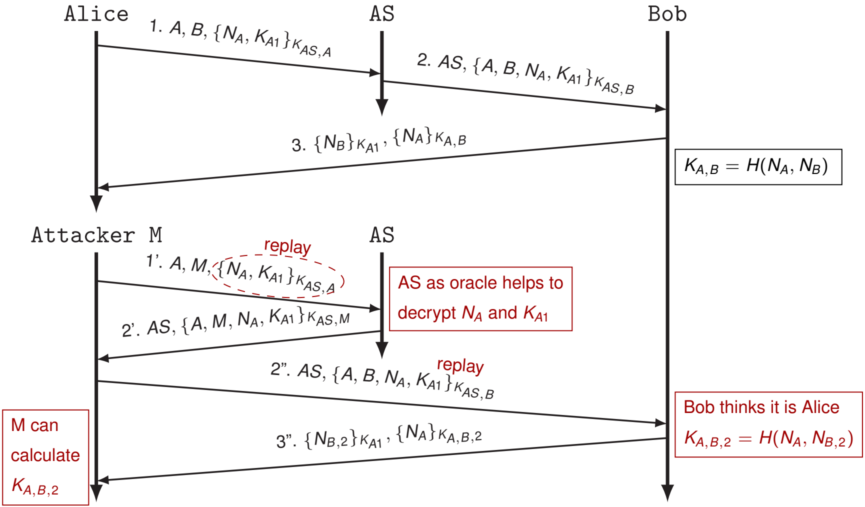
Oracle Attacks

- The attacker cannot break cryptography (assumption[2]).
- Yet maybe there are helpful principles that can help.
  - e.g. because they know the relevant keys
- Oracles are usually entities that can efficiently do something that a normal entity (here our attacker) cannot.



[2] see Dolev-Yao Attacker Model

Oracle Attacks - Example



Typing Attack

- Replace (usually encrypted) message field of one type with one of another (usually encrypted) type.

Alice

Attacker (M)

Shopping Server

in Protocol 1:

1. M, A, N_B

Attacker selects $N_B =$ "*Alice buys washing machine.*"

in Protocol 1:

2. $A, N_A, M, Sig_{K_{Alice\text{-}priv}}(N_B)$

Signing innocent nonce N_B in one protocol can mean signing a contract in another

in Protocol 2:

1. "*Alice buys washing machine.*", $Sig_{K_{Alice\text{-}priv}}($"*Alice buys washing machine.*"$)$

Other types of attacks

Think about more types of attacks. How would a protocol with a related weakness look like?

Other Types of Protocol Attacks

- Modification: Attacker alters messages sent.
- Preplay: The attacker takes part in a protocol run prior to a protocol run.
- Reflection: The attacker sends back protocol messages to principles who sent them. Related to Oracle attacks.
- Denial of Service: The attacker hinders legitimate principles to complete the protocol.
- Certificate Manipulation: Attacks using manipulated or wrongly-obtained certificates.
- Protocol Interaction: Make one protocol interact with another, e.g. by utilizing that principles use the same long-term keys in both protocols and utilizing that for an attack.

Desirable Properties of Cryptographic Protocols

Desirable Properties of Cryptographic Protocols

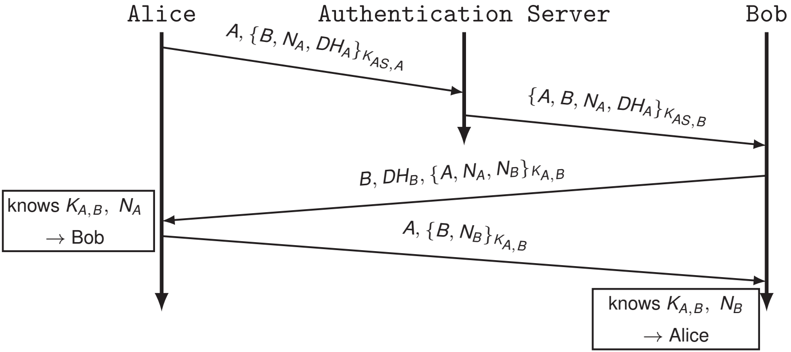
- Desirable Properties = what else we should want
- In this section:
  - Forward Secrecy and Key Agreement
  - Scalability
  - Avoidance of Single-Points-of-Failures
  - Selection of Algorithms
  - Generic Authentication Methods
  - Simplicity

Forward Secrecy (Repetition)

- Forward Secrecy (Repetition)
  - If long-term key is compromised, attacker cannot find out session key for older sessions.
  - If session key is compromised, other sessions and long-term key not affected.
- Can be achieved via Diffie-Hellman exchange.
  - DH_A is Diffie-Hellman information provided by Alice (e.g. in Textbook DH: $g, p, g^a \ mod \ p$)
  - DH_B is Diffie-Hellman information provided by Bob

Protocol Try 5 Adding Forward Secrecy

- Session key $K_{A,B}$ derived from DH_A and DH_B

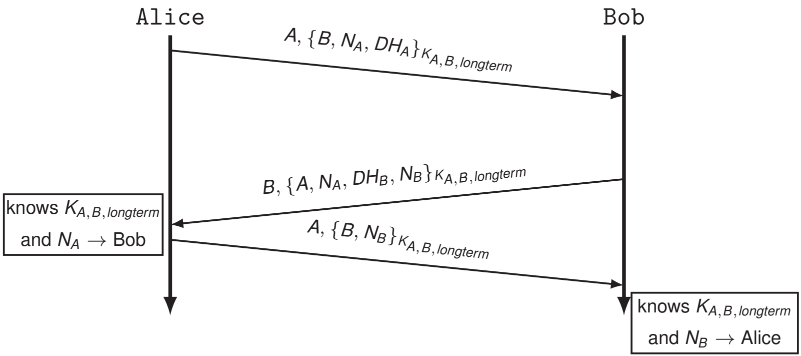


Scalability - revisted

- Scalability of keys $\rightarrow$ Authentication Server
- But having a central server is a single point of failure
- ... and bad for scalability of service
- Thus, good if server need not be contacted within a protocol run.
- While server may have provided keys or certificates (identity-key binding) beforehand.

Protocol Try 6 Removing Authentication Server

- Session key $K_{A,B}$ derived from Diffie-Hellman

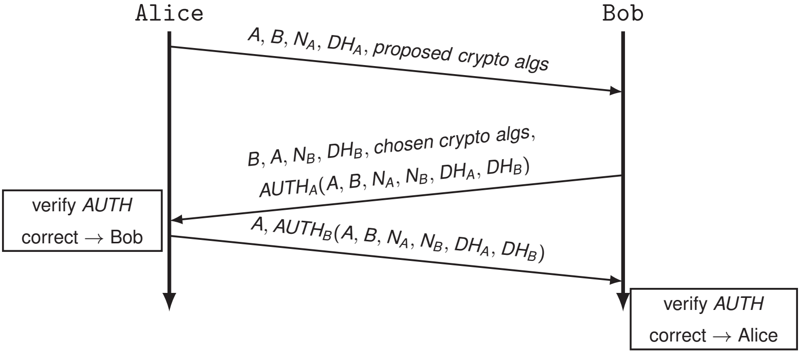


Selection of Used Algorithms

- Can we adapt the protocol, so that public key cryptography could be used?
- In practise, one might also want that all kinds of algorithms can be exchanged over time. $\rightarrow$ Do not become outdated!
- Concept:
  - Generic $Auth_X()$ function that can be realized with a suitable authentication function given either a public or shared key of X.
  - Alice and Bob have to agree on this function and used algorithms, e.g.
    - Alice proposes a set of functions and algoritms
    - Bob selects the ones that are then used

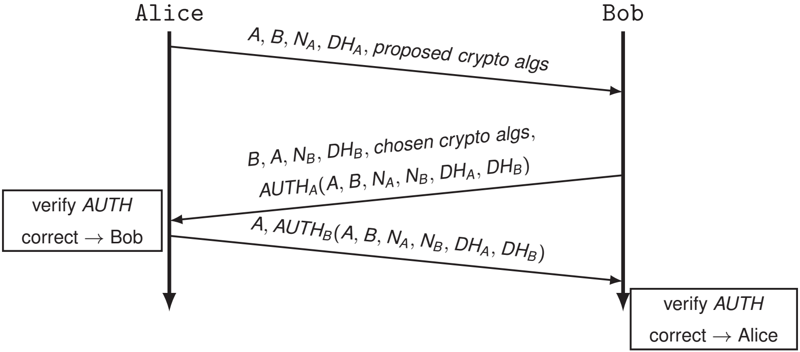
Protocol Try 7 Generic AUTH payload and Selection of Algorithms

- Session key $K_{A,B}$ derived from Diffie-Hellman



Protocol Try 7 Generic AUTH payload and Selection of Algorithms

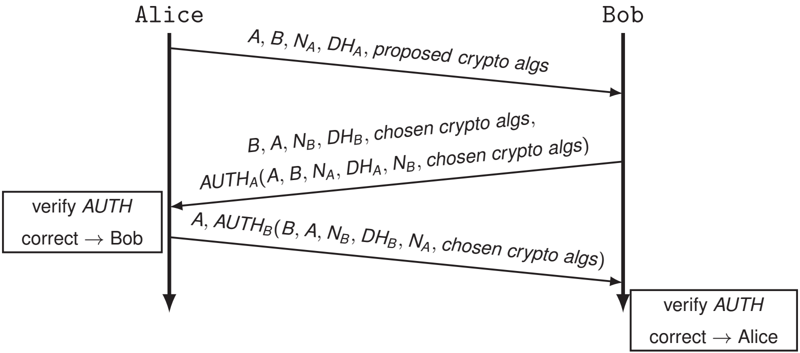
- Session key $K_{A,B}$ derived from Diffie-Hellman



- AUTH playload could be $MAC_{K_{A,B,longterm}}$. Then, Alice and Bob authenticate on identical messages → replay attack possible!

Protocol Try 8 AUTH Payload rework

- Session key $K_{A,B}$ derived from Diffie-Hellman



- AUTH playloads are different and contain information provided by both principles.

Simplicity

- Cryptographic protocols should be kept as simple as possible (but not any simpler)
- Complexity makes analysis harder and increases attack surface.
- Design Concept: Request-Response Pairs
  - $A \rightarrow B : Request1$
  - $B \rightarrow A : Response1$
  - ...

DoS Protection

- Cryptography is expensive (in particular asymmetric cryptography)
- Denial-of-Service attacker
  - Make victim do expensive operations
  - The attacker does not have to generate valid ciphertext, simple random numbers can work.
- Defense
  - Avoid expensive operations unless other principle has shown willingness to participate by replying with valid messages.
    - In final protocol try, we will avoid crypto until message 3.
  - Cookie mechanisms like TCP SYN Cookies could be used to avoid holding of state.

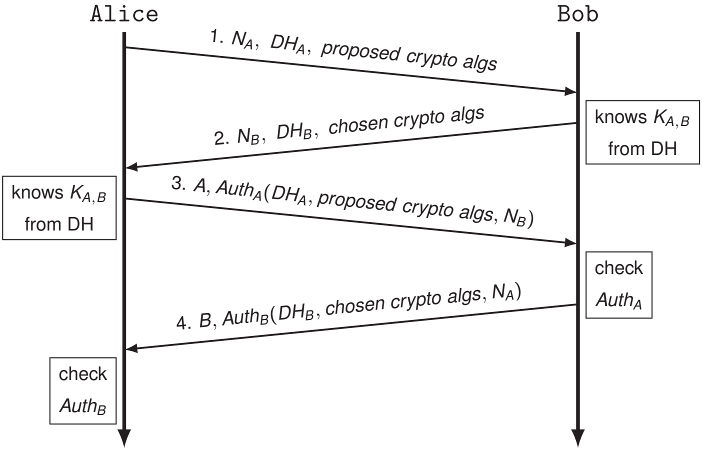
Final Protocol and Notation

Authentication and Key Establishment Problem Statement - Version 3 (FINAL)

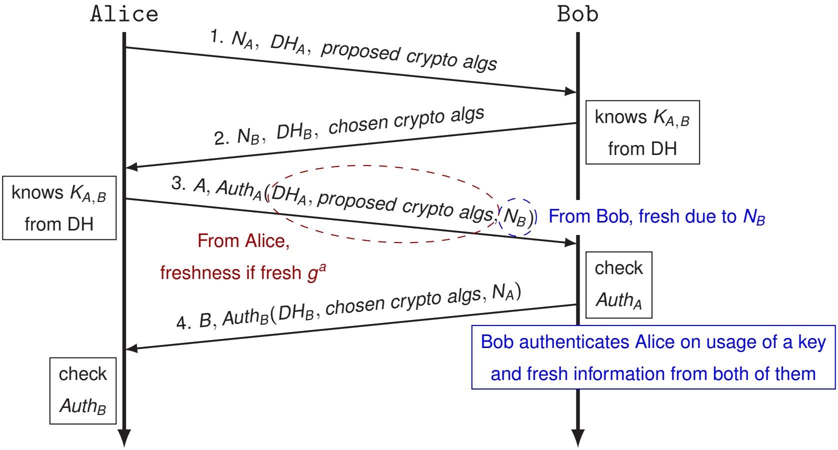
Goals:

- Run a key exchange protocol such that at the end of the protocol:
- Alice and Bob have a shared session key for a secure channel
- Alice and Bob have agreed on the cryptographic algorithms to be used for the secure channel
- Alice (Bob) must be able to verify that Bob (Alice) participated in the protocol run (authentication) and that he (she) is “alive” (freshness)
- Alice and Bob must know that $K_{A,B}$ is newly generated

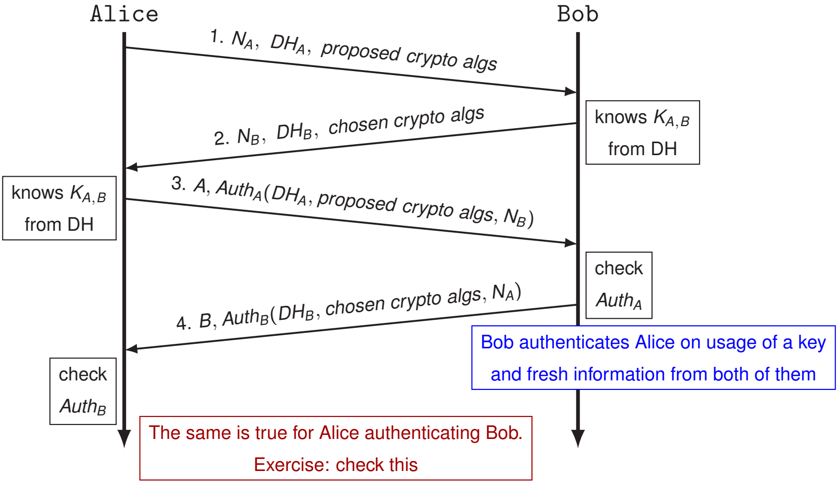
Final Protocol



Final Protocol



Final Protocol



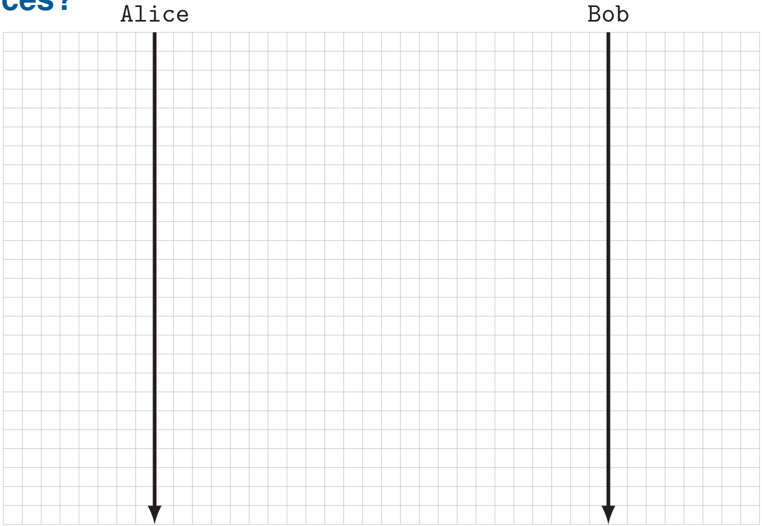
Final Protocol Explanations

- Explanation
  - Messages 1 and 2 form a request-response pair where only information is exchanged.
  - Messages 3 and 4 form the authentication request-response pair with identity information and authentication.
  - Alice or Bob need to stop the communication when authentication fails or a wrong entity authenticates.
  - Message 3: Alice authenticates on her first message and on the nonce N_B provided by Bob.
  - Message 4: Bob authenticates on his first message and on the nonce N_A provided by Alice.
- Final protocol is a simplified version of the IKEv2 protocol (IKE_SA_Init plus IKE_Auth Exchange) of IPSec (see IPSec chapter)

Repetition Exercise (later once we discussed IPSec): Compare with IKEv2

Write down "Final Protocol" in the terminology / fields used in IPSec.

Exercise: Final Protocol with Timestamps instead of Nonces?



Notation

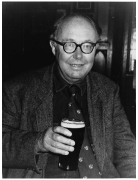
Notation	Meaning
A	Name of principle A (Alice), analogous for B, E, TTP, CA
CA_A	Certification Authority of A
r_A	Random value chosen by A
N_A	Nonce (number used once) chosen by A
t_A	Timestamp generated by A
$(m_1, .., m_n)$	Concatenation of m_1, ..., m_n
$A \rightarrow B : m$	A sends message m to B

Notation (continued)

Notation	Meaning
$K_{A\text{-}pub}$	Public Key of A
$K_{A\text{-}priv}$	Private Key of A
$K_{A,B}$	Shared symmetric key of A and B, only known to A and B
$H(m)$	Cryptographic hash value over m
$Enc_K(m)$	Encrypt m with key K, K can be symmetric or asymmetric
$Dec_K(c)$	Decrypt c with key K, K can be symmetric or asymmetric
$Sig_K(m)$	Signature of message m with key K, K is a private asymmetric key
$MAC_K(m)$	Message Authentication Code of m with key K, K is symmetric key
$\{m\}_K$	Message m encrypted and integrity-protected with symmetric key K
$[m]_K$	m integrity-protected with key K
$Cert_{CA}(A)$	Certificate of CA for public key $K_{A\text{-}pub}$ of A, signed by the private key of CA

Example: Needham Schroeder Protocol

Needham Schroeder Protocol



Roger Needham

Michael Schroeder

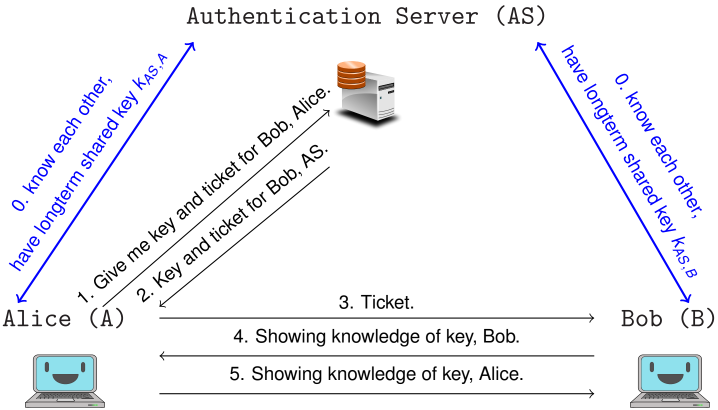
- Invented in 1978 by Roger Needham and Michael Schroeder [Nee78]
- The Needham-Schroeder Protocol is a protocol for mutual authentication and key establishment
- It aims to establish a session key between two users (or a user and an application server, e.g. email server) over an insecure network

Needham Schroeder Protocol - Introduction

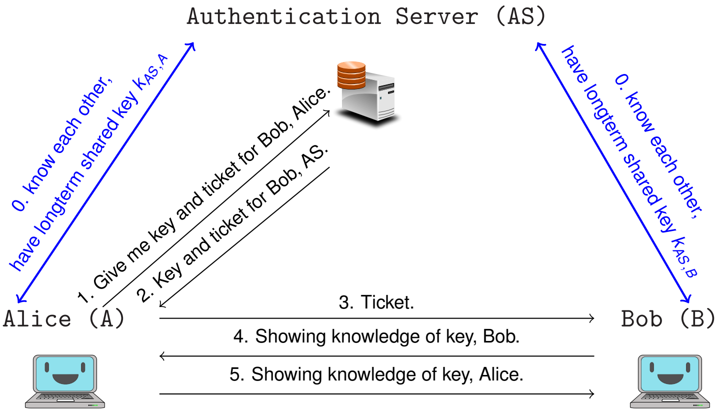
- The protocol has 2 versions:
  - The Needham Schroeder Symmetric Key Protocol: based on symmetric encryption, forms the basis for the Kerberos protocol
  - The Needham Schroeder Public Key Protocol: uses public key cryptography. A flaw in this protocol was published by Gavin Lowe [Lowe95] 17 years later! Lowe proposes also a way to fix the flaw in [Lowe95]

Gavin Lowe

Needham Schroeder Symmetric Key Protocol - Concept

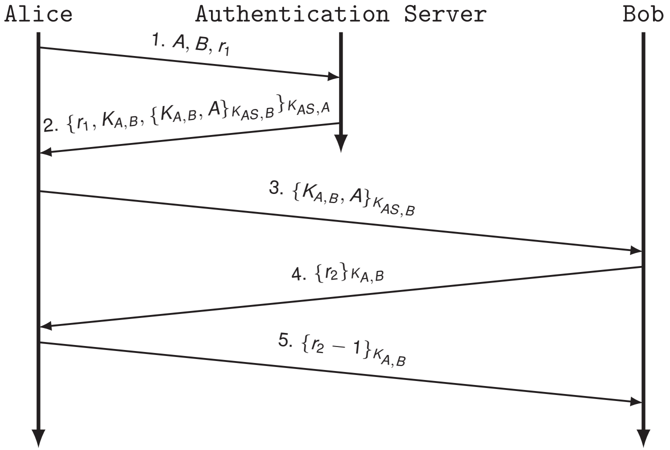


Needham Schroeder Symmetric Key Protocol - Concept

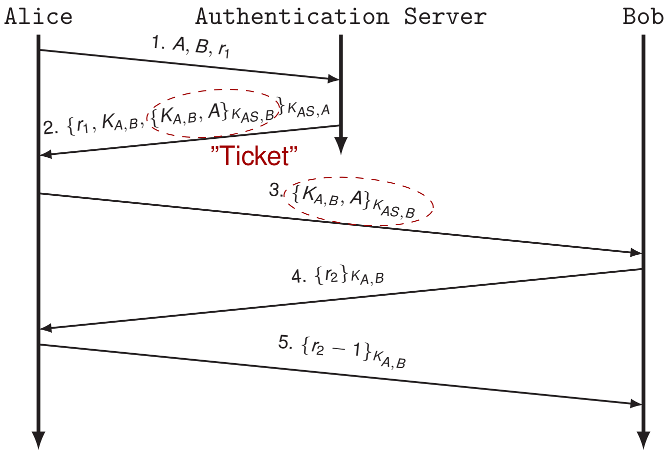


Only Alice and Bob (and the AS) know the shared key,
showing knowledge → one of them

Needham Schroeder Symmetric Key Protocol - Protocol



Needham Schroeder Symmetric Key Protocol - Protocol



Needham Schroeder Symmetric Key Protocol - Explanation

- 1. $A \rightarrow AS : A, B, r_1$
  - Alice informs AS that she (A) wants to contact Bob (B).
  - Random number r_1 is used as nonce to identify the session.
  - Notice, the AS cannot tell whether it is Alice or someone else. Still, this is ok as answer will be protected.
- 2. $AS \rightarrow A : \{r_1, K_{A,B}, \{K_{A,B}, A\}_{K_{AS,B}}\}_{K_{AS,A}}$
  - The AS encrypts the message with key $K_{AS,A}$ so that only Alice can read the message.
  - Alice notices nonce r_1 and assumes answer to be fresh.
  - Alice gets to know session key $K_{A,B}$ which is also part of the ticket.
  - Alice also gets to know the ticket $\{K_{A,B}, A\}_{K_{AS,B}}$.
  - Alice cannot read or modify ticket as it is protected with key $K_{AS,B}$ unknown to her.

 Needham Schroeder Symmetric Key Protocol - Explanation 2

- 3. $A \rightarrow B : \{K_{A,B}, A\}_{K_{AS,B}}$
  - Bob can decrypt the ticket and learns that Alice (A) wants to contact him.
  - Furthermore, he learns the session key $K_{A,B}$
- 4. $B \rightarrow A : \{r_2\}_{K_{A,B}}$
  - Bob sends nonce r_2 to Alice encrypted with the session key $K_{A,B}$
  - While Alice does not know about r_2, she knows $K_{A,B}$ as new session key. Integrity shows knowledge of session key by B, which means that B is Bob as only Bob (and the AS) also knows the session key.
- 5. $A \rightarrow B : \{r_2 - 1\}_{K_{A,B}}$
  - Alice sends nonce $r_2 - 1$ to Bob encrypted with the new session key $K_{A,B}$.
  - Since only Alice also knows r_2, this A must be Alice.
  - Notice, the change from r_2 to $r_2 - 1$ is to make messages 4 and 5 different to avoid e.g. replay attacks.
    - Modern encryption modes with Initialization Vectors (IV) also ensure this if both messages 4 and 5 would be $\{r_2\}_{K_{A,B}}$. However, an attacker could replay with the same IV and then the modified protocol would fail unless it takes further measures to forbid and prevent repeated IVs.

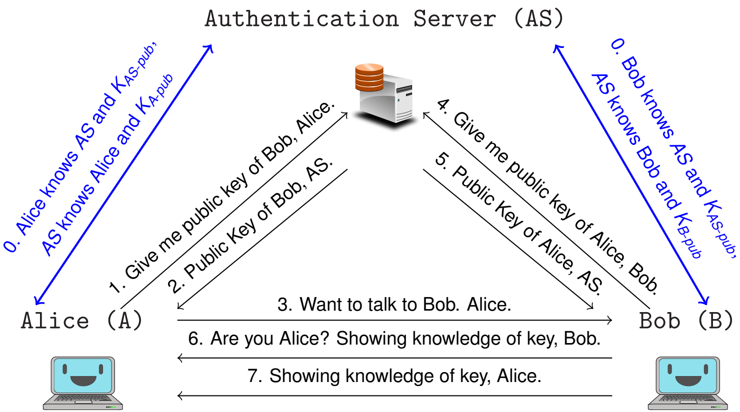
Needham Schroeder Symmetric Key Protocol - Ticket and Ticket Reuse

- Needham and Schroeder do not speak of tickets in their protocol, but from a modern point of view (relating to Kerberos) $\{K_{A,B}, A\}_{K_{AS,B}}$ is called a ticket.
- If Alice still trusts the ticket she has, Needham and Schroeder propose a shortened protocol:
  - 1. (3'.) $A \rightarrow B : \{K_{A,B}, A\}_{K_{AS,B}}, \{r_2\}_{K_{A,B}}$
  - 2. (4'.) $B \rightarrow A : \{r_3, r_2 - 1\}_{K_{A,B}}$
  - 3. (5'.) $A \rightarrow B : \{r_3 - 1\}_{K_{A,B}}$
- As the session key is not fresh anymore, Alice challenges Bob with r_2 and Bob Alice with r_3.

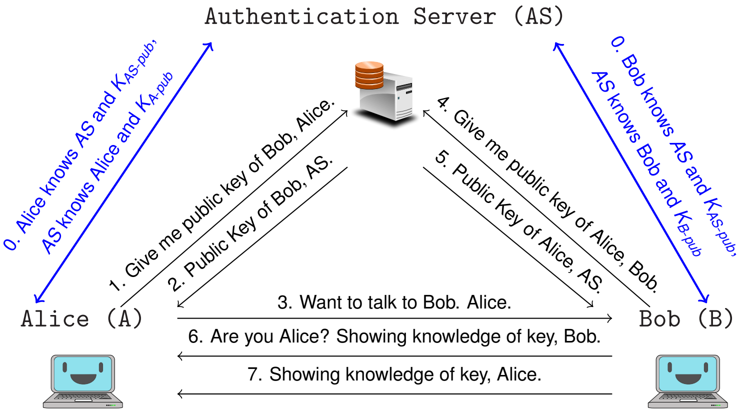
Needham Schroeder Symmetric Key Protocol - Ticket Reuse Issues and Forward Secrecy

- If an attacker learns about session key $K_{A,B}$ and observed the related ticket in a previous protocol run, then the attacker can impersonate Alice.
  - 1. (3'.) $Attacker \rightarrow B : \{K_{A,B}, A\}_{K_{AS,B}}, \{r_2\}_{K_{A,B}}$
  - 2. (4'.) $B \rightarrow A : \{r_3, r_2 - 1\}_{K_{A,B}}$ needs to be intercepted and decrypted by attacker.
  - 3. (5'.) $Attacker \rightarrow B : \{r_3 - 1\}_{K_{A,B}}$
- Thus, breaking session key $K_{A,B}$ would allow to impersonate Alice in the future.
- Also, the Needham Schroeder Protocols do not provide any forward secrecy.

Needham Schroeder Public Key Protocol - Concept

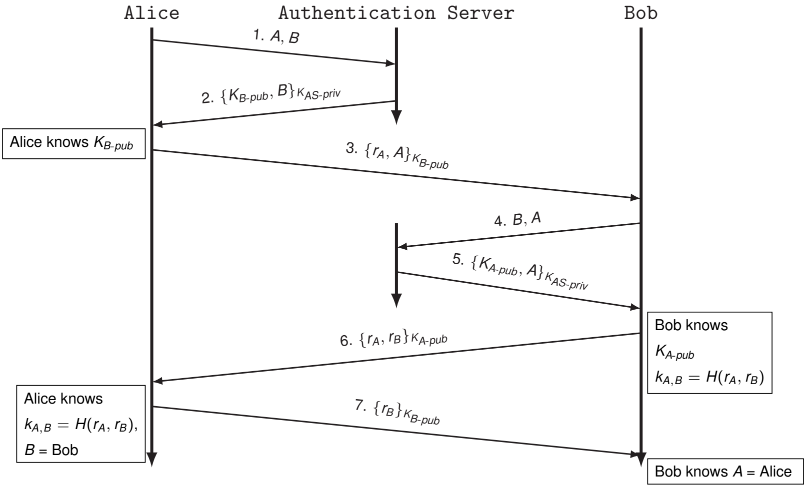


Needham Schroeder Public Key Protocol - Concept



AS asserts relationship public key ↔ person (Alice or Bob),
showing knowledge of the related private key → related person

Needham Schroeder Public Key Protocol - Protocol



As a one-time exception, we will use $\{\cdot\}$ with asymmetric keys. Do not mix up encryption/signing in practice!

Needham Schroeder Public Key Protocol - Explanation

- 1. $A \rightarrow AS : A, B$
- 2. $AS \rightarrow A : \{K_{B\text{-}pub}, B\}_{K_{AS\text{-}priv}}$
  - In this exchange, Alice asks for the public key of Bob. Her identity is irrelevant. Anyone can ask for Bob's public key.
  - AS encrypts $K_{B\text{-}pub}, B$ with its private key. Anyone can decrypt, but only the AS can generate this ”signature”.
- 3. $A \rightarrow B : \{r_A, A\}_{K_{B\text{-}pub}}$
  - Alice sends Bob a challenge r_A and the identity A that she claims to be (not yet proven!).
  - Only Bob can decrypt the message with his private key, so only he can know r_A later-on.
- 4. $B \rightarrow AS : B, A$
- 5. $AS \rightarrow B : \{K_{A\text{-}pub}, A\}_{K_{AS\text{-}priv}}$
  - 4. and 5. are the same as 1. and 2.

Needham Schroeder Public Key Protocol - Explanation 2

- 6. $B \to A : \{r_A, r_B\}_{K_{A\text{-}pub}}$
  - Bob answers Alice's challenge r_A. So, Alice knows he is Bob.
  - Bob challenges Alice with r_A. As her public key is used, only she can decrypt the message and know r_B.
  - The shared session key is $K_{A,B} = H(r_A, r_B)$, with H being a cryptographic hash function. As r_A and r_B are only sent encrypted with the public key of either Alice or Bob, no other entity knows r_A, r_B, and thus $K_{A,B}$.
- 7. $A \to B : \{r_B\}_{K_{B\text{-}pub}}$
  - Alice answers Bob's challenge r_B. So, Bob knows she is Alice.

Exercise: Proper usage of Encryption and Signing

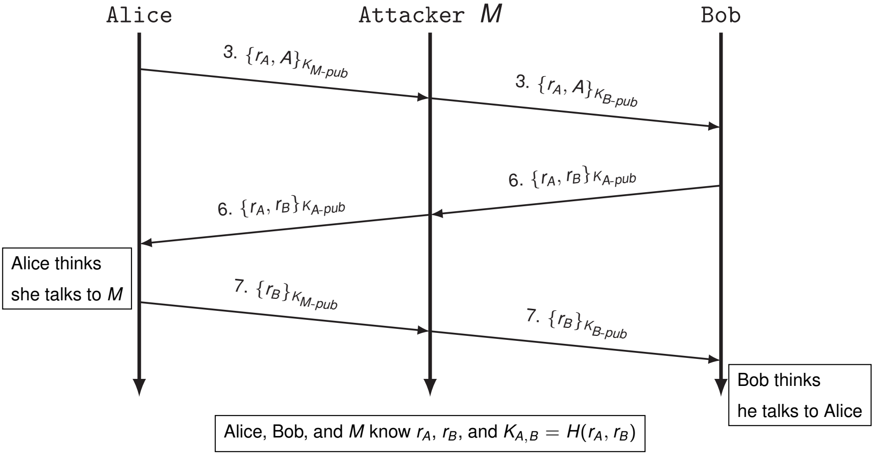
On the previous slides, we used $\{\cdot\}$ with asymmetric keys. It should combine $Enc_k(\cdot)$ and $Sig_k(\cdot)$. Why is this a bad idea in practice? How should the protocol look with only using $Enc_k(\cdot)$ and $Sig_k(\cdot)$?

Needham Schroeder Public Key Protocol - Attack

- In 1995, Lowe found a man-in-the-middle attack on the Needham Schroeder Public Key Protocol.
- Assumption: Attacker M can trick Alice A into a communication with him. So, Alice starts a communication session with M.
- Idea: make Bob believe, he talks to Alice instead of the attacker.

Needham Schroeder Public Key Protocol - Attack

- We skip the exchanges with the AS to obtain the public keys.



Needham Schroeder Public Key Protocol - Attack Resolution

- The attack fails when message 6 is modified to:
  6. $B \rightarrow A : \{r_A, r_B, B\}_{K_{A\text{-}pub}}$
- Exercise: Verify that the attack will now fail.

Conclusions - What have we learned

What have we learned

- Authentication and Key Establishment
  - Related to Formal Reasoning
  - Secure Authentication needs some pre-established keys, also see PKI chapter
  - Protocol weaknesses can be tricky
  - Learned to attack protocols on conceptual level
  - Learned some protocols, remember the ones with actual names[3]
  - Learned how authenticity and key establishment can be achieved
- Analyze protocols on the layers they operate
- Analyze complete systems over all layers

[3] "Try *N*" is not a name

Literature

- C. Boyd, A. Mathuria. *Protocols for Authentication and Key Establishment*, Springer, 2003.
- G. Schäfer. *Netzsicherheit - Algorithmische Grundlagen und Protokolle*, dpunkt Verlag, 2003.
- N. Ferguson, B. Schneier. *Practical Cryptography*, John Wiley & Sons, 2003.
- G. Lowe. *An Attack on the Needham-Schroeder Public-Key Authentication Protocol*, Information Processing Letters, volume 56, number 3, pages 131-133, 1995.
- R. Needham, M. Schroeder. *Using Encryption for Authentication in Large Networks of Computers.*, Communications of the ACM, Vol. 21, No. 12, 1978.