CNT4406/5412 Network Security

Security Handshake Pitfalls

Zhi Wang

Florida State University

Fall 2014

Introduction

- Secure communication almost always includes an initial authentication handshake
  ➠ in addition to integrity protection and/or encryption of data

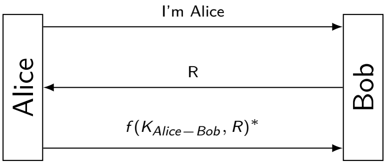
Introduction

- Secure communication almost always includes an initial authentication handshake
  - ➠ in addition to integrity protection and/or encryption of data
- Designing a secure authentication handshake is not trivial
  - ➠ different protocols have different trade-offs
  - ➠ different situations require different protocols

One-way Shared Secret Authentication

Weakness:

- Authentication is not mutual

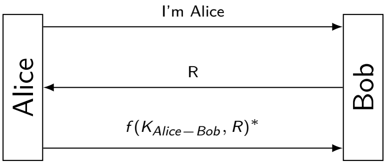


$^*f$ is a secret key crypto or a hash function

One-way Shared Secret Authentication

Weakness:

- Authentication is not mutual
- Trudy may hijack the conversation after initial exchange (if this is the entire protocol)

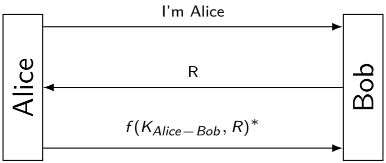


$^*f$ is a secret key crypto or a hash function

One-way Shared Secret Authentication

Weakness:

- Authentication is not mutual
- Trudy may hijack the conversation after initial exchange (if this is the entire protocol)
- An eavesdropper could mount an off-line password-guessing attack

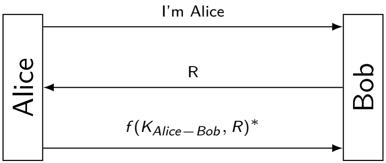


$^*f$ is a secret key crypto or a hash function

One-way Shared Secret Authentication

Weakness:

- Authentication is not mutual
- Trudy may hijack the conversation after initial exchange (if this is the entire protocol)
- An eavesdropper could mount an off-line password-guessing attack
- Trudy may compromise Bob's database and later impersonate Alice

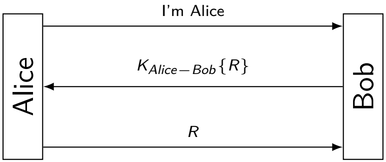


$^*f$ is a secret key crypto or a hash function

One-way Shared Secret Authentication: A Variant

Differences from the previous scheme:

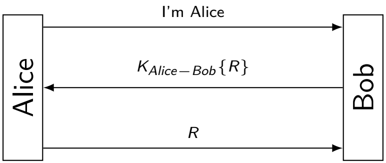
- Function f needs to be reversible (cryptography but not hash)



One-way Shared Secret Authentication: A Variant

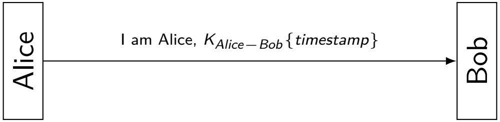
Differences from the previous scheme:

- Function f needs to be reversible (cryptography but not hash)
- Trudy can mount a dictionary attack without eavesdropping
  - ➠ R needs to be verifiable, such as having a structure
  - ➠ e.g., R is a 32-bit random number padded on the right zeros



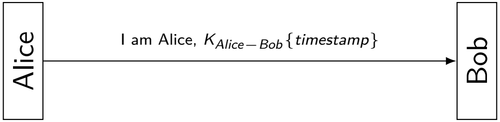
One-way Shared Secret Authentication: More Variant

- It is simpler to replace sending cleartext password



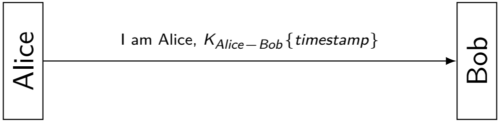
One-way Shared Secret Authentication: More Variant

- It is simpler to replace sending cleartext password
- The protocol is more efficient than the original
  - ➠ one message v.s. three messages; no states to keep for Bob



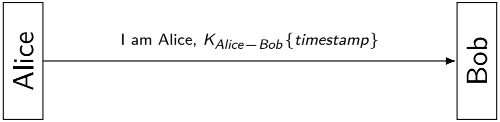
One-way Shared Secret Authentication: More Variant

- It is simpler to replace sending cleartext password
- The protocol is more efficient than the original
  ➠ one message v.s. three messages; no states to keep for Bob
- Trudy can impersonate Alice within the acceptable clock skew
  ➠ to the same server; to other servers (how to prevent?)



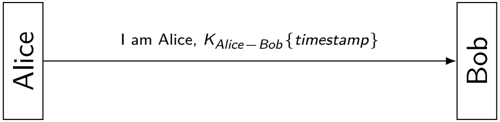
One-way Shared Secret Authentication: More Variant

- It is simpler to replace sending cleartext password
- The protocol is more efficient than the original
  ➠ one message v.s. three messages; no states to keep for Bob
- Trudy can impersonate Alice within the acceptable clock skew
  ➠ to the same server; to other servers (how to prevent?)
- Trudy can reuse encrypted timestamps if Bob sets his clock back



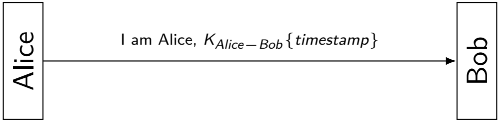
One-way Shared Secret Authentication: More Variant

- It is simpler to replace sending cleartext password
- The protocol is more efficient than the original
  ➠ one message v.s. three messages; no states to keep for Bob
- Trudy can impersonate Alice within the acceptable clock skew
  ➠ to the same server; to other servers (how to prevent?)
- Trudy can reuse encrypted timestamps if Bob sets his clock back
- How to use hash instead of encryption in this protocol?



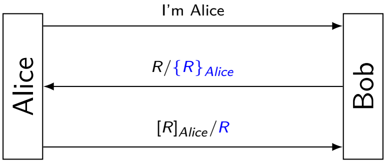
One-way Shared Secret Authentication: More Variant

- It is simpler to replace sending cleartext password
- The protocol is more efficient than the original
  ➠ one message v.s. three messages; no states to keep for Bob
- Trudy can impersonate Alice within the acceptable clock skew
  ➠ to the same server; to other servers (how to prevent?)
- Trudy can reuse encrypted timestamps if Bob sets his clock back
- How to use hash instead of encryption in this protocol?
  ➠ “I'm Alice, timestamp, $hash(K_{Alice-Bob}, timestamp)$”



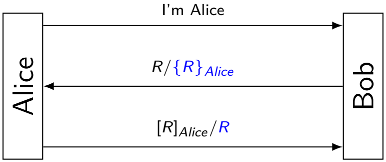
One-way Public Key Authentication

- Two variants: V_1: $R \rightarrow [R]_{Alice}$, V_2: $\{R\}_{Alice} \rightarrow R$



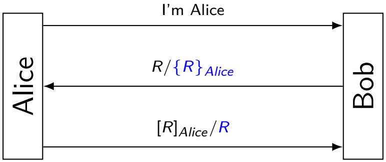
One-way Public Key Authentication

- Two variants: V_1: $R \rightarrow [R]_{Alice}$, V_2: $\{R\}_{Alice} \rightarrow R$
- Reading Bob's database is no longer security sensitive, why?
  ➠ unauthorized modification still is



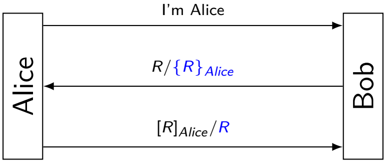
One-way Public Key Authentication

- Two variants: V_1: $R \rightarrow [R]_{Alice}$, V_2: $\{R\}_{Alice} \rightarrow R$
- Reading Bob's database is no longer security sensitive, why?
  ➠ unauthorized modification still is
- Trudy can trick Alice to provide a service to her (as a RPC):



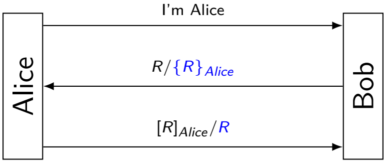
One-way Public Key Authentication

- Two variants: V_1: $R \rightarrow [R]_{Alice}$, V_2: $\{R\}_{Alice} \rightarrow R$
- Reading Bob's database is no longer security sensitive, why?
  ➠ unauthorized modification still is
- Trudy can trick Alice to provide a service to her (as a RPC):
  ➠ to sign something in V_1; to decrypt something in V_2



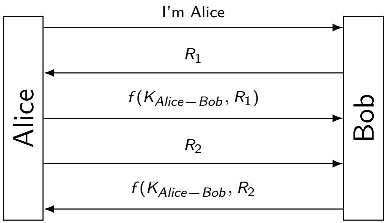
One-way Public Key Authentication

- Two variants: V_1: $R \rightarrow [R]_{Alice}$, V_2: $\{R\}_{Alice} \rightarrow R$
- Reading Bob's database is no longer security sensitive, why?
  - ➠ unauthorized modification still is
- Trudy can trick Alice to provide a service to her (as a RPC):
  - ➠ to sign something in V_1; to decrypt something in V_2
  - ➠ use structures to distinguish types of messages (e.g., PKCS)
  - ➠ use different keys for different purposes (e.g., auth, email...)



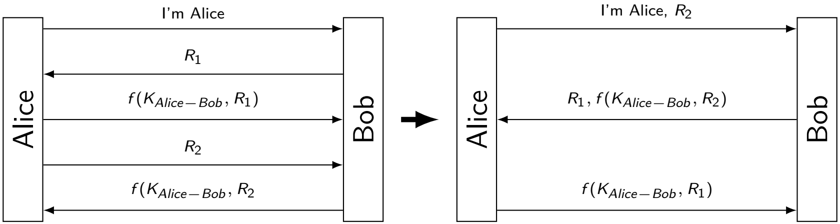
Mutual Authentication

- Bob and Alice challenge each other to achieve mutual authentication



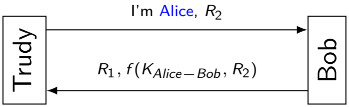
Mutual Authentication

- Bob and Alice challenge each other to achieve mutual authentication
- The protocol is inefficient ➠ reduce it into three messages (flaw?)



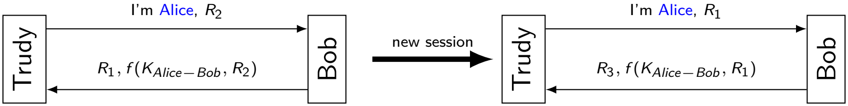
Reflection Attack

- Trudy connects to Bob and receives the challenge (R_1) from Bob
  - ➠ Trudy doesn't know how to respond it without $K_{Alice-Bob}$



Reflection Attack

- Trudy connects to Bob and receives the challenge (R_1) from Bob
  ➠ Trudy doesn't know how to respond it without $K_{Alice-Bob}$
- Trudy starts a second session and tricks Bob to encrypt R_2 for her
  ➠ Trudy can now impersonate Alice on the first connection


Trudy
Bob
I'm Alice, R_2
R_1, $f(K_{Alice-Bob}, R_2)$
new session
Trudy
Bob
I'm Alice, R_1
R_3, $f(K_{Alice-Bob}, R_1)$

Reflection Attack...

Lesson

Don't have Alice and Bob do exactly the same thing!

Reflection Attack...

Lesson

Don't have Alice and Bob do exactly the same thing!

- Use different keys to authenticate Alice from Bob
  ➠ two keys or a transformation of the shared key (e.g., $-K_{Alice-Bob}$)

Reflection Attack...

Lesson

Don't have Alice and Bob do exactly the same thing!

- Use different keys to authenticate Alice from Bob
  ➠ two keys or a transformation of the shared key (e.g., $-K_{Alice-Bob}$)
- Encode the challenges in different structures (e.g., $Bob|R$)

Reflection Attack...

Lesson

Don't have Alice and Bob do exactly the same thing!

- Use different keys to authenticate Alice from Bob
  ➠ two keys or a transformation of the shared key (e.g., $-K_{Alice-Bob}$)
- Encode the challenges in different structures (e.g., $Bob|R$)
- Make sure the initiator be the first to prove its identity
  ➠ assumption: Trudy is more likely to be the initiator

Reflection Attack...

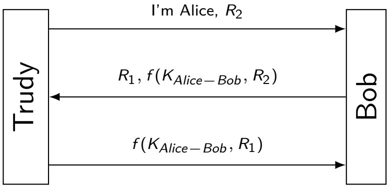
Lesson

Don't have Alice and Bob do exactly the same thing!

- Use different keys to authenticate Alice from Bob
  ➠ two keys or a transformation of the shared key (e.g., $-K_{Alice-Bob}$)
- Encode the challenges in different structures (e.g., $Bob|R$)
- Make sure the initiator be the first to prove its identity
  ➠ assumption: Trudy is more likely to be the initiator
  ➠ the five-message scheme doesn't have the problem!

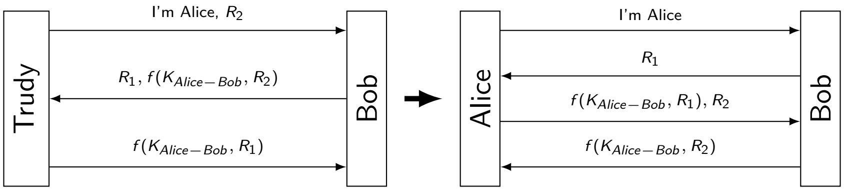
Password Guessing

- Another weakness: Trudy can mount an offline password-guessing without needing to eavesdrop



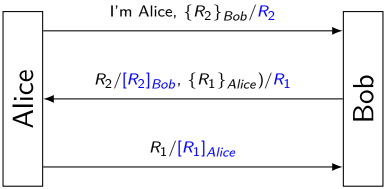
Password Guessing

- Another weakness: Trudy can mount an offline password-guessing without needing to eavesdrop
- To fix it, make sure the initiator is the first to prove its identity
  ➠ assumption: Trudy is more likely to be the initiator



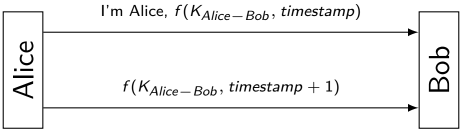
Mutual Authentication with Public Key

- Distribution of public keys is a critical issues
  - ➠ store Bob's public key encrypted with Alice's password
  - ➠ store a certificate (signed by Alice's key) for Bob's public key


Alice
Bob
I'm Alice, $\{R_2\}_{Bob}/R_2$
$R_2/([R_2]_{Bob}, \{R_1\}_{Alice})/R_1$
$R_1/[R_1]_{Alice}$

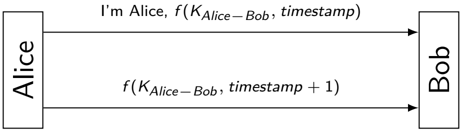
Mutual Authentication with Timestamps

- Alice and Bob's clocks should be reasonably synchronized


Alice
Bob
I'm Alice, $f(K_{Alice-Bob}, timestamp)$
$f(K_{Alice-Bob}, timestamp+1)$

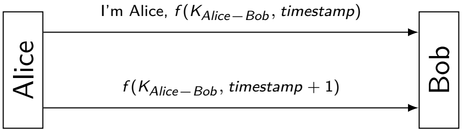
Mutual Authentication with Timestamps

- Alice and Bob's clocks should be reasonably synchronized
- Alice and Bob should do different things, how?


Alice
Bob
I'm Alice, $f(K_{Alice-Bob}, timestamp)$
$f(K_{Alice-Bob}, timestamp + 1)$

Mutual Authentication with Timestamps

- Alice and Bob's clocks should be reasonably synchronized
- Alice and Bob should do different things, how?
  - ➠ use different keys (two keys or key transformation)
  - ➠ indicate the sender in the message (e.g., `Bob|timestamp+1`)


Alice
Bob
I'm Alice, $f(K_{Alice-Bob}, timestamp)$
$f(K_{Alice-Bob}, timestamp+1)$

Data Integrity/Encryption: Session Key

- Communication after mutual authentication should be protected
  - ➠ establish a session key to avoid authentication key being overused
  - ➠ use the session key protect data integrity and secrecy

Data Integrity/Encryption: Session Key

- Communication after mutual authentication should be protected
  ➠ establish a session key to avoid authentication key being overused
  ➠ use the session key protect data integrity and secrecy
- Session key should be different for each session

Data Integrity/Encryption: Session Key

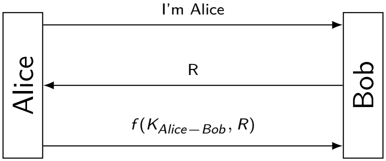
- Communication after mutual authentication should be protected
  - ➠ establish a session key to avoid authentication key being overused
  - ➠ use the session key protect data integrity and secrecy
- Session key should be different for each session
- Session key should be unpredictable

Data Integrity/Encryption: Session Key

- Communication after mutual authentication should be protected
  - ➠ establish a session key to avoid authentication key being overused
  - ➠ use the session key protect data integrity and secrecy
- Session key should be different for each session
- Session key should be unpredictable
- Breach of the authentication key should not reveal the session key
  - ➠ Diffie-Hellman key exchange

Session Key Establishment for Secret Key

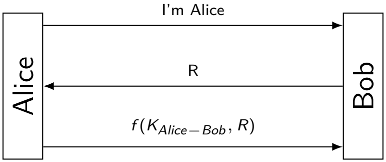
Modify $K_{Alice-Bob}$ and encrypt R with the modified key. Use the result as the session key (e.g., $(K_{Alice-Bob}+1)\{R\}$ or $(-K_{Alice-Bob})\{R\}$)



Session Key Establishment for Secret Key

Modify $K_{Alice-Bob}$ and encrypt R with the modified key. Use the result as the session key (e.g., $(K_{Alice-Bob}+1)\{R\}$ or $(-K_{Alice-Bob})\{R\}$)

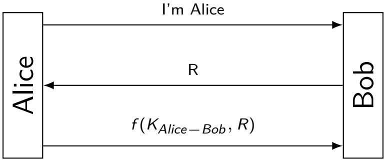
- Can we use $K_{Alice-Bob}\{R\}$ as the session key?



Session Key Establishment for Secret Key

Modify $K_{Alice-Bob}$ and encrypt R with the modified key. Use the result as the session key (e.g., $(K_{Alice-Bob}+1)\{R\}$ or $(-K_{Alice-Bob})\{R\}$)

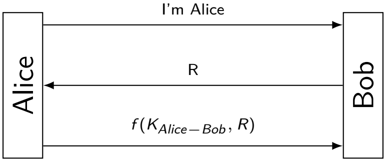
- Can we use $K_{Alice-Bob}\{R\}$ as the session key?
- Can we use $K_{Alice-Bob}\{R+1\}$ as the session key?



Session Key Establishment for Secret Key

Modify $K_{Alice-Bob}$ and encrypt R with the modified key. Use the result as the session key (e.g., $(K_{Alice-Bob}+1)\{R\}$ or $(-K_{Alice-Bob})\{R\}$)

- Can we use $K_{Alice-Bob}\{R\}$ as the session key?
- Can we use $K_{Alice-Bob}\{R+1\}$ as the session key?
  ➠ Trudy can store the data encrypted with $K_{Alice-Bob}\{R+1\}$, impersonate Bob, and challenge Alice with $R+1$



Session Key Establishment for One-way Public Key

In some cases (e.g., SSL), only one party has a public/private key ➠ it's only necessary to authenticate the server

- Alice selects R and sends $\{R\}_{Bob}$ to Bob. R is the session key

Session Key Establishment for One-way Public Key

In some cases (e.g., SSL), only one party has a public/private key ➠ it's only necessary to authenticate the server

- Alice selects R and sends $\{R\}_{Bob}$ to Bob. R is the session key ➠ Trudy can decrypt the (recorded) conversation if she gets Bob's key

Session Key Establishment for One-way Public Key

In some cases (e.g., SSL), only one party has a public/private key ➠ it's only necessary to authenticate the server

- Alice selects R and sends $\{R\}_{Bob}$ to Bob. R is the session key ➠ Trudy can decrypt the (recorded) conversation if she gets Bob's key
- Alice and Bob do a Diffie-Hellman exchange with Bob signing his T_B

Session Key Establishment for One-way Public Key

In some cases (e.g., SSL), only one party has a public/private key ➠ it's only necessary to authenticate the server

- Alice selects R and sends $\{R\}_{Bob}$ to Bob. R is the session key ➠ Trudy can decrypt the (recorded) conversation if she gets Bob's key
- Alice and Bob do a Diffie-Hellman exchange with Bob signing his T_B
- Alice is not authenticated, but entire session is with a single party

Session Key Establishment for Two-way Public Key

- Alice picks R and sends $\{R\}_{Bob}$ to Bob

Session Key Establishment for Two-way Public Key

- Alice picks R and sends $\{R\}_{Bob}$ to Bob
  ➠ Trudy can hijack (is it MITM?) the session by picking her R_T

Session Key Establishment for Two-way Public Key

- Alice picks R and sends $\{R\}_{Bob}$ to Bob
  ➠ Trudy can hijack (is it MITM?) the session by picking her R_T
- Alice picks R and sends $[\{R\}_{Bob}]_{Alice}$ to Bob

Session Key Establishment for Two-way Public Key

- Alice picks R and sends $\{R\}_{Bob}$ to Bob
  ➠ Trudy can hijack (is it MITM?) the session by picking her R_T
- Alice picks R and sends $[\{R\}_{Bob}]_{Alice}$ to Bob
  ➠ Trudy can decrypt the (recorded) session if she steals Bob's key

Session Key Establishment for Two-way Public Key

- Alice picks R and sends $\{R\}_{Bob}$ to Bob
  - ➠ Trudy can hijack (is it MITM?) the session by picking her R_T
- Alice picks R and sends $[\{R\}_{Bob}]_{Alice}$ to Bob
  - ➠ Trudy can decrypt the (recorded) session if she steals Bob's key
  - ➠ Can she do so by stealing Alice's key assuming Alice forgets R?

Session Key Establishment for Two-way Public Key...

- Alice picks R_1 and sends $\{R_1\}_{Bob}$ to Bob, Bob picks R_2 and sends $\{R_2\}_{Alice}$ to Alice, the session key is $R_1 \oplus R_2$

Session Key Establishment for Two-way Public Key...

- Alice picks R_1 and sends $\{R_1\}_{Bob}$ to Bob, Bob picks R_2 and sends $\{R_2\}_{Alice}$ to Alice, the session key is $R_1 \oplus R_2$
  ➠ Is it necessary to sign $\{R_1\}_{Bob}$ and $\{R_2\}_{Alice}$?

Session Key Establishment for Two-way Public Key...

- Alice picks R_1 and sends $\{R_1\}_{Bob}$ to Bob, Bob picks R_2 and sends $\{R_2\}_{Alice}$ to Alice, the session key is $R_1 \oplus R_2$
  ➠ Is it necessary to sign $\{R_1\}_{Bob}$ and $\{R_2\}_{Alice}$?
- Alice and Bob do a Diffie-Hellman exchange, each signs its T

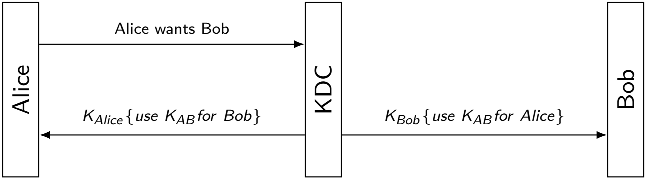
Session Key Establishment for Two-way Public Key...

- Alice picks R_1 and sends $\{R_1\}_{Bob}$ to Bob, Bob picks R_2 and sends $\{R_2\}_{Alice}$ to Alice, the session key is $R_1 \oplus R_2$
  ➠ Is it necessary to sign $\{R_1\}_{Bob}$ and $\{R_2\}_{Alice}$?
- Alice and Bob do a Diffie-Hellman exchange, each signs its T
  ➠ Trudy cannot decrypt session even she steals both private keys
  ➠ what's the assumption ?

Mediated Authentication with KDC

Some concerns:

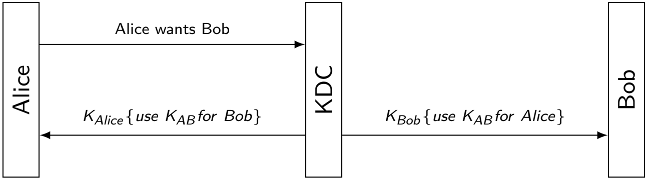
- Trudy may claim to be Alice and talk to KDC
  - ➠ Trudy may mount an offline dictionary attack if K_{AB} is structured


Alice
KDC
Bob
Alice wants Bob
$K_{Alice}\{use\ K_{AB} for\ Bob\}$
$K_{Bob}\{use\ K_{AB} for\ Alice\}$

Mediated Authentication with KDC

Some concerns:

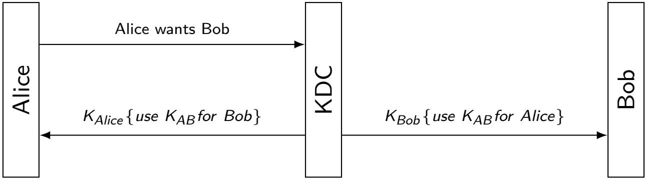
- Trudy may claim to be Alice and talk to KDC
  ➠ Trudy may mount an offline dictionary attack if K_{AB} is structured
- Messages from Alice may get to Bob before he gets K_{AB} from KDC


Alice
KDC
Bob
Alice wants Bob
$K_{Alice}\{use\ K_{AB} for\ Bob\}$
$K_{Bob}\{use\ K_{AB} for\ Alice\}$

Mediated Authentication with KDC

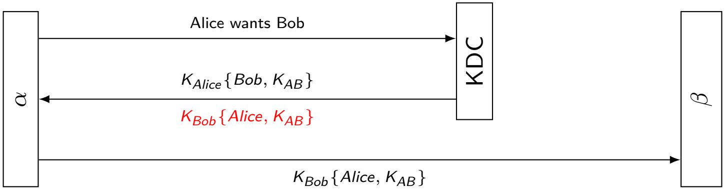
Some concerns:

- Trudy may claim to be Alice and talk to KDC
  - ➠ Trudy may mount an offline dictionary attack if K_{AB} is structured
- Messages from Alice may get to Bob before he gets K_{AB} from KDC
- It is inconvenient for KDC to connect to Bob
  - ➠ Alice is going to connect to Bob anyway


Alice
KDC
Bob
Alice wants Bob
$K_{Alice}\{use\ K_{AB} for\ Bob\}$
$K_{Bob}\{use\ K_{AB} for\ Alice\}$

Mediated Authentication with KDC...

KDC gives Alice a ticket (containing K_{AB}) to be forwarded to Bob

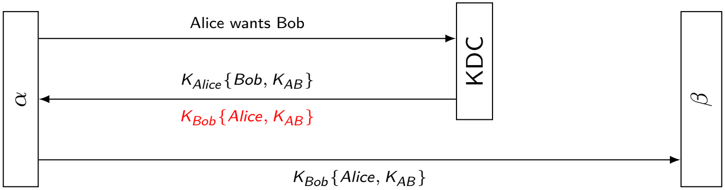


α
KDC
β
Alice wants Bob
$K_{Alice}\{Bob, K_{AB}\}$
$K_{Bob}\{Alice, K_{AB}\}$
$K_{Bob}\{Alice, K_{AB}\}$

Mediated Authentication with KDC...

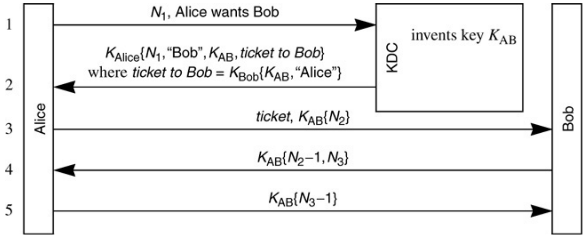
KDC gives Alice a ticket (containing K_{AB}) to be forwarded to Bob

➠ Alice and Bob must authenticate each other after this


α
Alice wants Bob
KDC
$K_{Alice}\{Bob, K_{AB}\}$
$K_{Bob}\{Alice, K_{AB}\}$
β
$K_{Bob}\{Alice, K_{AB}\}$

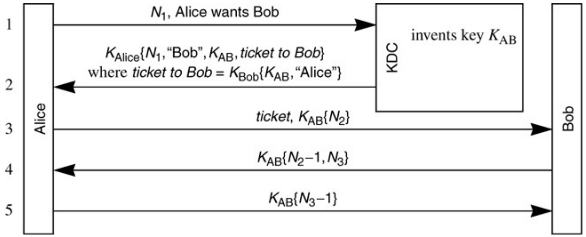
Needham-Schroeder

- Needham-Schroeder is a classic protocol for authentication with KDC
  ➠ many protocols have been modeled after it (e.g., Kerberos)


1
2
3
4
5
Alice
KDC
Bob
invents key K_{AB}
N_1, Alice wants Bob
$K_{Alice}\{N_1, \text{“Bob”}, K_{AB}, \textit{ticket to Bob}\}$
where $\textit{ticket to Bob} = K_{Bob}\{K_{AB}, \text{“Alice”}\}$
$\textit{ticket}, K_{AB}\{N_2\}$
$K_{AB}\{N_2 - 1, N_3\}$
$K_{AB}\{N_3 - 1\}$

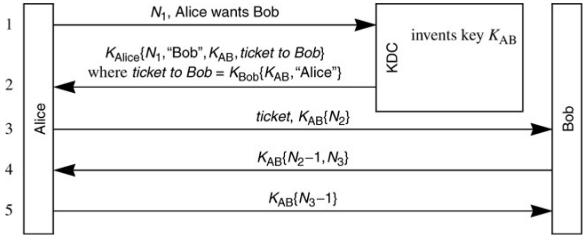
Needham-Schroeder

- Needham-Schroeder is a classic protocol for authentication with KDC
  ➠ many protocols have been modeled after it (e.g., Kerberos)
- A complete protocol: mediated authentication+mutual authentication


1
2
3
4
5
Alice
KDC
Bob
N_1, Alice wants Bob
invents key K_{AB}
$K_{Alice}\{N_1, \text{“Bob”}, K_{AB}, \textit{ticket to Bob}\}$
where $\textit{ticket to Bob} = K_{Bob}\{K_{AB}, \text{“Alice”}\}$
$\textit{ticket}, K_{AB}\{N_2\}$
$K_{AB}\{N_2-1, N_3\}$
$K_{AB}\{N_3-1\}$

Needham-Schroeder

- Needham-Schroeder is a classic protocol for authentication with KDC
  ➠ many protocols have been modeled after it (e.g., Kerberos)
- A complete protocol: mediated authentication+mutual authentication
- Nonce is a number that is used only once to prevent replay attacks


1
2
3
4
5
Alice
KDC
Bob
N_1, Alice wants Bob
invents key K_{AB}
$K_{Alice}\{N_1, \text{“Bob”}, K_{AB}, \textit{ticket to Bob}\}$
where $\textit{ticket to Bob} = K_{Bob}\{K_{AB}, \text{“Alice”}\}$
$\textit{ticket}, K_{AB}\{N_2\}$
$K_{AB}\{N_2-1, N_3\}$
$K_{AB}\{N_3-1\}$

Needham-Schroeder: A Limitation

The ticket to Bob (K_{AB}) is valid even after Alice changes her key

Needham-Schroeder: A Limitation

The ticket to Bob (K_{AB}) is valid even after Alice changes her key

- Trudy steals Alice's key, and impersonates Alice

Needham-Schroeder: A Limitation

The ticket to Bob (K_{AB}) is valid even after Alice changes her key

- Trudy steals Alice's key, and impersonates Alice
- Alice finds it and changes her key

Needham-Schroeder: A Limitation

The ticket to Bob (K_{AB}) is valid even after Alice changes her key

- Trudy steals Alice's key, and impersonates Alice
- Alice finds it and changes her key
- Trudy can still impersonate Alice to Bob because K_{AB} remains valid

Needham-Schroeder: A Limitation

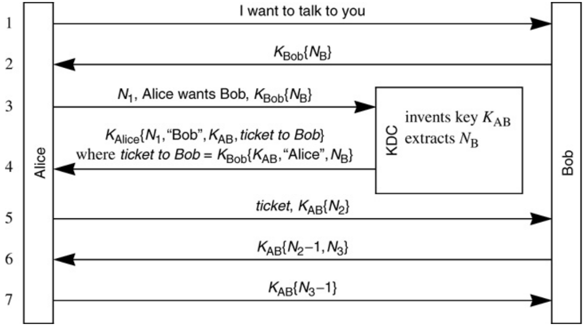
The ticket to Bob (K_{AB}) is valid even after Alice changes her key

- Trudy steals Alice's key, and impersonates Alice
- Alice finds it and changes her key
- Trudy can still impersonate Alice to Bob because K_{AB} remains valid

➠ To prevent it, make sure Alice has talked to KDC (using her active key)

Expanded Needham-Schroeder

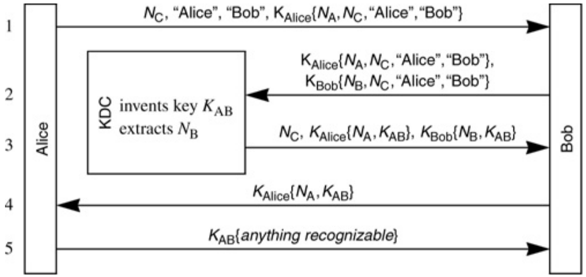
The first two messages assure Bob that Alice has talked to KDC since he generates N_B



Alice
Bob
KDC
1 I want to talk to you
2 $K_{Bob}\{N_B\}$
3 N_1, Alice wants Bob, $K_{Bob}\{N_B\}$
invents key K_{AB}
extracts N_B
4 $K_{Alice}\{N_1, \text{"Bob"}, K_{AB}, \textit{ticket to Bob}\}$
where $\textit{ticket to Bob} = K_{Bob}\{K_{AB}, \text{"Alice"}, N_B\}$
5 $\textit{ticket}, K_{AB}\{N_2\}$
6 $K_{AB}\{N_2 - 1, N_3\}$
7 $K_{AB}\{N_3 - 1\}$

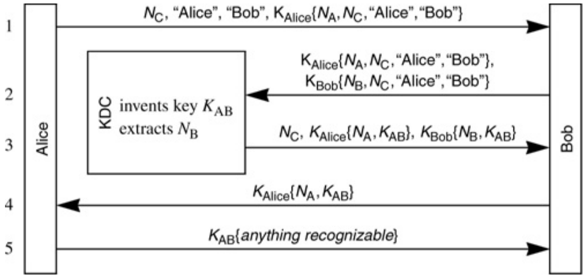
Otway-Rees Protocol

- How does KDC make sure it is really Alice and Bob?


1
N_C, "Alice", "Bob", $K_{Alice}\{N_A, N_C$, "Alice", "Bob"$\}$
2
$K_{Alice}\{N_A, N_C$, "Alice", "Bob"$\}$, $K_{Bob}\{N_B, N_C$, "Alice", "Bob"$\}$
KDC
invents key K_{AB}
extracts N_B
3
N_C, $K_{Alice}\{N_A, K_{AB}\}$, $K_{Bob}\{N_B, K_{AB}\}$
4
$K_{Alice}\{N_A, K_{AB}\}$
5
$K_{AB}\{anything\ recognizable\}$
Alice
Bob

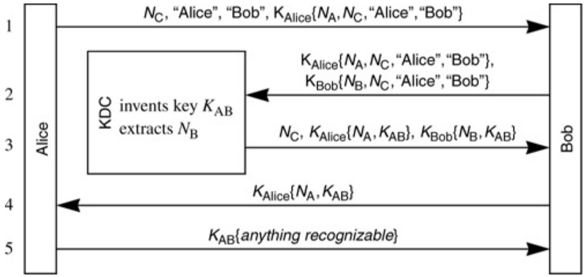
Otway-Rees Protocol

- How does KDC make sure it is really Alice and Bob?
- How does Alice make sure it is really KDC and Bob?


1
N_C, "Alice", "Bob", $K_{Alice}\{N_A, N_C$, "Alice", "Bob"$\}$
2
$K_{Alice}\{N_A, N_C$, "Alice", "Bob"$\}$,
$K_{Bob}\{N_B, N_C$, "Alice", "Bob"$\}$
KDC
invents key K_{AB}
extracts N_B
3
N_C, $K_{Alice}\{N_A, K_{AB}\}$, $K_{Bob}\{N_B, K_{AB}\}$
4
$K_{Alice}\{N_A, K_{AB}\}$
5
$K_{AB}\{anything\ recognizable\}$
Alice
Bob

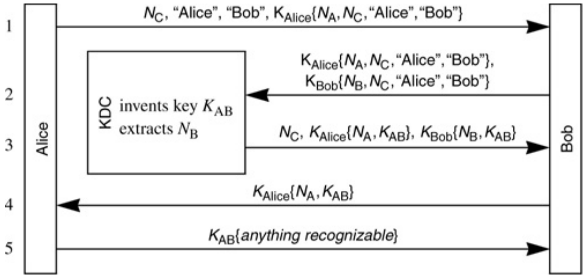
Otway-Rees Protocol

- How does KDC make sure it is really Alice and Bob?
- How does Alice make sure it is really KDC and Bob?
- How does Bob make sure it is really KDC and Alice?


1
N_C, "Alice", "Bob", $K_{Alice}\{N_A, N_C$, "Alice", "Bob"$\}$
2
$K_{Alice}\{N_A, N_C$, "Alice", "Bob"$\}$,
$K_{Bob}\{N_B, N_C$, "Alice", "Bob"$\}$
KDC
invents key K_{AB}
extracts N_B
3
N_C, $K_{Alice}\{N_A, K_{AB}\}$, $K_{Bob}\{N_B, K_{AB}\}$
4
$K_{Alice}\{N_A, K_{AB}\}$
5
$K_{AB}\{$*anything recognizable*$\}$
Alice
Bob

Otway-Rees Protocol

- How does KDC make sure it is really Alice and Bob?
- How does Alice make sure it is really KDC and Bob?
- How does Bob make sure it is really KDC and Alice?
- How does this protocol invalidate tickets?


1
N_C, "Alice", "Bob", $K_{Alice}\{N_A, N_C, \text{"Alice"}, \text{"Bob"}\}$
2
$K_{Alice}\{N_A, N_C, \text{"Alice"}, \text{"Bob"}\}$, $K_{Bob}\{N_B, N_C, \text{"Alice"}, \text{"Bob"}\}$
KDC
invents key K_{AB}
extracts N_B
3
N_C, $K_{Alice}\{N_A, K_{AB}\}$, $K_{Bob}\{N_B, K_{AB}\}$
4
$K_{Alice}\{N_A, K_{AB}\}$
5
$K_{AB}\{\textit{anything recognizable}\}$
Alice
Bob

Summary

- One-way Authentication
- Mutual Authentication
- Session Key Establishment
- Mediated Authentication
- Needham-Schroeder and Otway-Rees Protocols

- Next lecture: PKI