

Secure Communication Channels

March 22, 2020

Roadmap

Introduction

Shared Key Authentication Protocols

Public-key Authentication Protocols

Session Keys

Secure Channel Implementation Layer

Key Management

Further Reading

Roadmap

Introduction

Shared Key Authentication Protocols

Public-key Authentication Protocols

Session Keys

Secure Channel Implementation Layer

Key Management

Further Reading

Secure Communication Channels

- ▶ Many network security problems can be mitigated with the help of **secure channels**, which can guarantee:
 - Authentication** of the communicating parties, i.e. that the entities at the ends of the channel are who they claim to be
 - Integrity/Authenticity** i.e. that the messages were not modified in transit
 - Confidentiality** i.e. that an attacker cannot observe the contents of a message
- ▶ Usually, integrity or confidentiality do not make sense without authentication
 - ▶ And authentication does not make much sense without integrity

Authentication

- ▶ Usually, during the setting up of a secure (communication) channel the two entities authenticate each other
 - ▶ Actually, on the Web, most often only one of the entities is authenticated
- ▶ Often, the channel set up phase, includes also the establishment of a **session key** that is used to ensure integrity or confidentiality
- ▶ Passwords are not appropriate for authentication while setting up a secure channel
 - ▶ Instead, one often uses *challenge/response* protocols

Roadmap

Introduction

Shared Key Authentication Protocols

Public-key Authentication Protocols

Session Keys

Secure Channel Implementation Layer

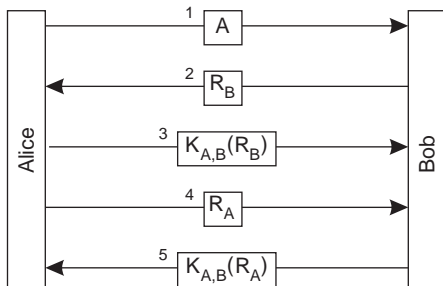
Key Management

Further Reading

Shared Key Authentication Protocols

Assumptions The parties (Alice/A and Bob/B) at the two ends share a secret key ($K_{A,B}$)

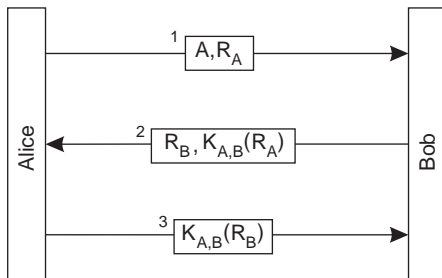
- ▶ How they can get the secret key, will be discussed shortly



- ▶ Messages 2 and 3 allow B to authenticate A;
- ▶ Messages 4 and 5 allow A to authenticate B;

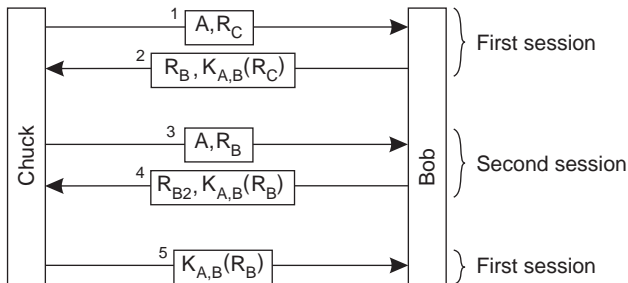
"Optimized" Authentication Protocol with Shared Key

- We could **optimize** this protocol as follows:



- But this is vulnerable to a **reflection attack**.

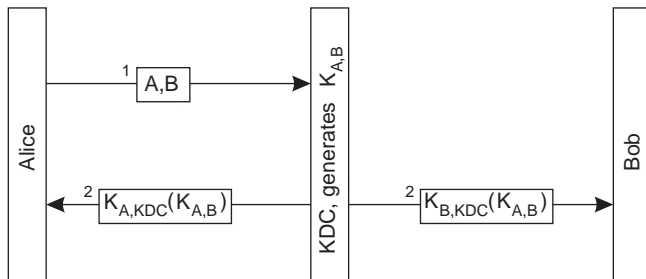
Reflection Attack



- ▶ The problem is that the 2 parties use the same challenge in two different executions
- ▶ A principle to avoid reflection attacks is to make the protocol asymmetrical:
 - ▶ E.g. require one party to use an odd challenge and the other one an even challenge

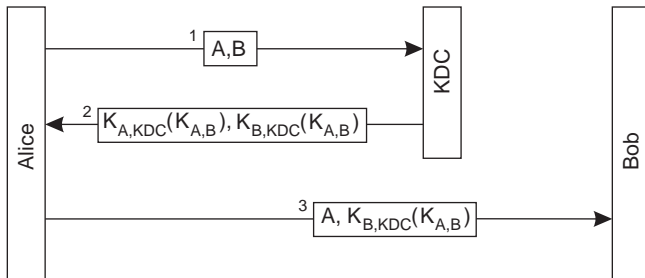
Mediated Authentication (with KDC) (1/2)

- ▶ Shared key authentication (without a KDC) is not scalable:
 - ▶ Each pair of principals must share a secret key.
- ▶ A solution is to use an mediator, the *Key Distribution Center* (KDC), in which all principal must **trust**.
 - ▶ The KDC shares a secret key with each principal
 - ▶ Upon request, the KDC generates secret keys for sharing between principals that wish to communicate securely



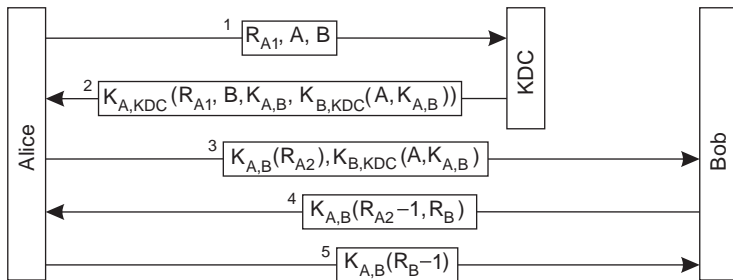
Mediated Authentication (with KDC) (2/2)

- ▶ What if B receives A's first message to B, before it receives the key from the KDC?



- ▶ This protocol is not complete:
 - ▶ A and B must authenticate mutually
 - ▶ I.e. prove that they know $K_{A,B}$

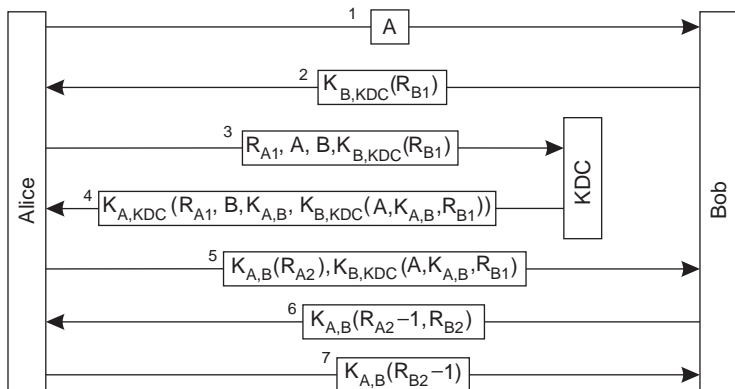
Needham-Schroeder's Protocol (1/2)



- ▶ The *nonce* (R_{A1}) is used to ensure that A communicates with the KDC (it prevents *replay attacks*).
- ▶ The KDC includes B's identity in the response, to prevent C from impersonating B, by replacing B with C, in message 1.
- ▶ Messages 3 and 4 allow A to authenticate B
- ▶ Messages 4 and 5, allow B to authenticate A
- ▶ $K_{B,KDC}(A, K_{A,B})$ in message 2 is known as the *ticket to Bob*

Needham-Schroeder's Protocol (2/2)

- ▶ In 1981, Denning and Sacco found a vulnerability, if C learns A's key ($K_{A,KDC}$), even if it has been replaced:
 - ▶ In this case, C may impersonate A, in its communication with B
- ▶ In 1987, Needham and Schroeder published a new version of the protocol that fixed that vulnerability



Roadmap

Introduction

Shared Key Authentication Protocols

Public-key Authentication Protocols

Session Keys

Secure Channel Implementation Layer

Key Management

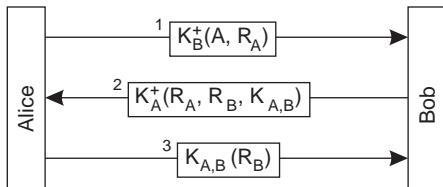
Further Reading

Public-key Authentication

Assumptions

1. The principals (Alice/A and Bob/B) know the public keys of one another
 - ▶ Below, we discuss how one can learn the public key of a principal
2. Private keys are *known exclusively* by the respective principal

Basic Protocol



- ▶ In addition to authenticate both principals, this protocol also negotiates a **session key**
 - ▶ The share key can be used to ensure confidentiality and integrity

Roadmap

Introduction

Shared Key Authentication Protocols

Public-key Authentication Protocols

Session Keys

Secure Channel Implementation Layer

Key Management

Further Reading

Data Integrity/Confidentiality

- ▶ Authentication upon setting up a secure channel is not enough
 - ▶ If after authentication no measures are taken, i.e. the messages are exchanged in the clear, an attacker can not only intercept these messages but also modify them
- ▶ To ensure authenticity/integrity and confidentiality of the messages exchanged after authentication, A and B can use cryptography
- ▶ Although encrypting a message can ensure confidentiality, it may not be sufficient to ensure integrity
 - ▶ In secure communication, by **integrity** we mean that the channel should be able to detect modification of the messages
 - ▶ But the secure channel may not have enough context information to determine whether or not the message has been modified

One needs to use **authenticated encryption**

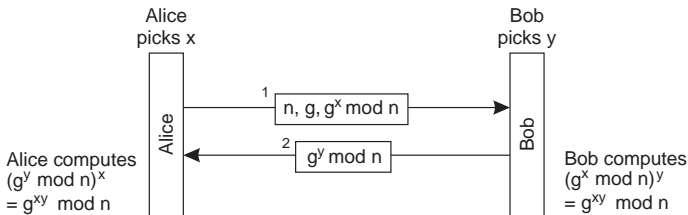
- ▶ An obvious approach is **Encrypt-then-MAC (EtM)**, i.e. first encrypt the message then compute a MAC of the ciphertext (note that different keys should be used)
- ▶ But there are approaches that use a single key

Session Key

- ▶ To ensure confidentiality, it is important to use a **session key** that is different from the key used for principal authentication:
 - ▶ Public key encryption is less efficient than shared key encryption
 - ▶ Keys *wear out* with use: the more ciphertext an attacker has, more likely it is she will succeed finding the key
 - ▶ The use of the same key over multiple sessions, makes *replay attacks* more likely to succeed
 - ▶ If a session key is compromised, the attacker will not be able to decrypt messages exchanged in other sessions
- ▶ Actually, most secure channels have provisions to change keys in the middle of a session
 - ▶ This prevents compromising the key or replay attacks in long running sessions

Diffie-Hellman Key-Agreement Protocol (1/2)

- ▶ Let n be a large prime and g a number less than n with some properties (for additional security)
 - ▶ These number must be known *a priori*, and may be public
- ▶ Each principal chooses a private and secret large number, x and y and executes the following protocol:



- ▶ The session key can be computed as $g^{xy} \bmod n$

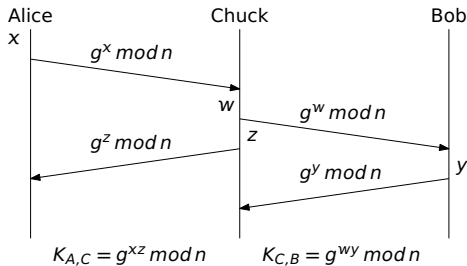
Question Why can't an attacker overhearing the communication do the same?

Answer It would have to compute the discrete logarithm, which is considered computationally intractable

- ▶ However, there are known efficient algorithms for special cases

Diffie-Hellman Key-Agreement Protocol (2/2)

- ▶ As described, the DH protocol is vulnerable to a man-in-the-middle attack:



- ▶ To defend it against such an attack, we can use:
 - Published DH numbers** e.g. A would publish $g^x \bmod n$ somewhere, and always reuse x for computing the session key (same for B)
 - Authenticated DH** i.e. messages of the DH-protocol are authenticated to prevent tampering. This requires:
 - ▶ Either a secret key shared among A and B
 - ▶ Or a public-keys (and private) for A and B

Perfect Forward Secrecy

Question Why do we need yet another protocol (DH)?

- ▶ It is possible to generate a secure session key from the nonces, i.e. random numbers, that are usually exchanged by the authentication protocols
- ▶ If we follow certain rules

Answer **Perfect forward secrecy**, i.e. an attacker will not be able to decrypt a recorded session even if (s)he later

- ▶ Breaks into both A and B
- ▶ Steals their long-term secrets

as long as A and B delete their secret numbers (x and y , respectively)

Watch out Schor has invented polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer

- ▶ DH and RSA will become insecure when quantum computing becomes practical
- ▶ According to the Wikipedia, the largest integer factored using a quantum computer was 291 311, a 19-bit value, in 2017

Session Key for Unidirectional Authentication

- ▶ The Web (SSL/TLS) uses mostly public key cryptography **unidirectional** authentication:
 - ▶ Clients authenticate Web servers using public key cryptography
 - ▶ Servers do not authenticate clients
 - ▶ Public key management at Internet scale is not easy
- ▶ In this case, the session key can be computed as follows:
 1. The client can generate (randomly) the session key and send it to the server encrypted with the latter's public key
 2. Client and server can execute Diffie-Hellman, but only the server's messages are authenticated
- ▶ In any case:
 - ▶ The client is guaranteed (under some assumptions) that it has set up a secure channel with the server
 - ▶ The server
 - ▶ has **no** idea who is on the other end of the channel
 - ▶ is guaranteed that on the other end of the channel is always the same client

Roadmap

Introduction

Shared Key Authentication Protocols

Public-key Authentication Protocols

Session Keys

Secure Channel Implementation Layer

Key Management

Further Reading

Secure Channel Implementation Layer

- ▶ In principle, it is possible to implement a secure channel at any layer of the communication stack:

Data Link e.g. Wi-Fi Protected Access (WPA)

- ▶ Protects only the communication in one "segment".

Network - e.g. IPSec

- ▶ Usually this is implemented by the OS
- ▶ Uses IP addresses for identification – and authentication

Transport - e.g. SSL/TLS

- ▶ Requires applications to be modified – (*sockets*);

Application - e.g. `ssh`, SMIME, PGP

- ▶ Protects application-specific *objects*, e.g. email messages stored on servers

Roadmap

Introduction

Shared Key Authentication Protocols

Public-key Authentication Protocols

Session Keys

Secure Channel Implementation Layer

Key Management

Further Reading

Key Distribution

Problem How to get the keys required by the authentication protocol?

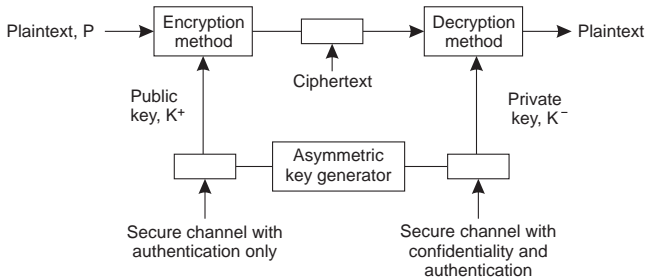
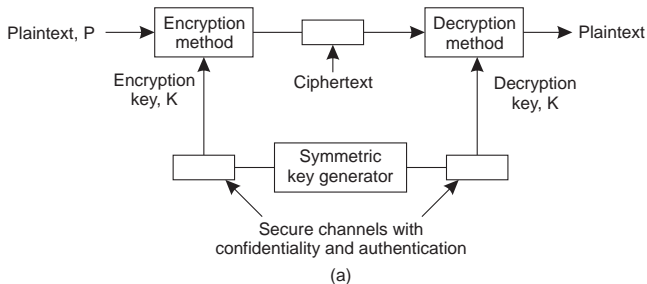
- ▶ All protocols assume that a principal knows a key bound to the other principal

Shared key in the case of symmetrical cryptographic systems;

Public key in the case of asymmetrical cryptographic systems

Solution Depends on the type of cryptographic system

Authentication Key Distribution



Public Key Certificates (1/3)

- ▶ The challenge with shared keys is to ensure that they are kept secret
- ▶ The challenge with public keys is to ensure the binding between a public key and a principal
 - ▶ Cryptographic protocols can only check whether a public key matches a private key
 - ▶ If C convinces A that B 's public key is a key matches a private key C knows, then C can impersonate B
- ▶ The solution to address this challenge is based on **public-key certificates/digital certificate**, which contain:
 1. The subject's (principal's) name
 2. A public key that matches the subject's private key
 3. A signature of the remaining information by a **Certification Authority (CA)**
 4. The name of the CA

Public Key Certificates (2/3)

- ▶ The assumption is that the CA's public keys are well known
 - ▶ Web browsers are shipped with the public keys of (too) many CAs
- ▶ When a browser validates/accepts a certificate, the user **trusts** that the principal's it wants to communicate with has the public key in the certificate
 - ▶ But CAs can be tricked into issuing certificates for someone else to attackers (e.g. Verisign issued a certificate for Microsoft)
- ▶ On the Web, the trust model used by digital certificates is not only oligarchic but also hierarchical
 - ▶ A CA can delegate on other CAs, i.e. issue a certificate vouching for their trustworthiness
- ▶ PGP uses the "anarchy model"
 - ▶ Each user can choose which public-keys it trusts (**trust-anchors**)
 - ▶ A user can sign certificates for anyone else
 - ▶ To get a key's certificate, one needs to find a path starting on some trust anchor
- ▶ A key issue in this scheme is naming: is the John Smith whose key I need the John Smith in some certificate?

Public Key Certificates (3/3)

- ▶ Digital certificates have expiration dates, often of several months or even years:
 - ▶ Browsers typically warn the user when a server presents an expired certificate

Problem What if the private key is compromised before the certificate expires?

Solution Revoke the certificate

- ▶ CAs periodically publish **Certificate Revocation List (CRL)** with certificates that have been revoked.
 - ▶ What should the period be?
 - ▶ To reduce the amount of data transferred, CAs can publish a full CRL with a larger period and delta CRLs with a shorter period
- ▶ The Online Certificate Status Protocol (OCSP) (RFC 2560) allows browsers to verify the validity of a certificate in real-time
 - ▶ If the CA runs an on-line revocation server

Roadmap

Introduction

Shared Key Authentication Protocols

Public-key Authentication Protocols

Session Keys

Secure Channel Implementation Layer

Key Management

Further Reading

Further Reading

- ▶ Chapter 9, Tanenbaum e van Steen, *Distributed Systems, 3rd Ed.*
 - ▶ Section 9.2: *Secure Channels*
 - ▶ Subsection 9.5[.1]: *Key Management*