Entity(user) Authentication Protocols

2019. 4. 30

Contents

- Entity Authentication Protocols
  - by symmetric key
  - by public key
  - along with session establishment and perfect forward security
- Zero Knowledge Proofs

terms

- Entity authentication is the process of verifying an identity claimed by or for an entity
- This process has two steps:
  - Identification: presenting an identifier to the authentication system
  - Verification: presenting authentication information that corroborates the binding between the entity and the identifier

Authentication Information

- Something we know
  - Eg, password, PIN, answers to a question(prearranged or challenge)
- Something we possess
  - eg, crypt keys, electric keycards, smart cards
  - Often called token
- Something we are (static bio info)
  - Eg, Fingerprint, retina, face, palm, etc.
- Something we does (dynamic bio info)
  - Eg, voice pattern, handwriting, etc.
- For network-based user authentication, the mostly used information is something we know or possess.

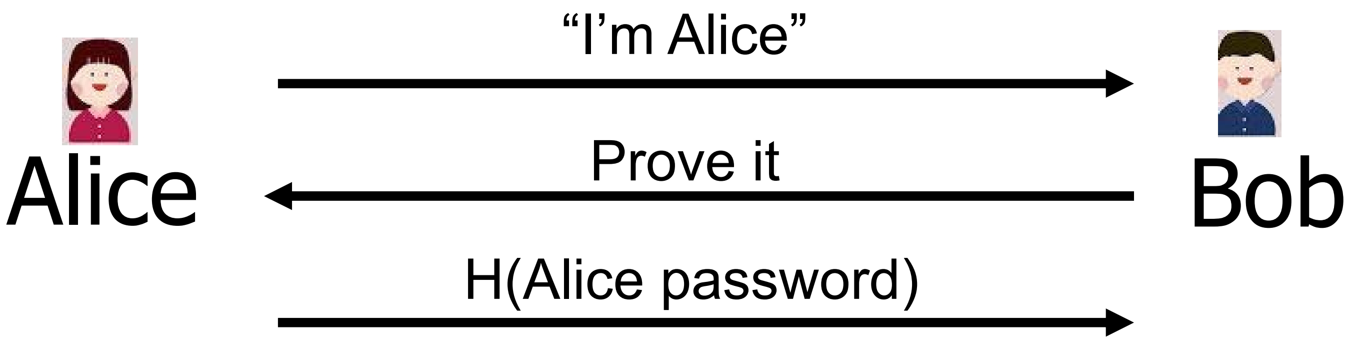
Requirements

- Alice must prove her identity to Bob (one-way authentication)
  - Alice and Bob can be humans or computers
- May also require Bob to prove he's Bob (mutual authentication)
- At the same time, may also need to establish a session key
- May have other requirements, such as
  - Use only public keys
  - Use only symmetric keys
  - Use only a hash function
  - Anonymity, plausible deniability, etc.

Entity Authentication

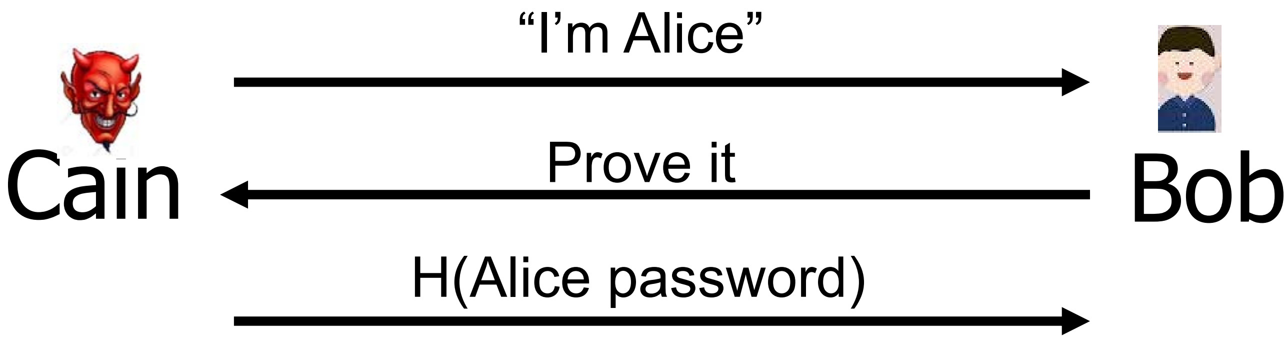
- Authentication on a stand-alone computer is relatively simple
  - "Secure path" is the primary issue
  - Main concern is an attack on authentication software
- Authentication over a network is much more complex: also called remote entity(user) authentication
  - Attacker can passively observe messages
  - Attacker can replay messages
  - Other active attacks may be possible (insert, delete, change messages)

Simple Authentication



- Simple and may be OK for standalone system
- But insecure for networked system
  - Subject to a replay attack (next 2 slides)
  - Bob must know Alice’s password

Replay Attack



- This is a **replay** attack
- How can we prevent a replay?

Challenge-Response

- Challenge-response method is commonly used to prevent replay attack.
- Suppose Bob wants to authenticate Alice (prove Alice identity)
  - Challenge sent from Bob to Alice
  - If only Alice can provide the correct response, Bob believe it's Alice.
  - Challenge chosen so that replay is not possible
- How to accomplish this?
  - Password is something only Alice should know…
  - For freshness, a "number used once" or **nonce**

Challenge-Response

- Nonce is the **challenge for** preventing replay, insures freshness
- Secret is an authentication information with which Bob prove Alice identity.
- What can we use for secrets?
  - Password is a typical secret.
  - Symmetric key, private key, and hash key are also secrets.

Using various secret for authentication

- Now our concerns are how secure the **protocol is,** not the crypto algorithm is.
- We assume that crypto algorithm is secure
- What can we use to secrets?
  - Password is a typical secret.
  - And there are many other secrets.
  - But crypto algorithm can provide much better secrets.
- Authentication by crypto algorithms
  - Symmetric key
  - private key
  - keyed hash function (HMAC)

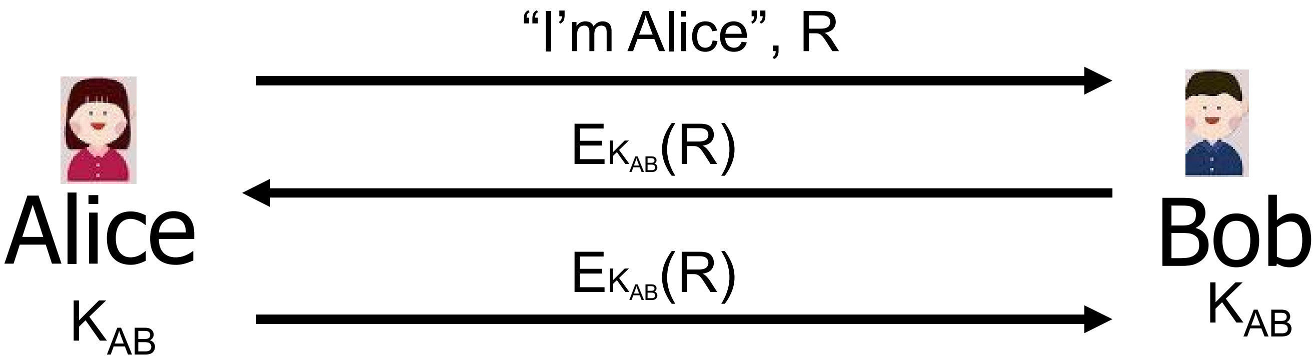
Authentication by symmetric key

- Alice and Bob share symmetric key K_{AB}
- Key K_{AB} known only to Alice and Bob
- Authenticate by proving the knowledge of a shared symmetric key
- How to accomplish this?
  - Must not reveal key
  - Must not allow replay attack

Authentication by symmetric key

- Secure method for Bob to authenticate Alice
- Alice does not authenticate Bob (one-way authentication)
- Can we achieve mutual authentication?

1st Mutual Authentication



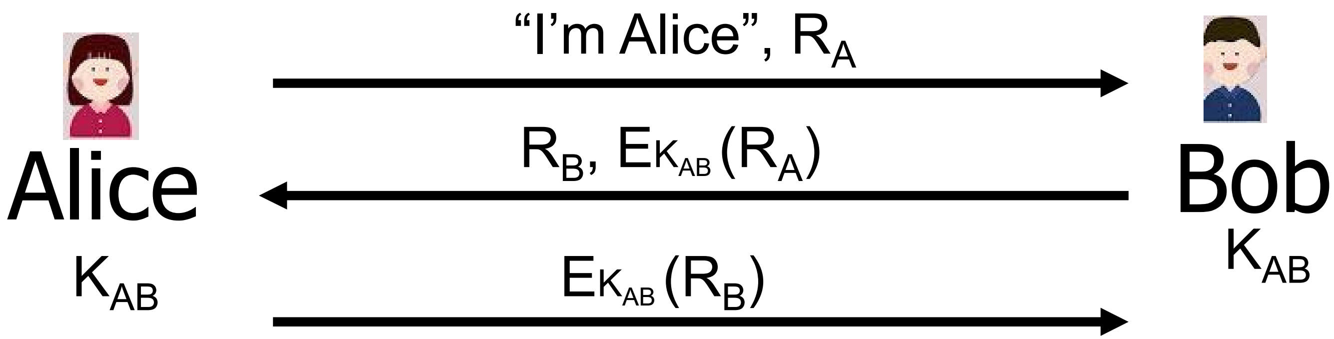
- What's wrong with this picture?
- "Alice" could be Cain (or anybody else)!

Mutual Authentication

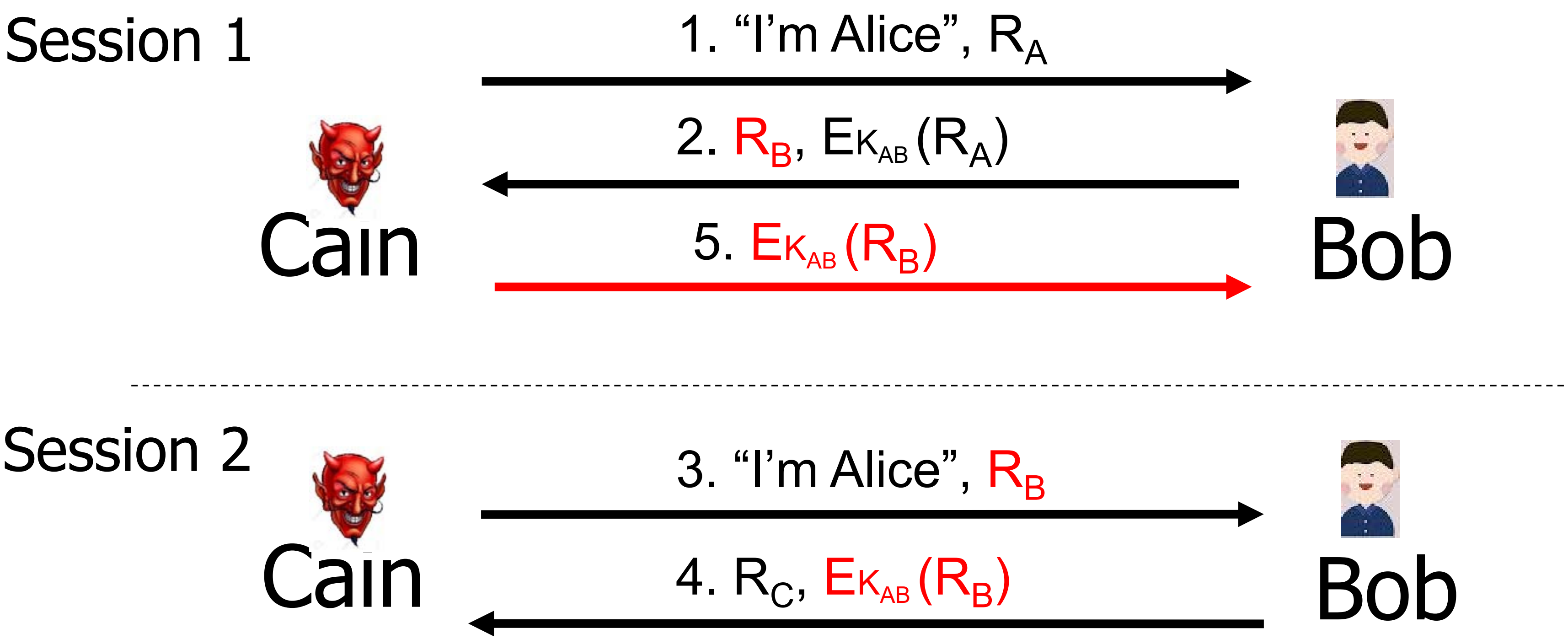
- Since we have a secure one-way authentication protocol…
- The obvious thing to do is to use the protocol twice
  - Once for Bob to authenticate Alice
  - Once for Alice to authenticate Bob
- This has to work, doesn't it?

2nd Mutual Authentication

- This provides mutual authentication
- Is it secure?

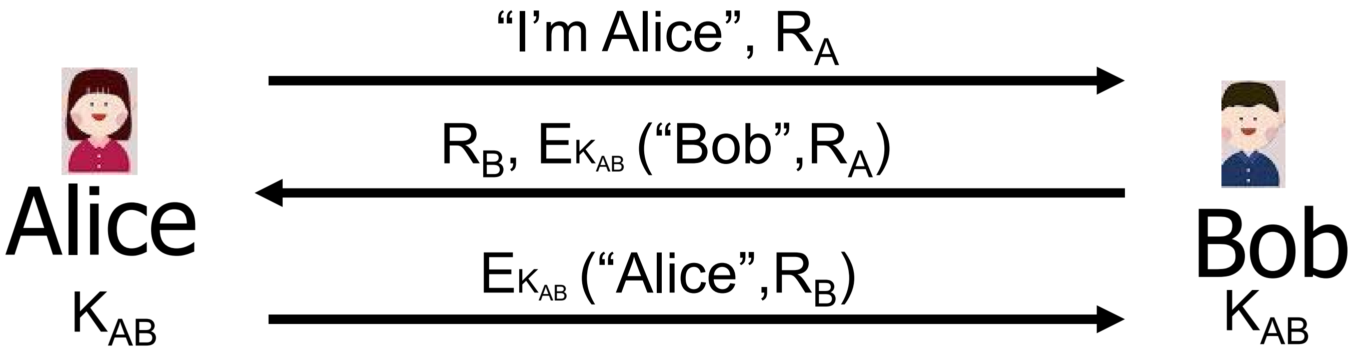


Attack



Session 1
1. "I'm Alice", R_A
2. R_B, $E_{K_{AB}}(R_A)$
5. $E_{K_{AB}}(R_B)$
Cain
Bob
Session 2
3. "I'm Alice", R_B
4. R_C, $E_{K_{AB}}(R_B)$
Cain
Bob

3rd Mutual Authentication by sym key



- Do these "insignificant" changes help?
- Remember we learned in the security protocol design! Naming is the key in this case.

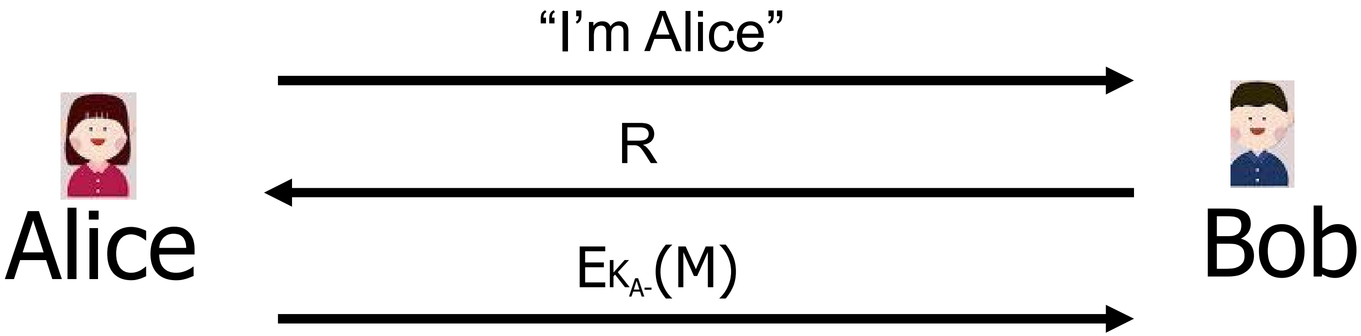
Remarks

- Our one-way authentication protocol **not** secure for mutual authentication
- The "looks-obvious" thing may not be secure
- Also, if assumptions or environment changes, protocol may not work
  - This is a common source of security failure

Authentication by Public Key

- Signing with a private key can prove it is Alice since only Alice has her private key.
- Encrypt M with Alice's public key: $E_{K_{A+}}(M)$
- Sign M with Alice's private key: $E_{K_{A-}}(M)$
- Two sequences:
  - First sign and encrypt
  - First encrypt and sign

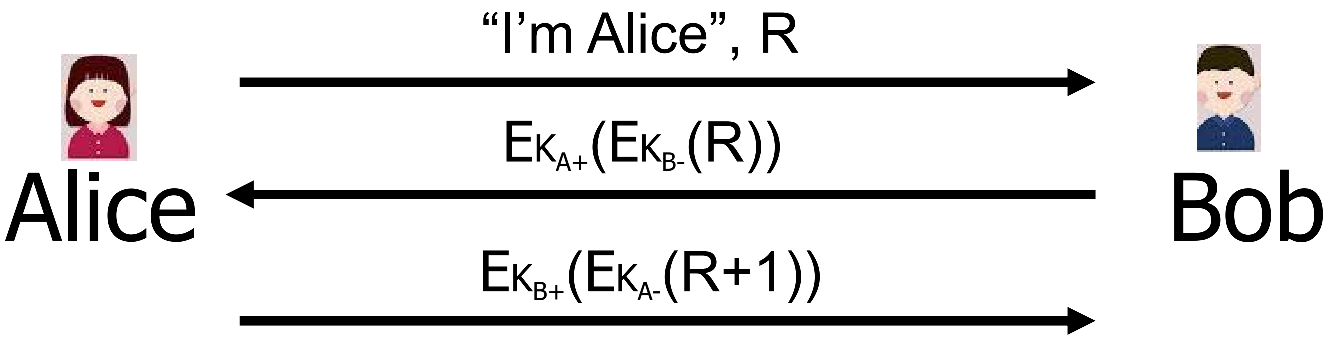
Authentication by Public Key



- Is this secure?
- Trudy can get Alice to sign anything!

Authentication by Public Key

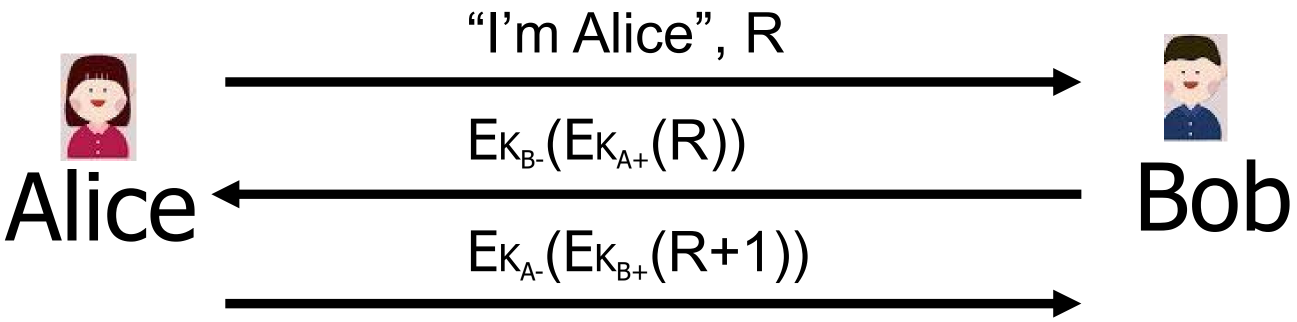
First Sign and encrypt



- Is this secure?
- Seems to be OK
- Mutual authentication!

Authentication by Public Key

First encrypt and Sign



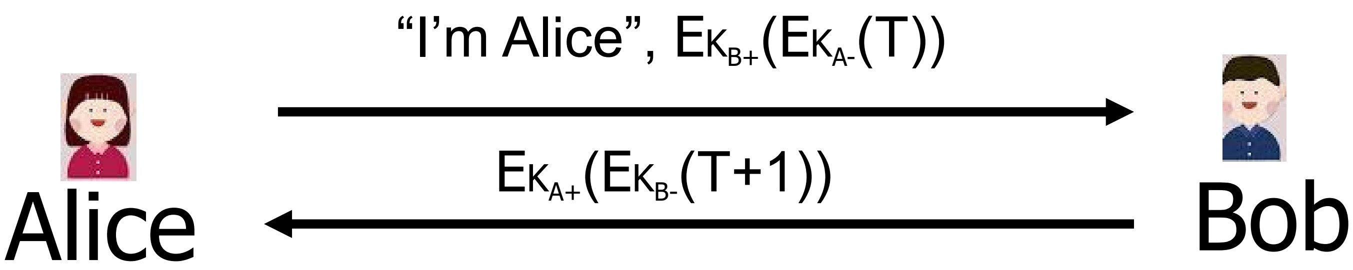
- Is this secure?
- Seems to be OK
  - Though anyone can see $E_{K_{A+}}(R)$ and $E_{K_{B+}}(R+1)$

Timestamps

- A timestamp T is the current time
- Timestamps can replace nonce for freshness.
- Timestamps reduce number of messages
  - Like a nonce that both sides know in advance
- Timestamps used in many security protocols (Kerberos, for example)
- But, use of timestamps implies that time is a security-critical parameter
- Clocks never exactly the same, so must allow for clock skew — risk of replay
- How much clock skew is enough?

Pub Key Authen with T

First Sign and encrypt



- Is this secure?
- Seems to be OK

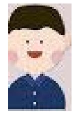
Another public key protocol

First encrypt and Sign

Alice

“I’m Alice”, $E_{K_{A-}}(E_{K_{B+}}(T))$

$E_{K_{B-}}(E_{K_{A+}}(T+1))$



Bob

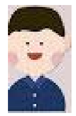
- Is this secure?
- An attacker can use Alice’s public key to find $E_{K_{B+}}(T, K)$ and then use it.

Attack

Cain

“I’m Trudy”, $E_{K_{T-}}(E_{K_{B+}}(T))$

$E_{K_{B-}}(E_{K_{T+}}(T+1))$



Bob

- Cain obtains Alice-Bob session key K
- Cain must act within clock deviation.

Public Key Authentication

- First sign and then encrypt with nonce
  - **Secure**
- First encrypt and then sign with nonce
  - **Secure**
- First sign and encrypt with timestamp
  - **Secure**
- First encrypt and then sign with timestamp
  - **Insecure**
- Protocols can be subtle!

Authentication and establishing session key

- Session key: temporary symmetric key, used for a short time period for encryption
- Usually, a session key is required in addition to authentication
  - Limit symmetric key for a particular session
  - Limit damage if one session key compromised
- Can we authenticate and establish a shared symmetric key?

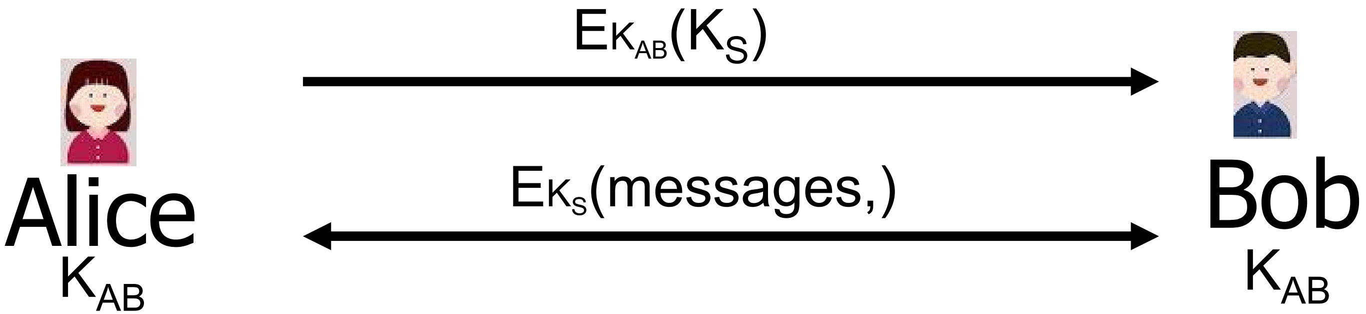
Perfect Forward Secrecy (PFS)

- In some cases, we may also require perfect forward secrecy.
- The concern…
  - Alice encrypts message with shared key K_{AB} and sends ciphertext to Bob
  - An attacker records ciphertext and later attacks Alice's (or Bob's) computer to find K_{AB}
  - Then he decrypts recorded messages
- Perfect forward secrecy (PFS):
  - An attacker cannot later decrypt recorded ciphertext even if he gets key K_{AB} or other secret(s).

Perfect Forward Secrecy and session key

- For perfect forward secrecy, Alice and Bob cannot use K_{AB} to encrypt.
- Instead, they must use a session key K_S and forget it after it's used.
- Problem: How can Alice and Bob agree on a session key K_S and insure PFS?

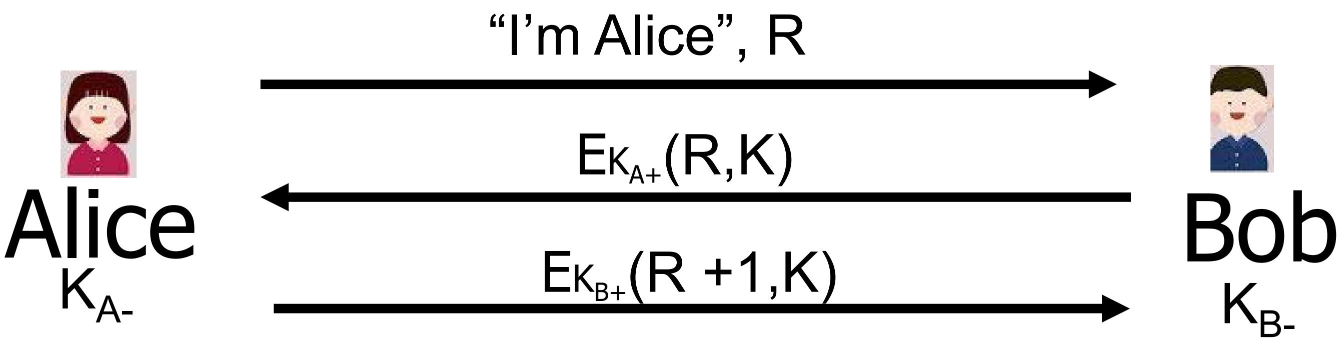
Session Key Protocol and PFS?



- An attacker could also record $E_{K_{AB}}(K_S)$
- If Trudy gets K_{AB}, she gets K_S

Authentication and Sess Key by public key

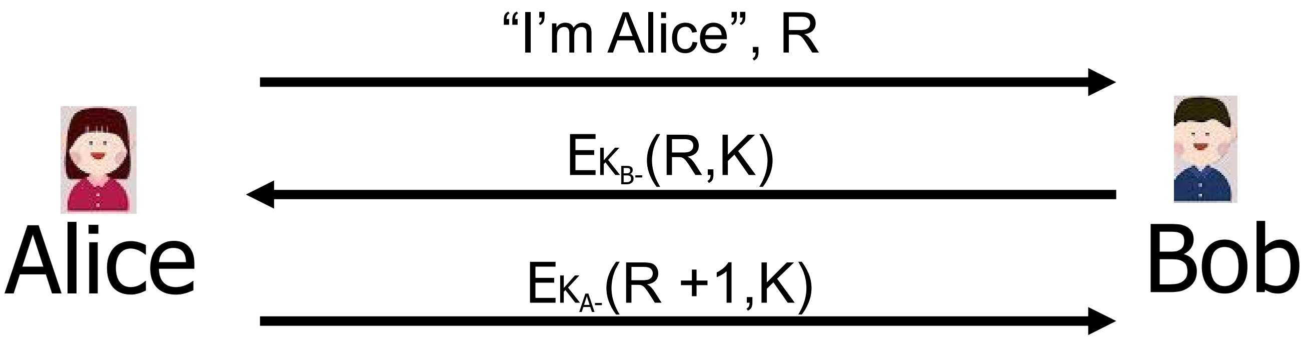
Using Public Key Encryptions by Alice and Bob



- Is this secure?
- OK for key, but no mutual authentication
  - Alice can not authenticate Bob

Authentication and Sess Key by public key

Using Signs of Alice and Bob



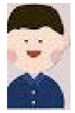
- Mutual authentication but key is not secret!

Authentication and Sess Key by public key

First Sign and encrypt

Alice

- "I'm Alice", R
- $E_{K_{A+}}(E_{K_{B-}}(R,K))$
- $E_{K_{B+}}(E_{K_{A-}}(R+1,K))$

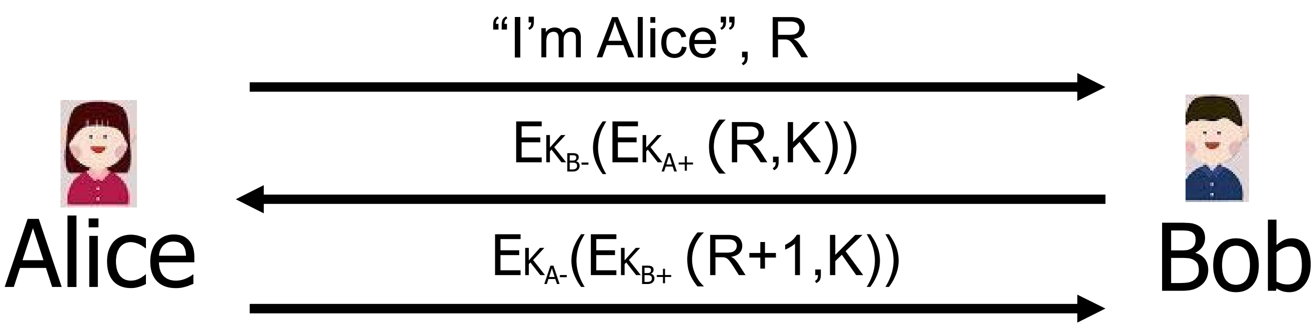


Bob

- Is this secure?
- Seems to be OK
- Mutual authentication and session key!

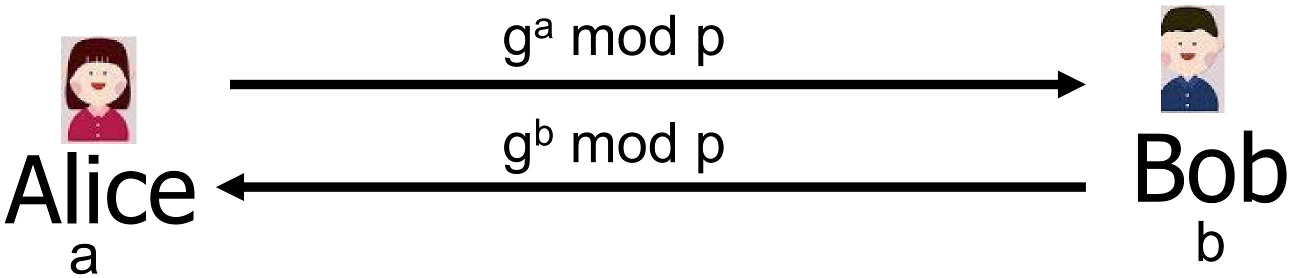
Authentication and Sess Key by public key

First encrypt and Sign



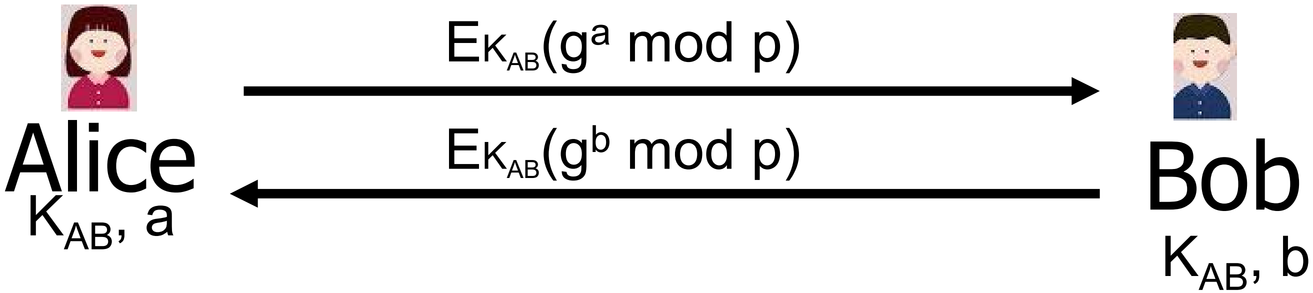
Diffie-Hellman: Perfect Forward Secrecy

- Recall Diffie-Hellman: public g and p



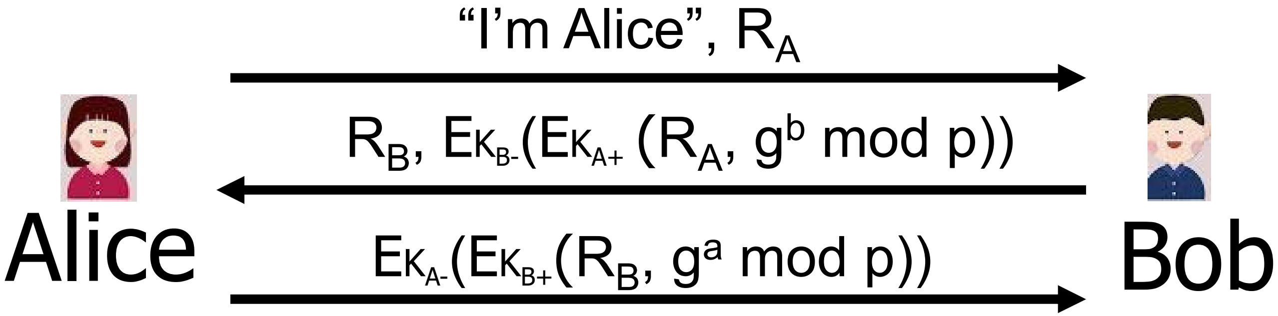
- But Diffie-Hellman is subject to MiM
- How to get PFS and prevent MiM?

Diffie-Hellman: Perfect Forward Secrecy



- Session key $K_S = f(g^{ab} \bmod p)$
- Alice forgets a, Bob forgets b
- Ephemeral Diffie-Hellman
- Not even Alice and Bob can later recover K_S
- Other ways to do PFS?

D-H: Mutual Authen, Sess Key & PFS



- Session key(of input for session key) is $K = g^{ab} \bmod p$
- Alice forgets a and Bob forgets b
- If an attacker later gets Bob’s and Alice’s secrets, she cannot recover session key K

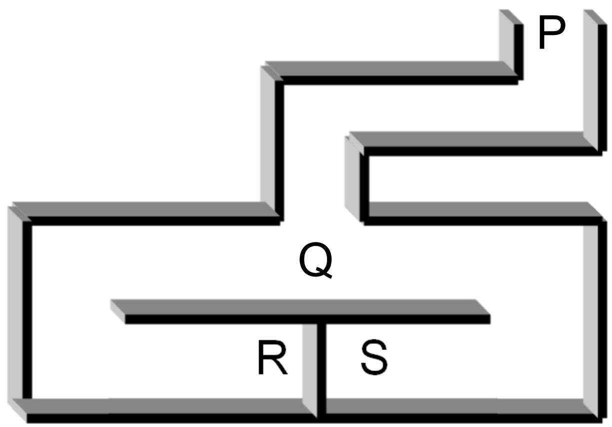
Zero Knowledge Proofs

Zero Knowledge Proof (ZKP)

- Alice wants to prove that she knows a secret without revealing **any** info about it
- Bob must verify that Alice knows secret
  - Even though he gains no info about the secret
- Process is probabilistic
  - Bob can verify that Alice knows the secret to an arbitrarily high probability
- An "interactive proof system"

Bob’s Cave

- Alice claims to know secret phrase to open path between R and S (“open sasparilla”)
- Can she convince Bob that she knows the secret without revealing phrase?

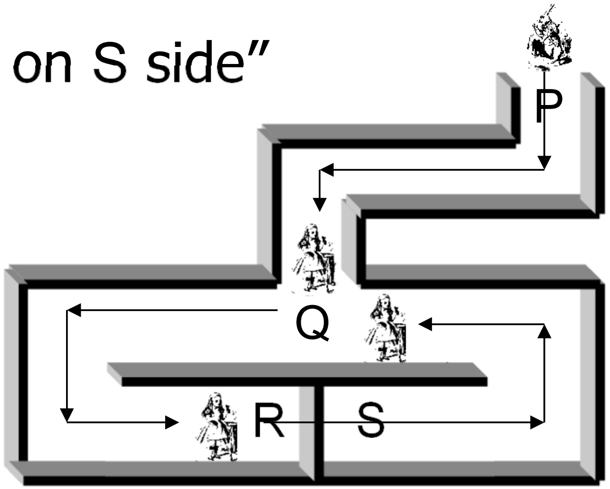


Bob's Cave

- Bob: "Alice come out on S side"

- Alice (quietly): "Open sasparilla"

- Apse Alice does not know secret

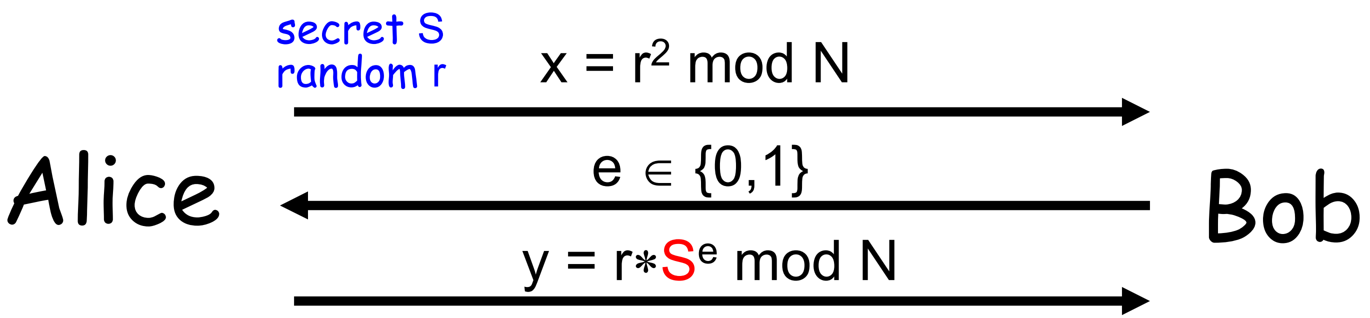


  - Without knowing secret, Alice could come out from the correct side with probability ½
  - If Bob repeats this n times, then Alice can only fool Bob with probability $1/2^n$

Fiat-Shamir Protocol

- Cave-based protocols are inconvenient
  - Can we achieve same effect without a cave?
- It is known that finding square roots modulo N is difficult (like factoring)
- Suppose N = pq, where p and q prime
- Alice has a **secret** S
- N and v = S^2 mod N are public, S is secret
- Alice must convince Bob that she knows S without revealing any information about S

Fiat-Shamir



- **Public:** Modulus N and $v = S^2 \bmod N$
- Alice selects random r
- Bob chooses $e \in \{0,1\}$
- Bob verifies that $y^2 = r^2 * S^{2e} = r^2 * (S^2)^e = x * v^e \bmod N$

Fiat-Shamir: e = 1

Secret S
Random r

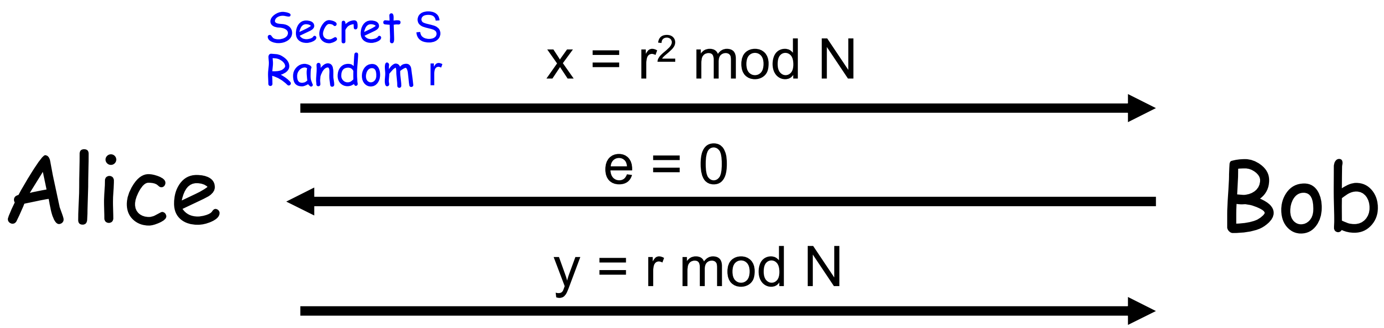
Alice $\xrightarrow{x = r^2 \bmod N}$ Bob

Alice $\xleftarrow{e = 1}$ Bob

Alice $\xrightarrow{y = r * S \bmod N}$ Bob

- **Public:** Modulus N and $v = S^2 \bmod N$
- Alice selects random r
- Suppose Bob chooses e =1
- Bob must verify that $y^2 = x * v \bmod N$
- Alice must know S in this case

Fiat-Shamir: e = 0



- **Public:** Modulus N and $v = S^2 \bmod N$
- Alice selects random r
- Suppose Bob chooses e = 0
- Bob must verify that $y^2 = x \bmod N$
- Alice does **not** need to know S in this case!

Fiat-Shamir

- **Public:** modulus N and $v = S^2 \bmod N$
- **Secret:** Alice knows S
- Alice selects random r and **commits** to r by sending $x = r^2 \bmod N$ to Bob
- Bob sends **challenge** $e \in \{0,1\}$ to Alice
- Alice **responds** with $y = r * S^e \bmod N$
- Bob checks that $y^2 = x * v^e \bmod N$
- **Does this prove response is from Alice?**

Does Fiat-Shamir Work?

- The math works since
  - Public: $v = S^2$
  - Alice to Bob: $x = r^2$ and $y = r*S^e$
  - Bob verifies $y^2 = x*v^e \bmod N$
- Can Trudy convince Bob she is Alice?
  - If Trudy expects $e = 0$, she can send $x = r^2$ in msg 1 and $y = r$ in msg 3 (i.e., follow protocol)
  - If Trudy expects Bob to send $e = 1$, she can send $x = r^2*v^{-1}$ in msg 1 and $y = r$ in msg 3
- If Bob chooses $e \in \{0,1\}$ at random, Trudy can fool Bob with probability 1/2

Fiat-Shamir Facts

- Trudy can fool Bob with prob 1/2, but…
- After n iterations, the probability that Trudy can fool Bob is only $1/2^n$
- Just like Bob's cave!
- Bob's $e \in \{0,1\}$ must be unpredictable
- Alice must use new r each iteration or else
  - If e = 0, Alice sends r in message 3
  - If e = 1, Alice sends $r*S$ in message 3
  - Anyone can find S given **both** r and $r*S$

Fiat-Shamir Zero Knowledge?

- Zero knowledge means that Bob learns **nothing** about the secret S
  - **Public:** $v = S^2 \bmod N$
  - Bob sees $r^2 \bmod N$ in message 1
  - Bob sees $r*S \bmod N$ in message 3 (if $e = 1$)
  - If Bob can find r from $r^2 \bmod N$, he gets S
  - But that requires modular square root
  - If Bob can find modular square roots, he can get S from **public** v
- The protocol does not "help" Bob to find S

ZKP in the Real World

- Public keys identify users
  - No anonymity if public keys transmitted
- ZKP offers a way to authenticate without revealing identities
- ZKP supported in Microsoft's Next Generation Secure Computing Base (NGSCB)
  - ZKP is used to authenticate software "without revealing machine identifying data"
  - ZKP is **not** just fun and games for mathematicians!