The Magic behind Remote-control Service of Firefox OS TV

J-PAKE over TLS

Chun-Min Chang, 2016/5/30

Remote-control

What we are trying to do?

Turn Firefox Android into a TV remote-controller

Demo

https://ody.sh/vtsKB0OGVv

J-PAKE over TLS

mozilla

When you see this topic, you must think...

- what is J-PAKE ?
- is it a kind of PAKE ?
- and what is PAKE ?
- what's the difference between J-PAKE and the others ?
- hmm....TLS is stranger again for me...

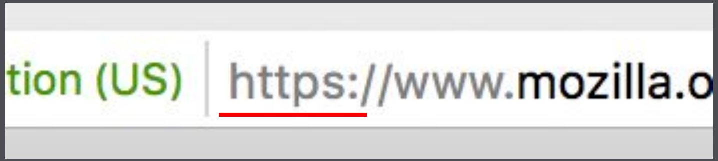
So, here is the outline

- TLS
- PAKE
    - Security Requirements
    - General Two-stage Framework
    - Diffie-Hellman Key Exchange
    - DH-EKE
    - SPEKE
- J-PAKE
    - Intro
    - Protocol
    - Zero-Knowledge Proof
- J-PAKE over TLS
- Discussion

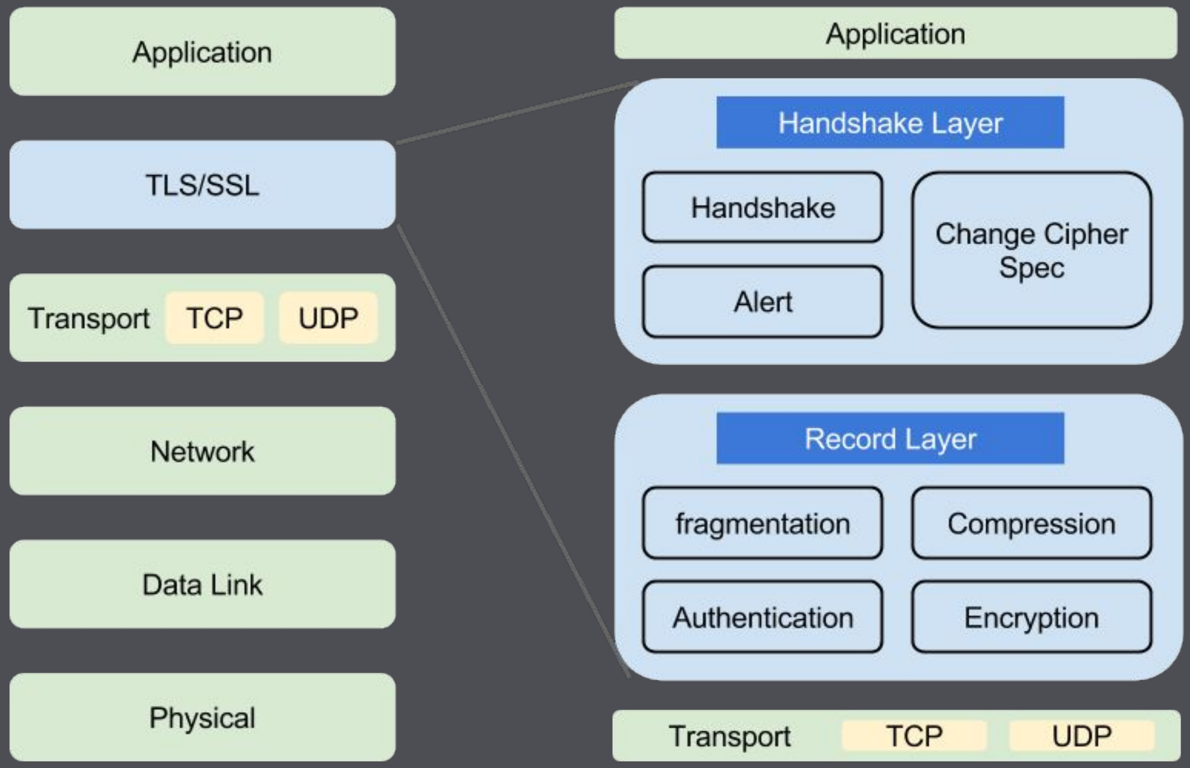
TLS

Intro of TLS

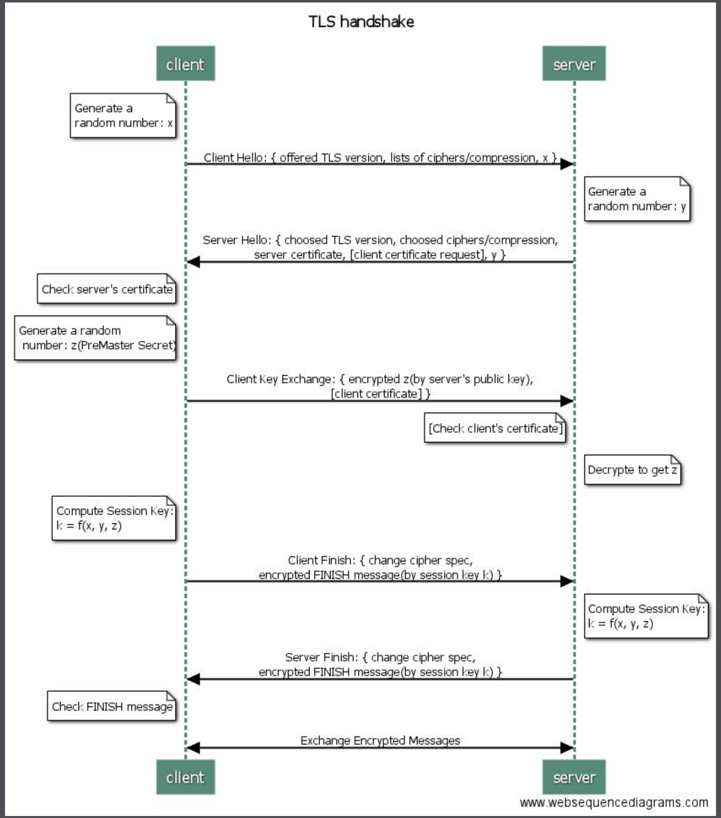
- Its predecessor is SSL, invented by NetScape
- https = http over TLS
- Make sure the channel is
  - Confidential
  - Authenticated
- Needs a trusted third party
  - Public key infrastructure(PKI)
- Symmetric encryption



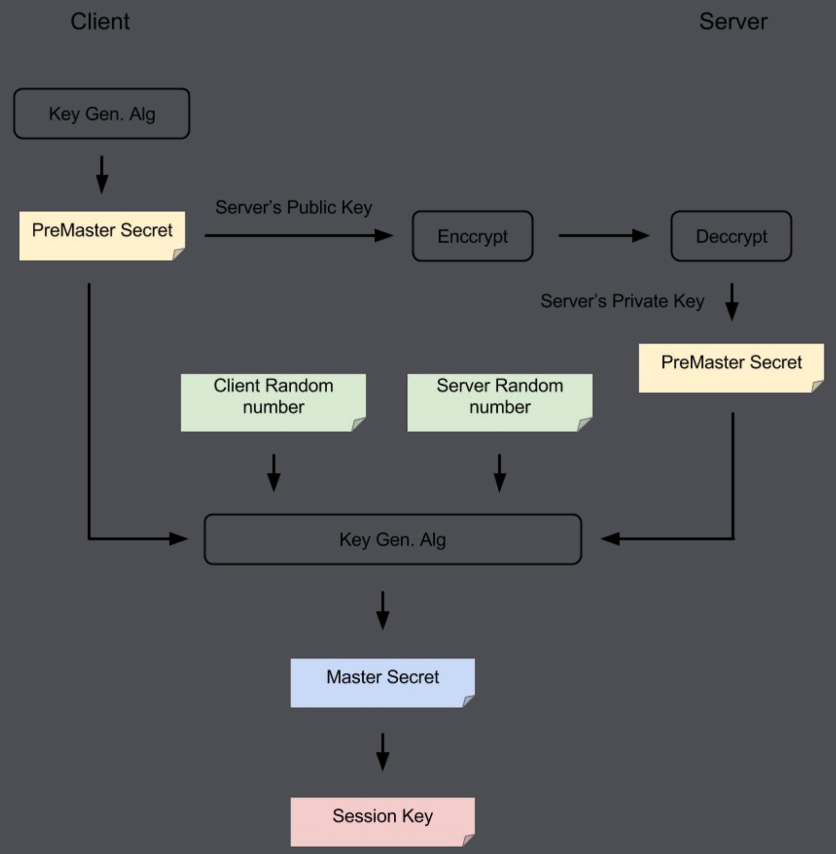
TLS layer in OSI model



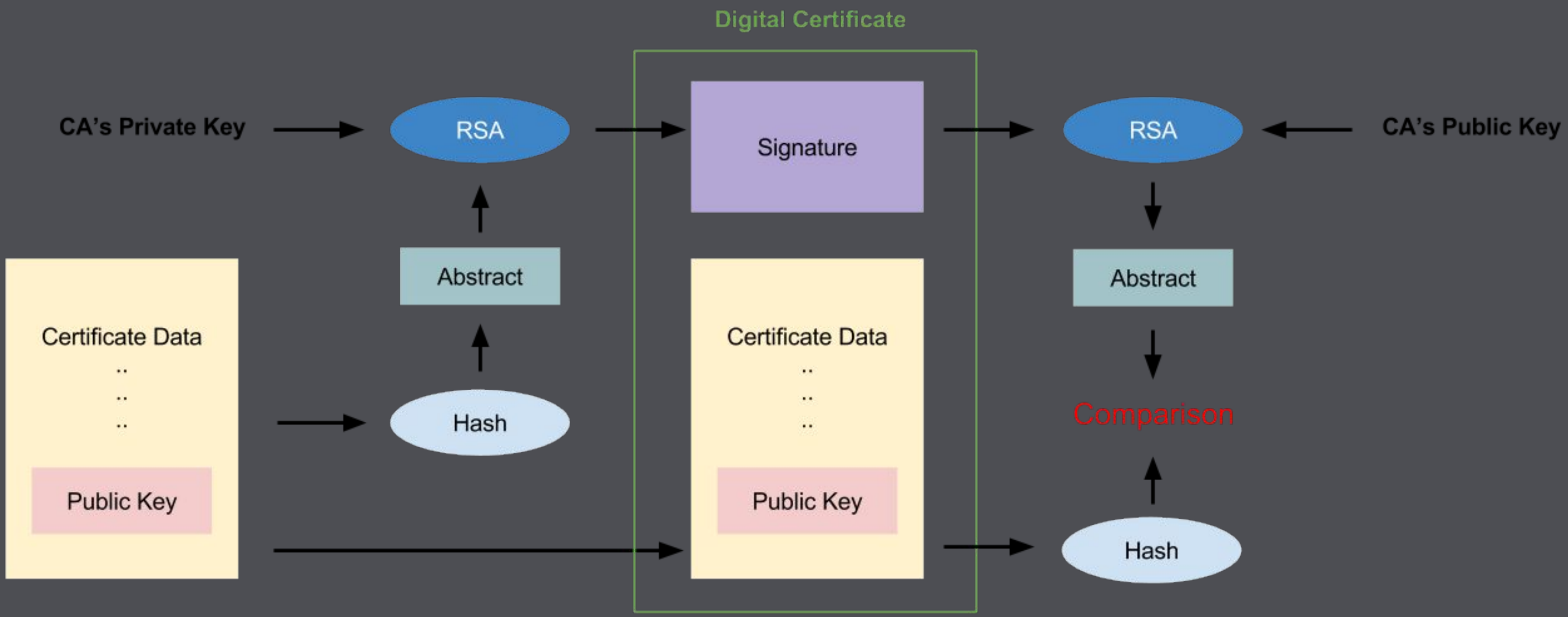
TLS handshake



TLS session key



TLS certificate authentication



why don’t we just use TLS?

- There is no PKI in local network
- TLS can still establish a confidential channel without authentication
- So, we need to use other alternative to authenticate
  - PAKE can be used as an authentication method
- PAKE over TLS
  - Secure Modular Password Authentication for the Web Using Channel Bindings

PAKE

Intro of PAKE

mozilla

- Multiple parties can establish a **shared** cryptographic keys based on their same knowledge of a password by messages exchange via an insecure channel
- The unauthorized party who doesn't possess the password has no way to get the password
- It can use weak **human-memorable** passwords to generate a **high-entropy** session key

Intro of PAKE

mozilla

- Applications
  - **Mutual authentication**
  - Alternative for computationally expensive authentication
- Common PAKE
  - EKE: Encrypted Key Exchange
  - SPEKE: Simple Password Exponential Key Exchange
  - J-PAKE: Password Authenticated Key Exchange by Juggling

Security Requirements

mozilla

- Off-line dictionary attack resistance
  It doesn't leak any info that allows attackers to perform offline-exhaustive search to find the password
- On-line dictionary attack resistance
  An active attacker can only test one password per protocol
- Forward secrecy
  The session keys still keep secure even the password is later leaked out
- Known-session security
  If one session is compromised, other established session won't be affected

General Two-stage Framework

- Key establishment
  - Negotiate a session key for their communication
  - Common method is **Diffie–Hellman** key exchange
- Key Confirmation
  - Authenticate each other

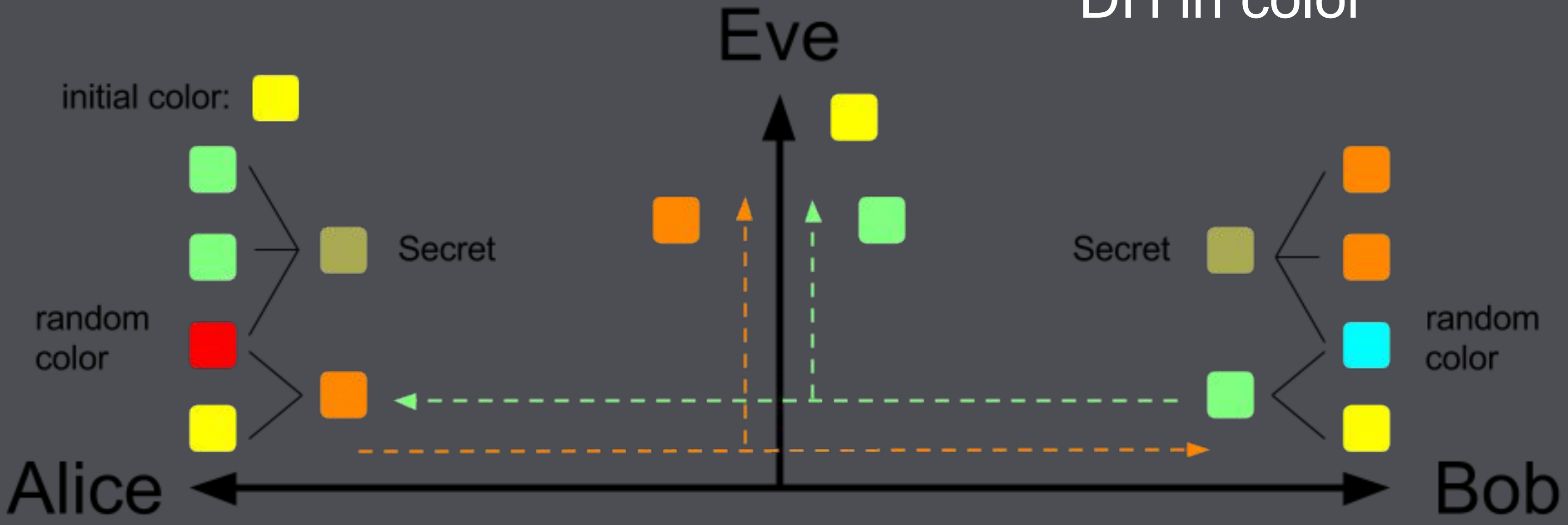
Diffie–Hellman Key Exchange

Eve is a passive attacker

Diffie–Hellman Key Exchange

mozilla

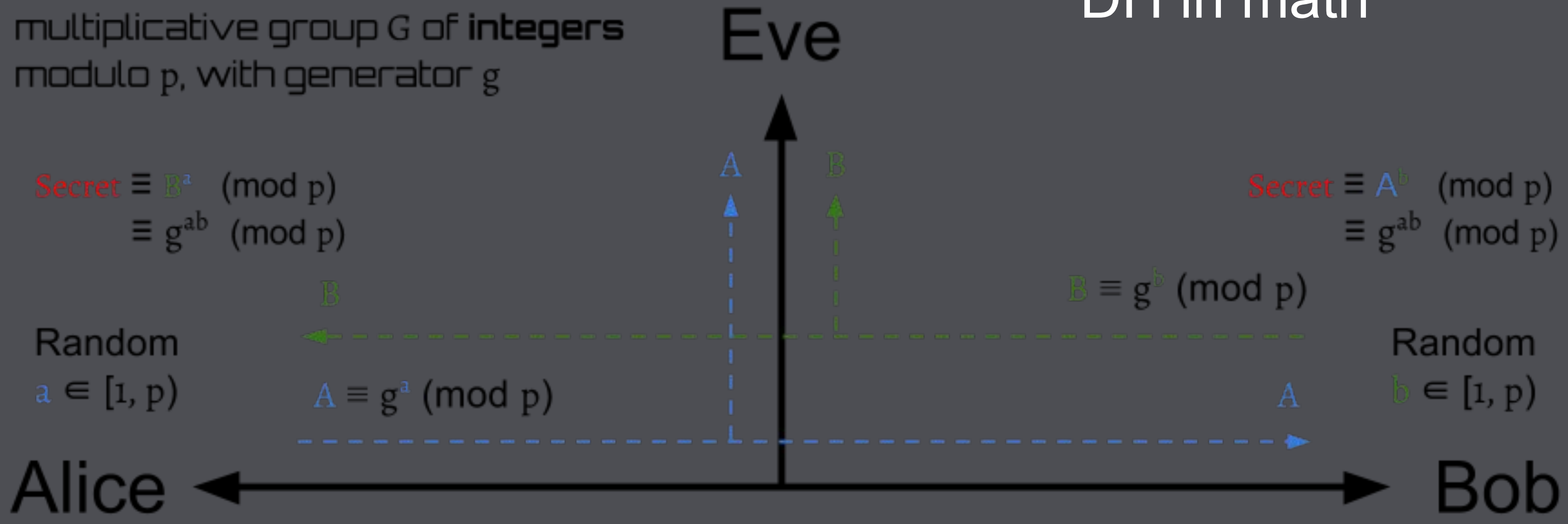
DH in color



Try yourself: http://goo.gl/I52ecS

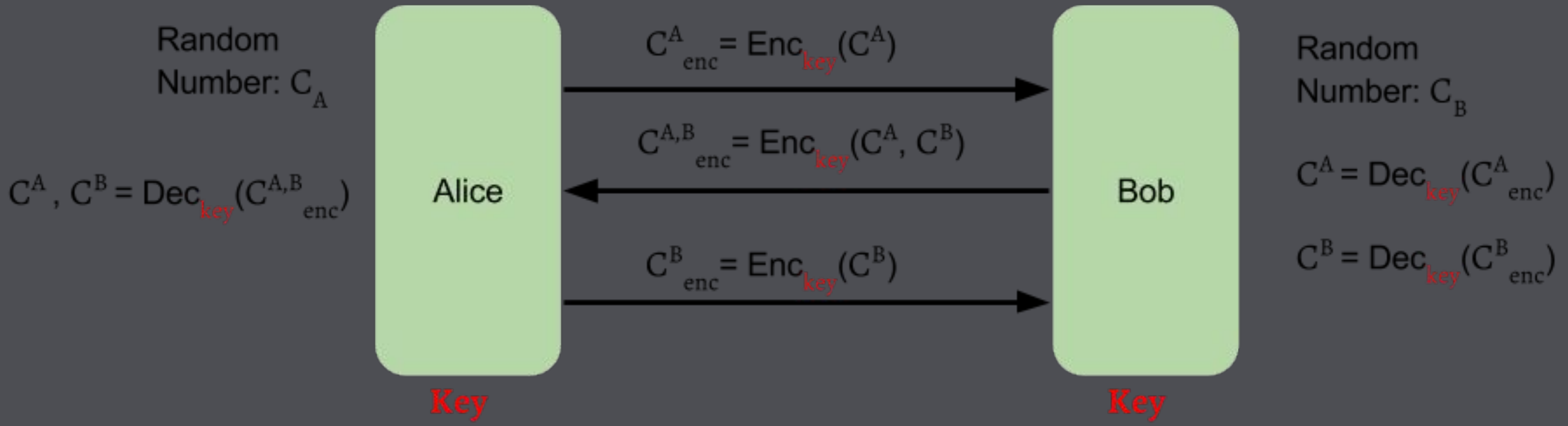
Diffie–Hellman Key Exchange

DH in math



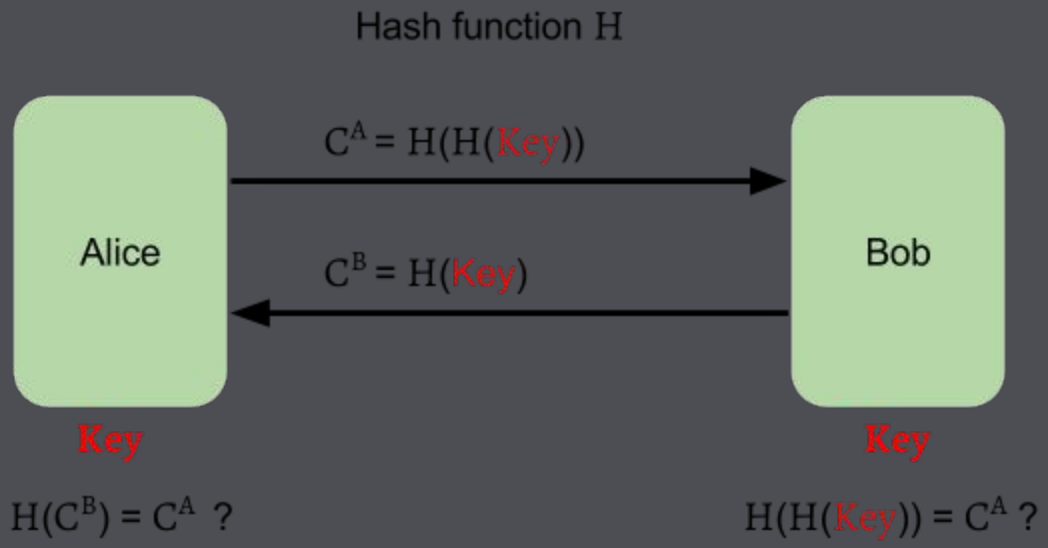
Key Confirmation

Challenge–Response Authentication



Key Confirmation

Hash–Key Authentication



Encrypted Key Exchange

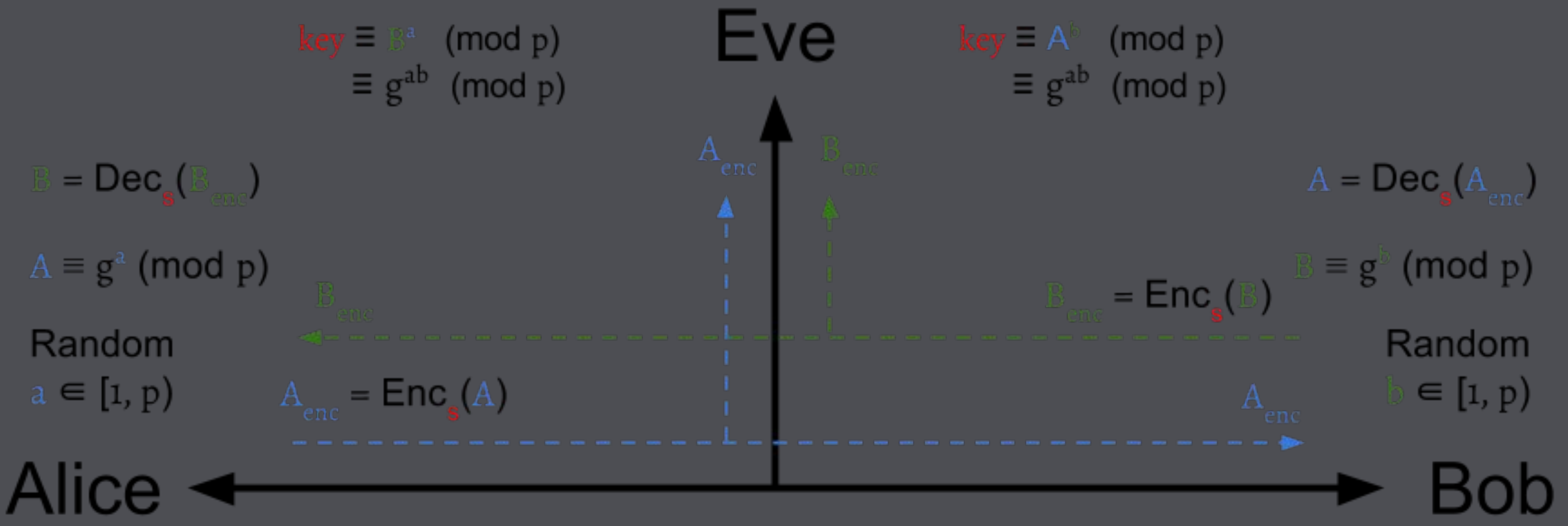
mozilla

- Key establishment by DH
- But use password to encrypt/decrypt the public key instead of directly sending it

Encrypted Key Exchange

- multiplicative group G of **integers** modulo p, with generator g
- Shared password: pwd, Shared secret: $s = \mathrm{Hash}(pwd)$
- Encryption with key k: $\mathrm{Enc}_k()$, Decryption with key k: $\mathrm{Dec}_k()$

simplified version



Encrypted Key Exchange

mozilla

- Drawbacks
  - It needs a very large exponent
  - It needs to choose modulo p carefully
    - If $p = 263 = (00000001\ 00000111)_2$, then group element must be in [1, 262]. We can guess the first seven bits in the first byte is all 0

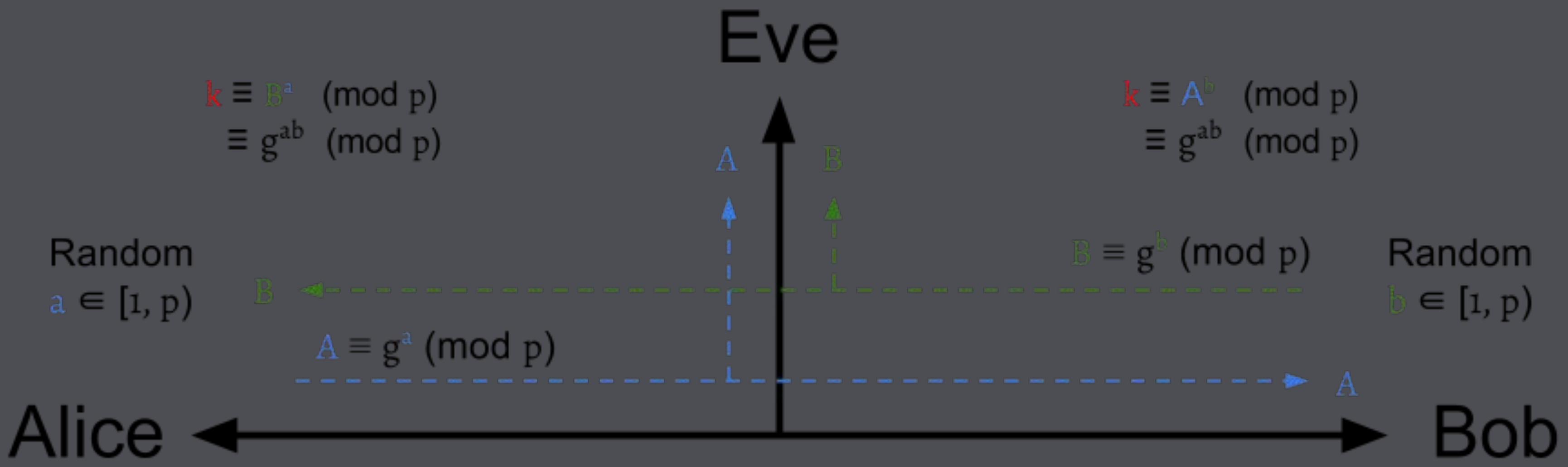
Simple Password Exponential Key Exchange

- Key establishment by DH
- But the group generator is derived by password instead
- The prime p must be a safe prime, $p = 2q + 1$, where q is also a prime.

Simple Password Exponential Key Exchange

simplified version

- **Safe prime** p, Shared password: pwd
- $g \equiv pwd^2 \pmod{p}$

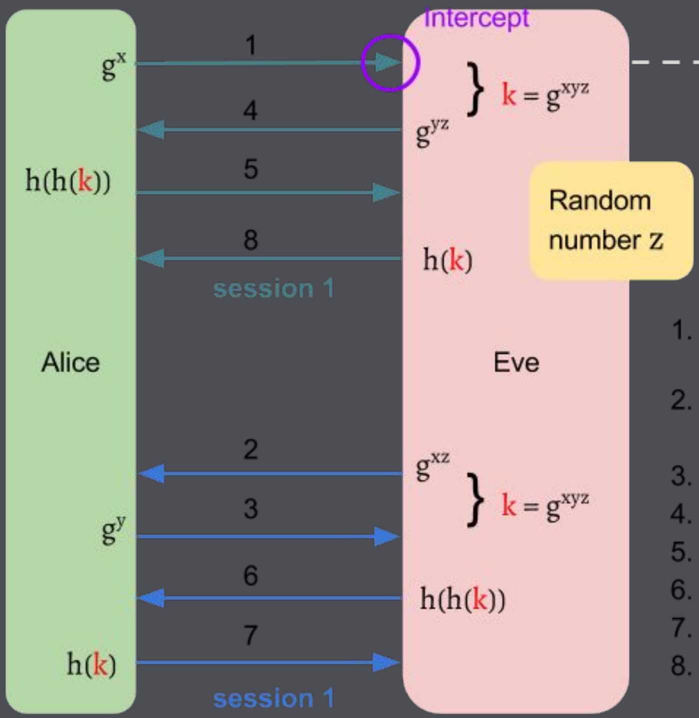


Simple Password Exponential Key Exchange

- Drawbacks
  - An attacker may test multiple password per protocol
    - $g \equiv (pwd)^2 \pmod{p}$
    - $(q+k+1)^2 \equiv (q-k)^2 \pmod{p = 2q + 1}$
    - suppose $q = 11$, then
      - $k = 0$: $11^2 \equiv 12^2 \equiv 6$
      - g is not 6, then pwd must not be 11 or 12
      - $k = 1$: $10^2 \equiv 13^2 \equiv 8$
      - if g is not 8, then pwd must not be 10 or 13

Simple Password Exponential Key Exchange

Impersonation Attack



1. Eve intercepts Alice's g^x for Bob to initiate session 1
2. Eve picks a random z then pretends Bob to send $(g^x)^z = g^{xz}$ to initiate session 2
3. Alice replies g^y on session 2
4. Eve replies $(g^y)^z = g^{yz}$ on session 1
5. Alice sends $h(h(k))$ on session 1
6. Eve sends $h(h(k))$ on session 2
7. Alice replies $h(k)$ on session 1
8. Eve replies $h(k)$ on session 2

J-PAKE

Intro of J-PAKE

mozilla

- Why we need J-PAKE?
  - EKE needs large exponents and may leak partial information about password
  - SPEKE allows an attacker tests multiple password in one protocol execution
  - EKE and SPEKE are **patented**
- Applications
  - Thread (IoT network protocol)
  - OpenSSH and OpenSSL
  - Firefox Sync
  - Palemoon sync(forked from Firefox)
- Zero-Knowledge Proof
  - Provides a valid knowledge proof of a discrete logarithm without revealing it

J-PAKE

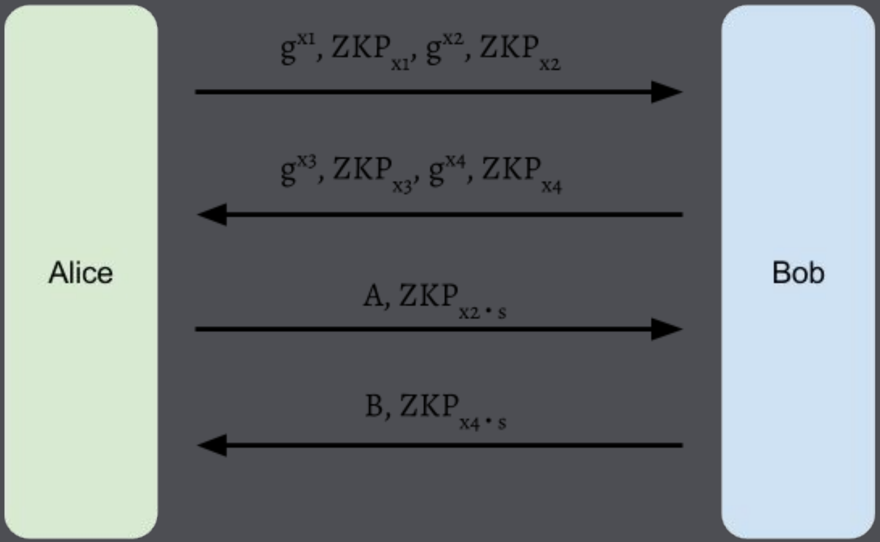
- Group G with generator g of prime order q
- Shared secret: s
- ZKP_n : To prove knowledge of $N = g^n$, sends $\{ ID, V = g^v, r = v - n \cdot h \}$, where ID is user identifier, $v \in [1, q - 1]$, and $h = Hash(g, V, N, ID)$

random:
$x_1 \in [0, q-1]$,
$x_2 \in [1, q-1]$

check: $g^{x4} \neq 1$

$A = (g^{x1} \cdot g^{x3} \cdot g^{x4})^{x2 \cdot s}$
$= g^{(x1 + x3 + x4) \cdot x2 \cdot s}$

$K = (B / (g^{x4})^{x2 \cdot s})^{x2}$
$= (g^{(x1 + x3) \cdot x4 \cdot s})^{x2}$
$= g^{(x1 + x3) \cdot x2 \cdot x4 \cdot s}$



random:
$x_3 \in [0, q-1]$,
$x_4 \in [1, q-1]$

check: $g^{x2} \neq 1$

$B = (g^{x1} \cdot g^{x2} \cdot g^{x3})^{x4 \cdot s}$
$= g^{(x1 + x2 + x3) \cdot x4 \cdot s}$

$K = (A / (g^{x2})^{x4 \cdot s})^{x4}$
$= (g^{(x1 + x3) \cdot x2 \cdot s})^{x4}$
$= g^{(x1 + x3) \cdot x2 \cdot x4 \cdot s}$

ZKP by Schnorr signature

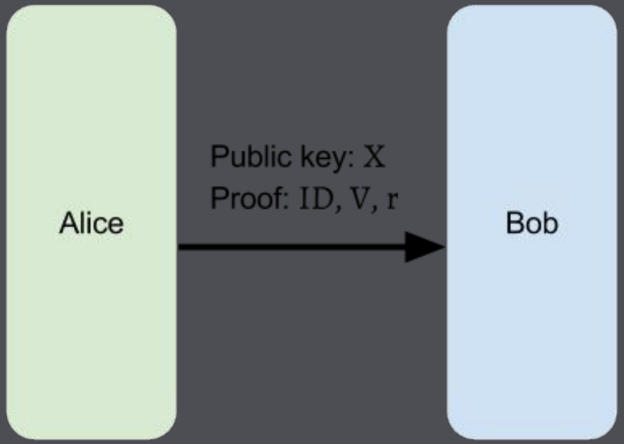
- Group G of prime order q with generator g (Thus $g^q \equiv 1$)
- Secure hash function H

Key Pair Generating

1. Pick **private key**:
   random number x
2. Get **public key**: $X = g^x$

Proof of exponent x

1. Pick random number v
2. Compute $V = g^v$
3. $h = H(g, V, X, ID)$
4. $r = v - x \cdot h$



Verifying Proof

1. $h' = H(g, V, X, ID)$
2. $V' = g^r \cdot X^{h'}$
   $= g^{v - xh} \cdot (g^x)^{h'}$
   $= g^{v + x(h' - h)}$
3. Check $V' = V$

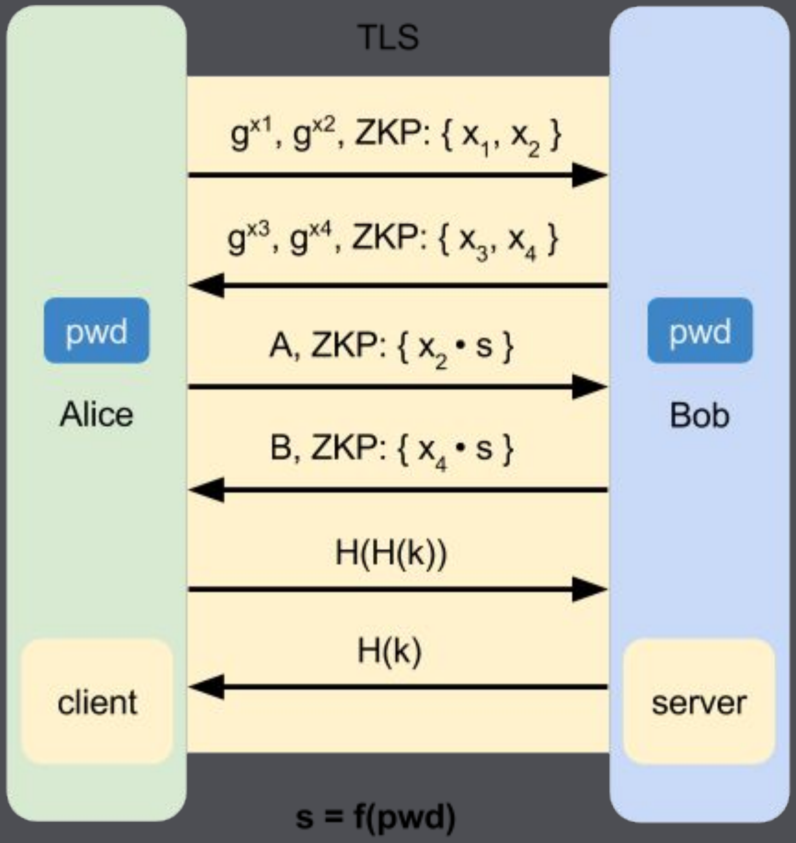
(If $h' = h$, then $V' = V$)

- ID is a unique user identifier
- $r, h, h' \in Z_q$, the set of congruence classes modulo q
- $x, v \in Z_q^{\times}$, the multiplicative group of integers modulo q
- $X, V, V' \in G$

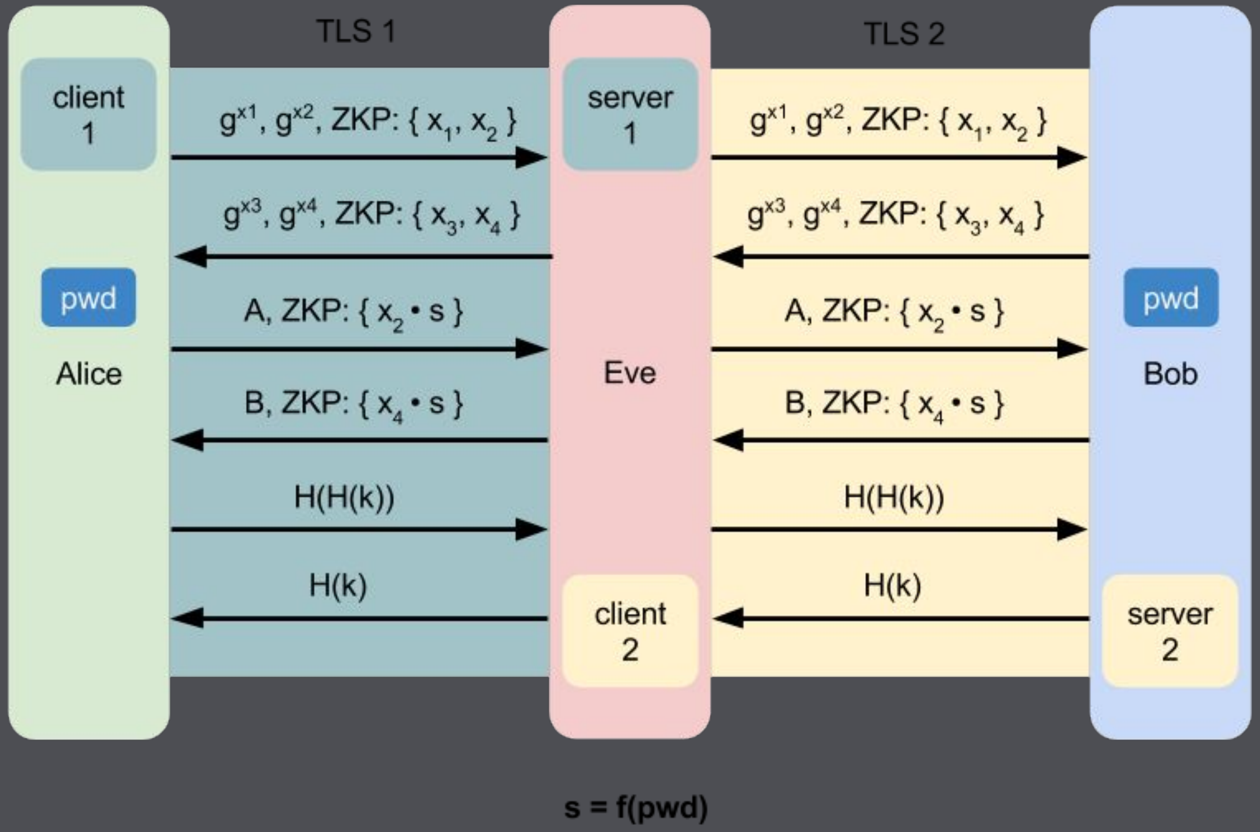
Ref - F. Hao, P. Ryan. J-PAKE: Authenticated Key Exchange Without PKI (http://eprint.iacr.org/2010/190.pdf)

J-PAKE over TLS

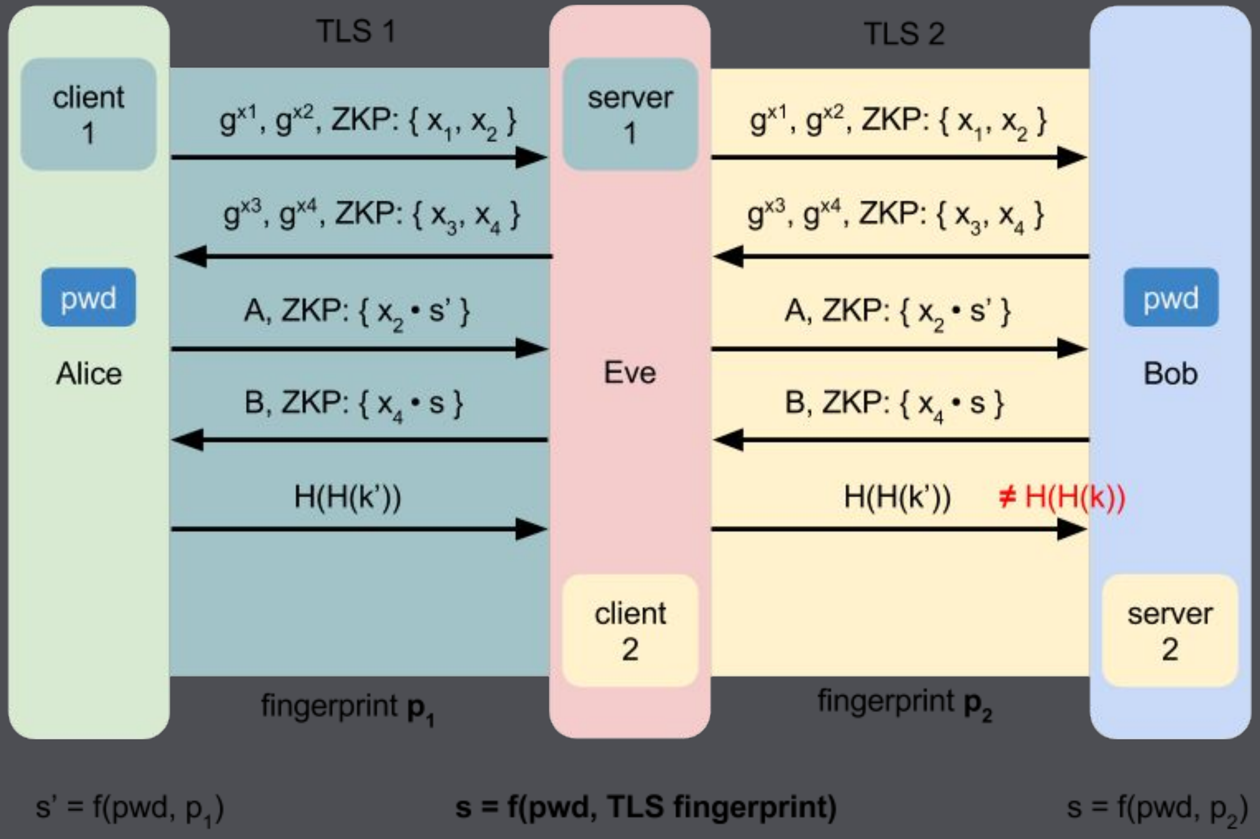
J-PAKE over TLS



J-PAKE over TLS

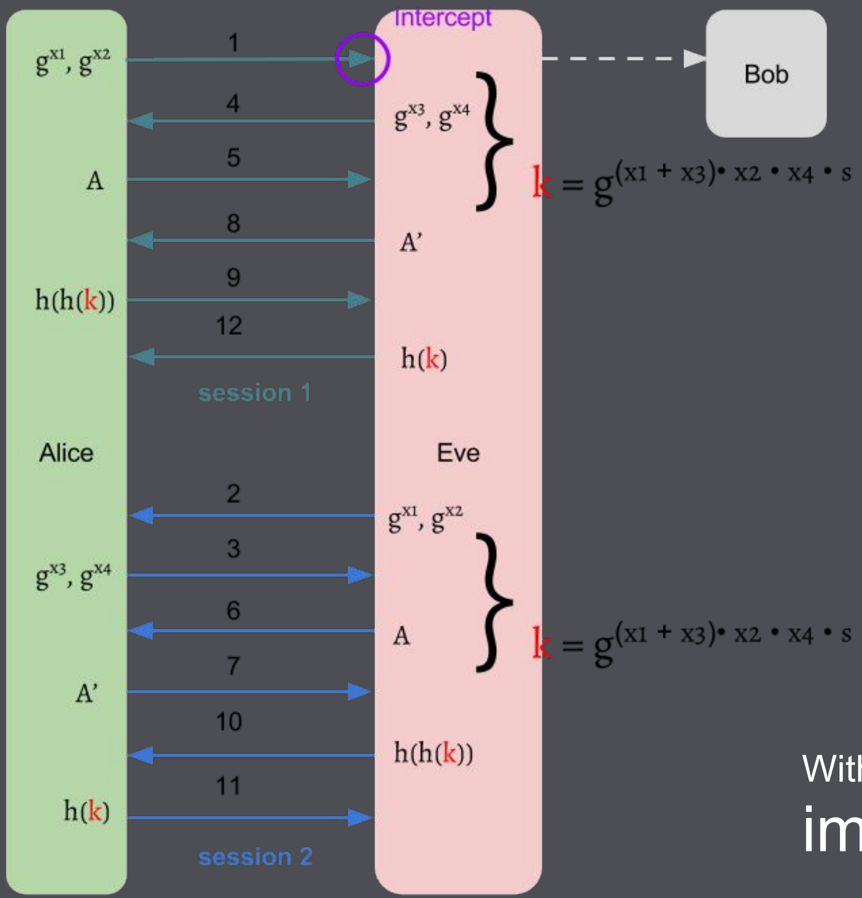


J-PAKE over TLS



Discussion

Why does J-PAKE need ZKP?



1. Eve intercepts Alice's g^{x1}, g^{x2} for Bob to initiate session 1
2. Eve sends g^{x1}, g^{x2} back to Alice to initiate session 2
3. Alice replies g^{x3}, g^{x4} on session 2
4. Eve sends g^{x3}, g^{x4} back on session 1
5. Alice sends A on session 1
6. Eve sends A back on session 2
7. Alice replies A' on session 2
8. Eve replies A' back on session 1
9. Alice sends h(h(k)) on session 1
10. Eve sends h(h(k)) on session 2
11. Eve replies h(k) on session 2
12. Alice replies h(k) on session 1

Without ZKP, J-PAKE will suffer

impersonation-attack

Why do we need TLS instead of using J-PAKE key directly?

- Save the effort to establish TCP channel and negotiate the encryption module

Why don’t we just use password to authenticate each other?

- Password is human-memorable weak secret
- PAKE can keep safe of the established session

When do we use J-PAKE

- If the device is unable to operate large exponents
- If the network latency isn't too long

Why x_2, x_4 can not be 0?

- If x_2 , x_4 , s = 0, then K = 1
- An attacker can intentionally choose x_2 = 0 or x_4 = 0 to get K = 1 even he doesn't know the password

mozilla

See more here