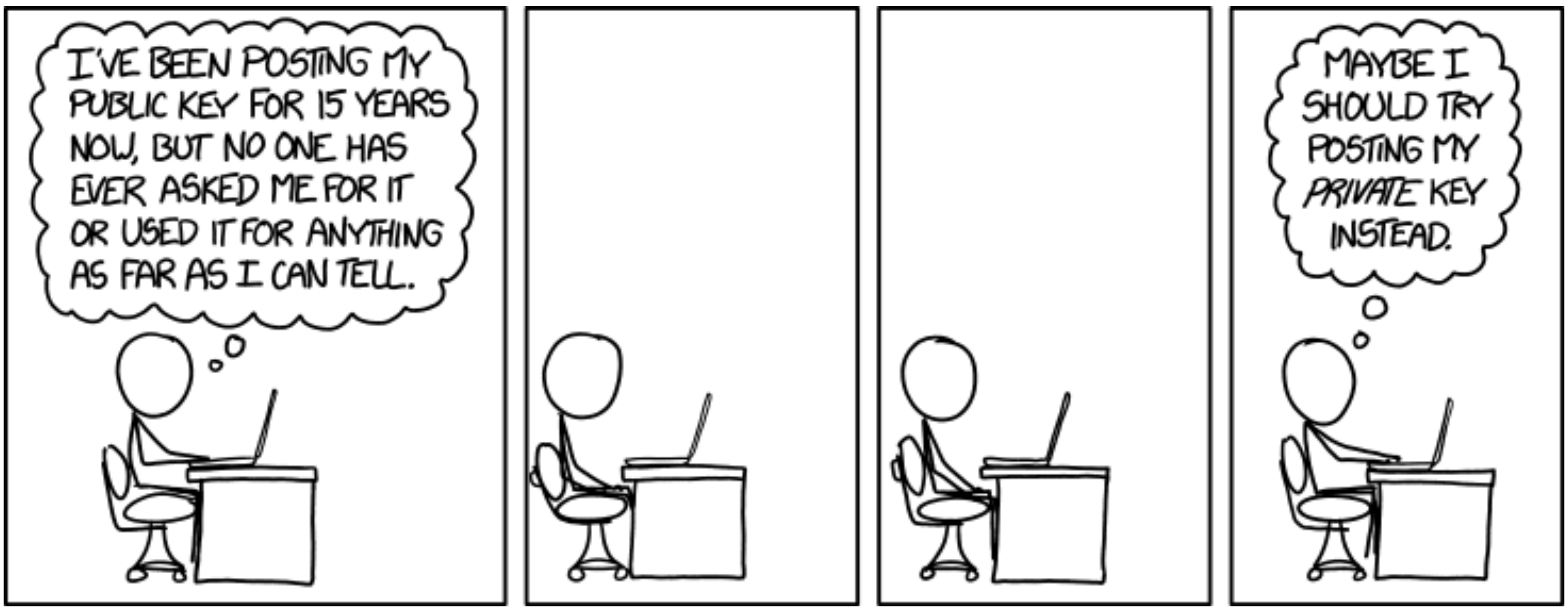
Public-Key Distribution

February 17, 2022



© xkcd.com

Homework 1, Midterm Exam, and Today

- Homework 1: due **February 20th** (Sunday) at 11:59pm
- Midterm exam: on March 1st (Tuesday after next)
  - detailed list of topics and sample midterm will be available on Blackboard
  - in person, during class
  - closed book, based on first five weeks of classes
- Today: **public-key distribution**
  - Diffie-Hellman key exchange
  - digital certificates

Feedback: `https://forms.gle/JGbNCmCsU69iWaTv8`

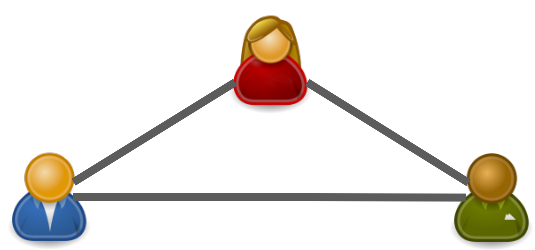
Key Distribution using Public-Key Cryptography

Reminder:
Key Distribution

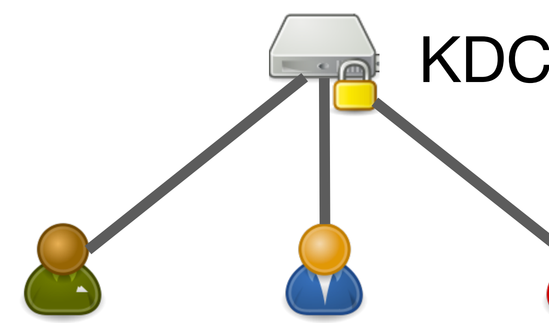
- **Symmetric-key cryptography:**
  efficient, but requires frequently setting up fresh secret keys

  → distribute **short-term session keys** using **long-term master keys**

Decentralized



Centralized



How to set up master keys?

may be difficult or impossible

- deliver **manually** or using some **secure channel**

- use **public-key cryptography** to set up secret keys
  → communication parties **do not need to have a shared secret**

Diffie-Hellman Key Exchange

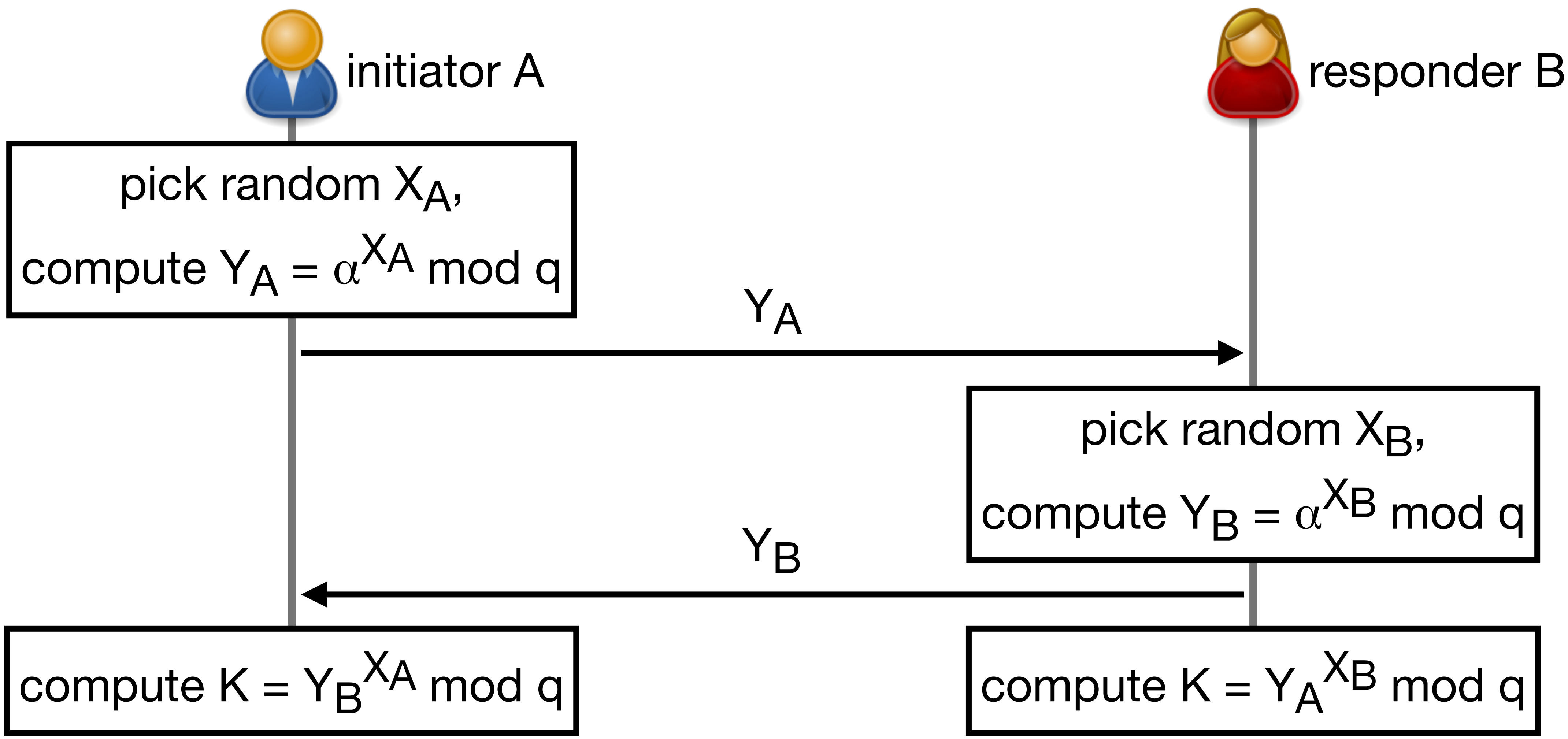
- Designed by Whitfield Diffie and Martin Hellman in 1976
  - first published public-key algorithm / protocol
- Very widely used
  - *example*: SSL/TLS, SSH
- ElGamal (and similar crypto primitives) are based on the idea of D-H
- Security depends on the **hardness of finding discrete logarithms**: given α, **y**, and **q**, find an **x** that satisfies

$$y = \alpha^x \bmod q$$

  - widely believed to be a computationally-hard problem
- If **q** is prime, then α is a **primitive root** if $\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{(q-1)}$ are all different modulo **q**

Diffie-Hellman Protocol

- Public: let **q** be a large prime number, and α be a primitive root of **q**

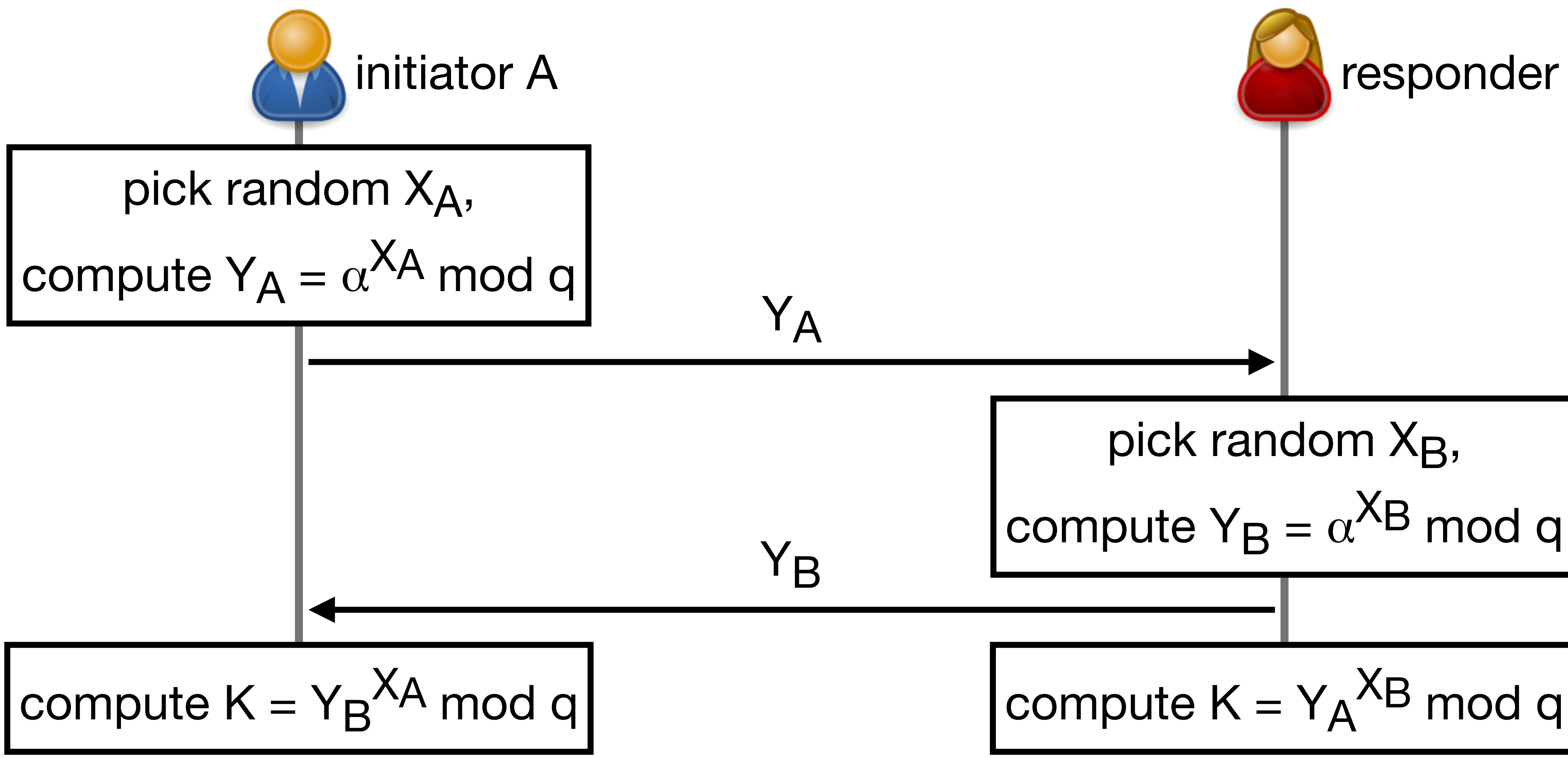


$$K = {Y_B}^{X_A} = (\alpha^{X_B})^{X_A} = \alpha^{X_A X_B} = (\alpha^{X_A})^{X_B} = {Y_A}^{X_B} = K \bmod q$$

- Secure against eavesdropping since computing X_A or X_B is hard
- Elliptic Curve D-H (ECDH): same principle, more efficient

~~Diffie-Hellman Protocol~~ Public-Key Encryption

- Public: let **q** be a large prime number, and α be a primitive root of **q**

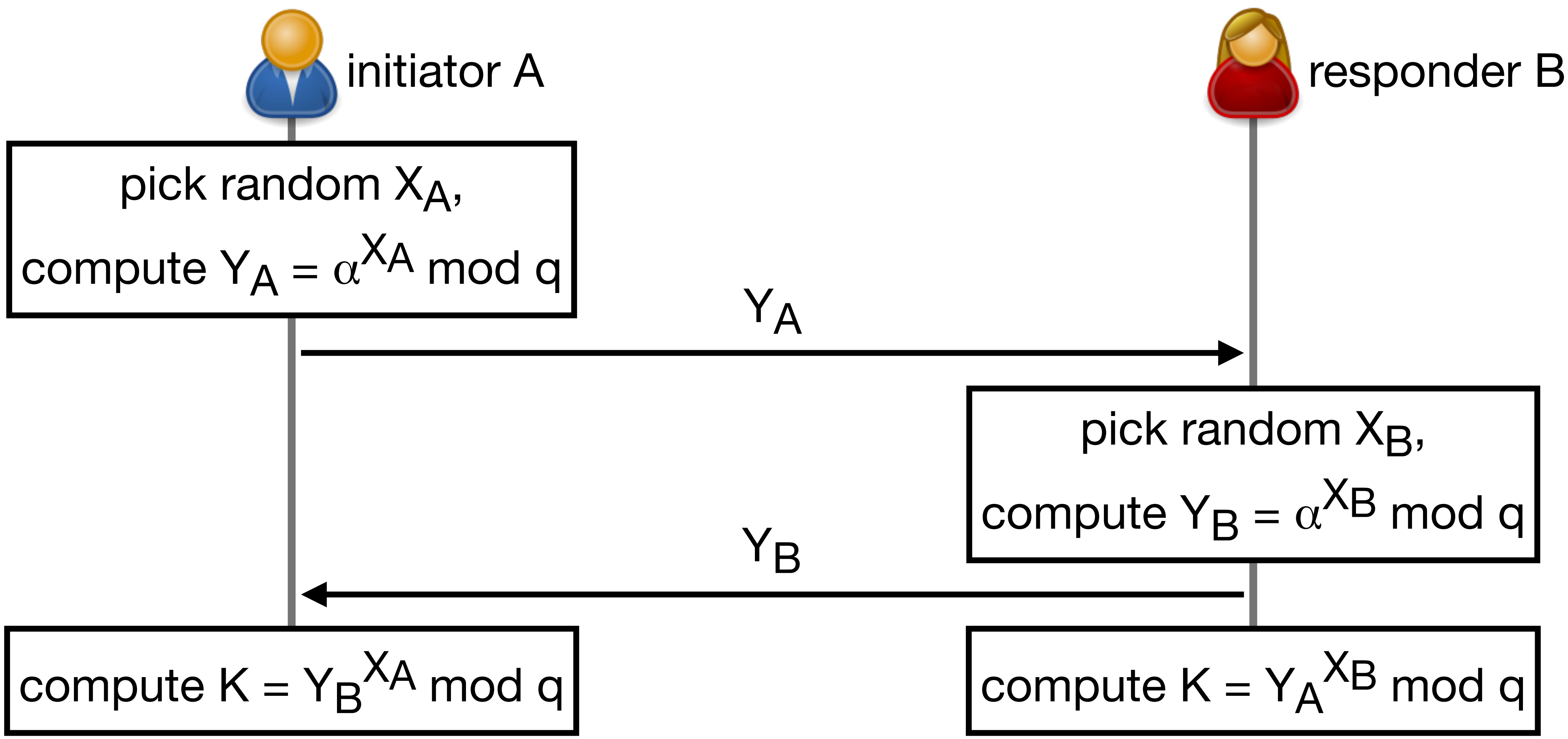


$$K = {Y_B}^{X_A} = (\alpha^{X_B})^{X_A} = \alpha^{X_A X_B} = (\alpha^{X_A})^{X_B} = {Y_A}^{X_B} = K \bmod q$$

- Secure against eavesdropping since computing X_A or X_B is hard
- Elliptic Curve D-H (ECDH): same principle, more efficient

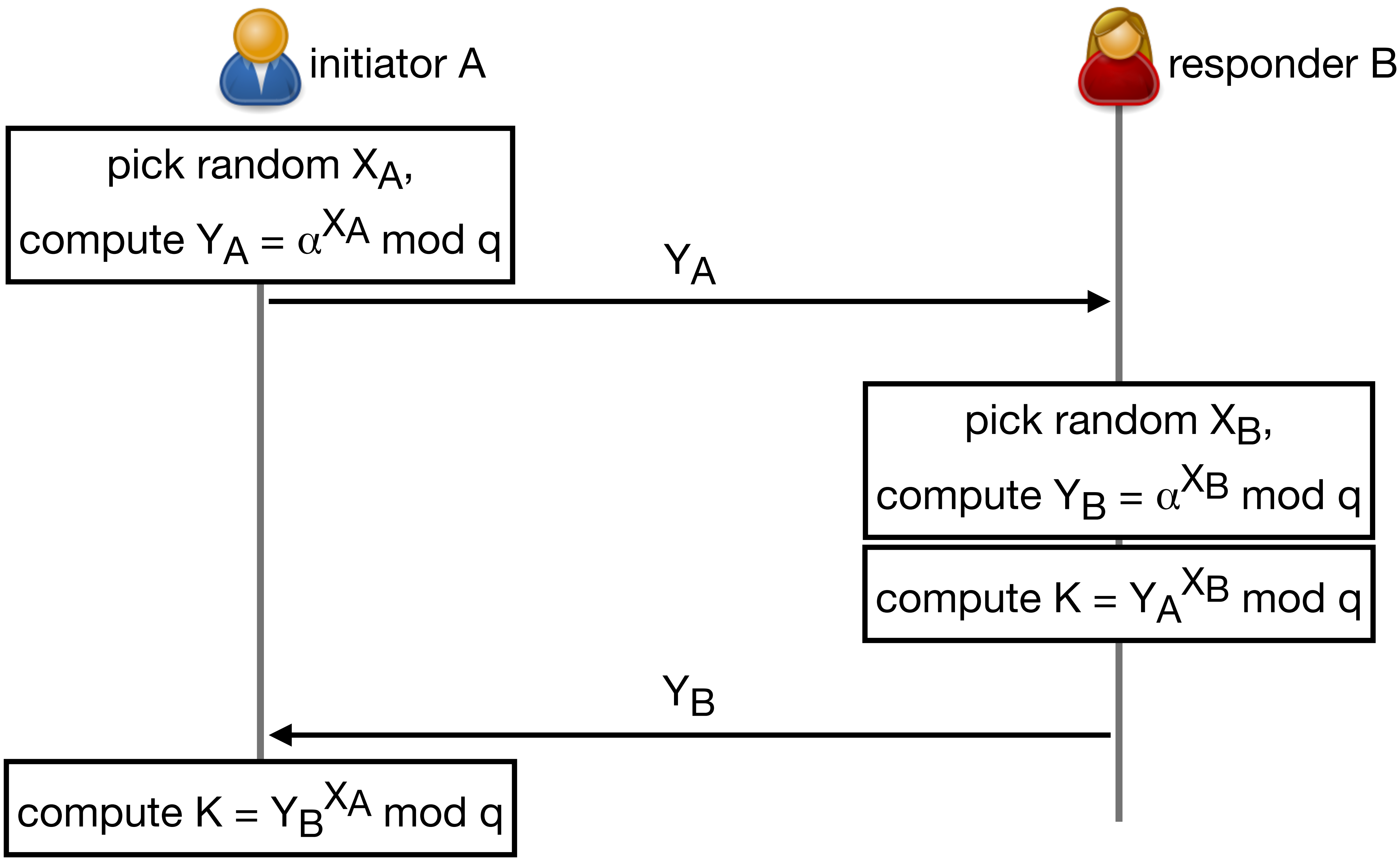
~~Diffie-Hellman Protocol~~ Public-Key Encryption

- Public: let **q** be a large prime number, and α be a primitive root of **q**



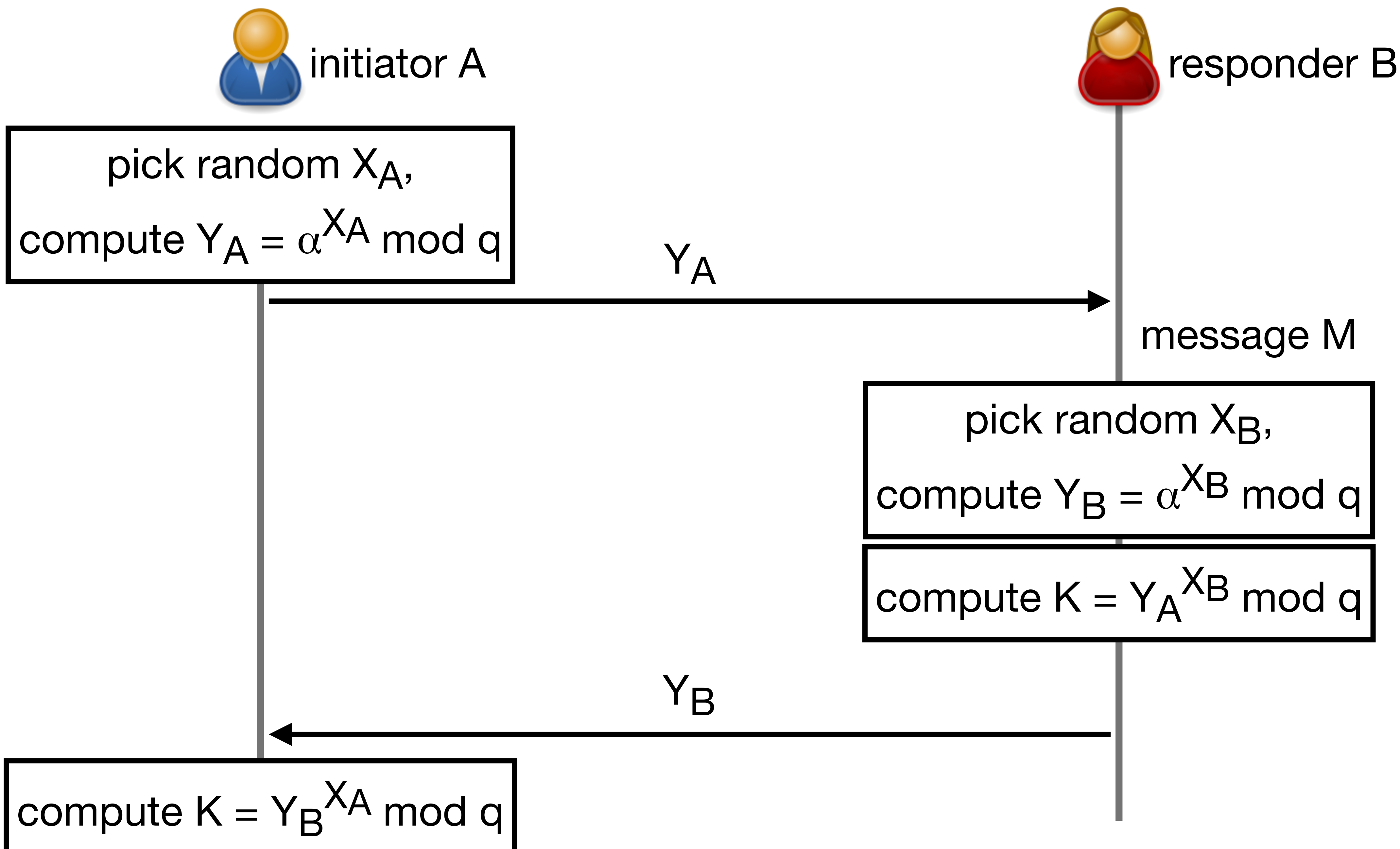
~~Diffie-Hellman Protocol~~ Public-Key Encryption

- Public: let **q** be a large prime number, and α be a primitive root of **q**



~~Diffie-Hellman Protocol~~ Public-Key Encryption

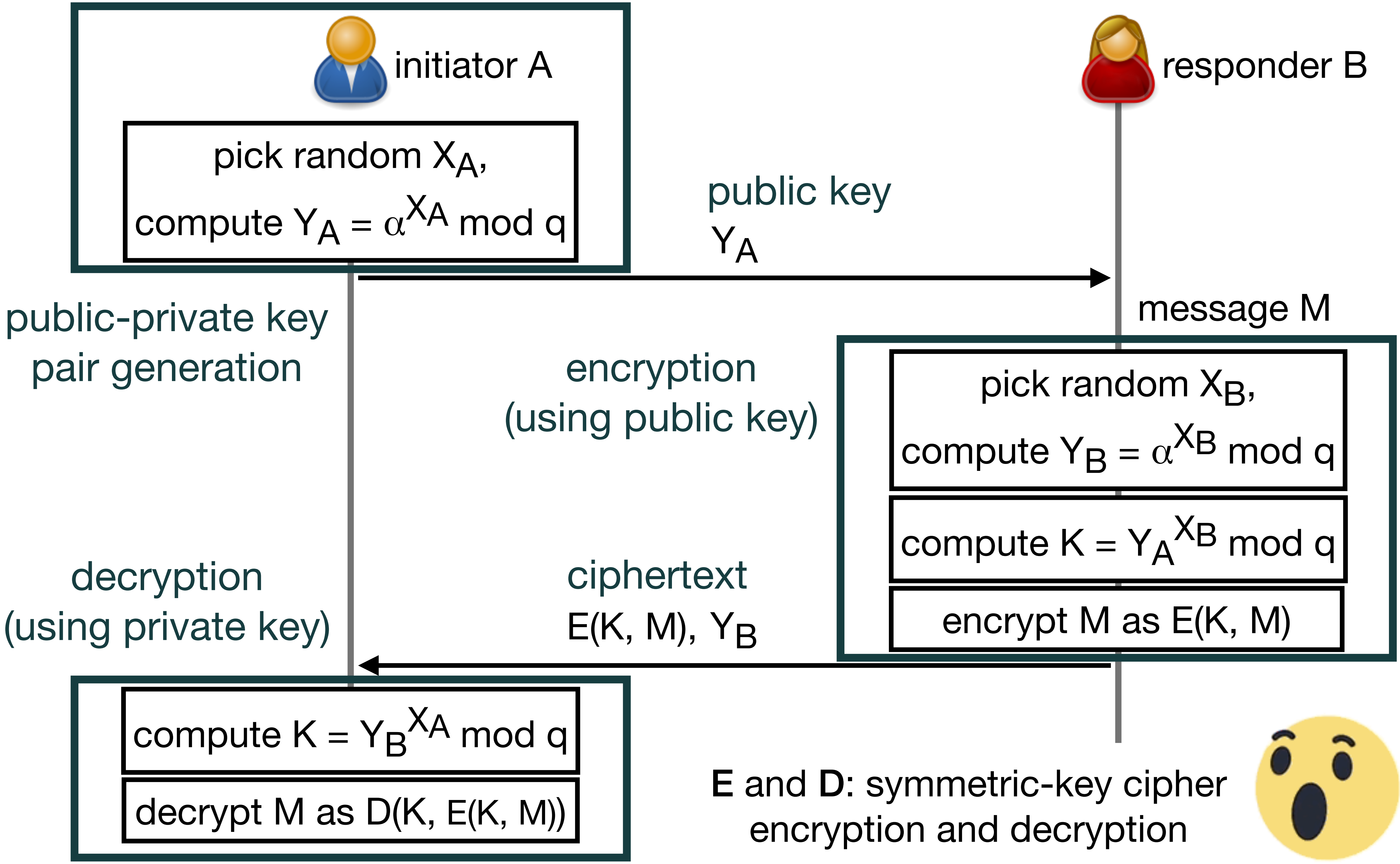
- Public: let **q** be a large prime number, and α be a primitive root of **q**



E and **D**: symmetric-key cipher encryption and decryption

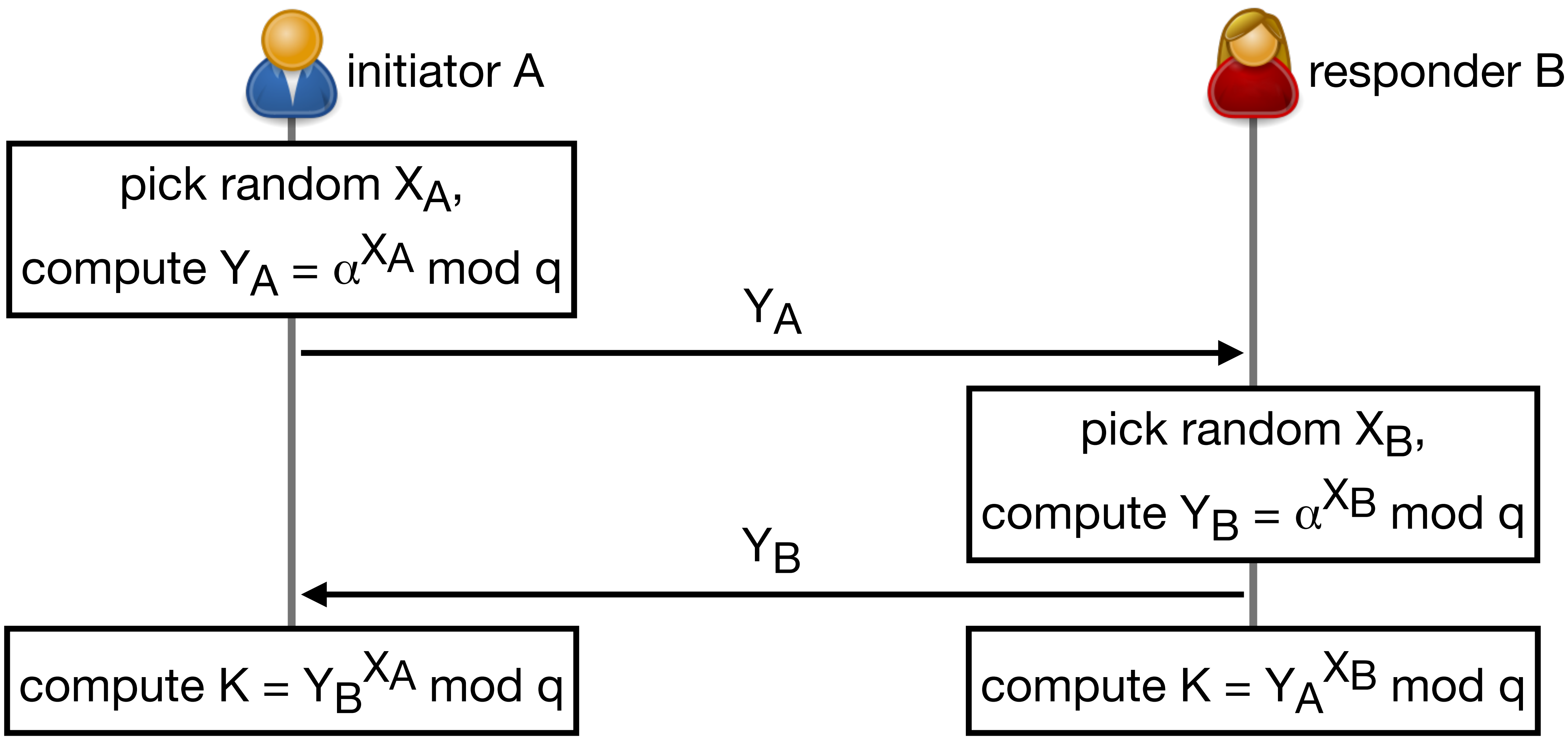
~~Diffie-Hellman Protocol~~ Public-Key Encryption

- Public: let **q** be a large prime number, and α be a primitive root of **q**



Diffie-Hellman Protocol

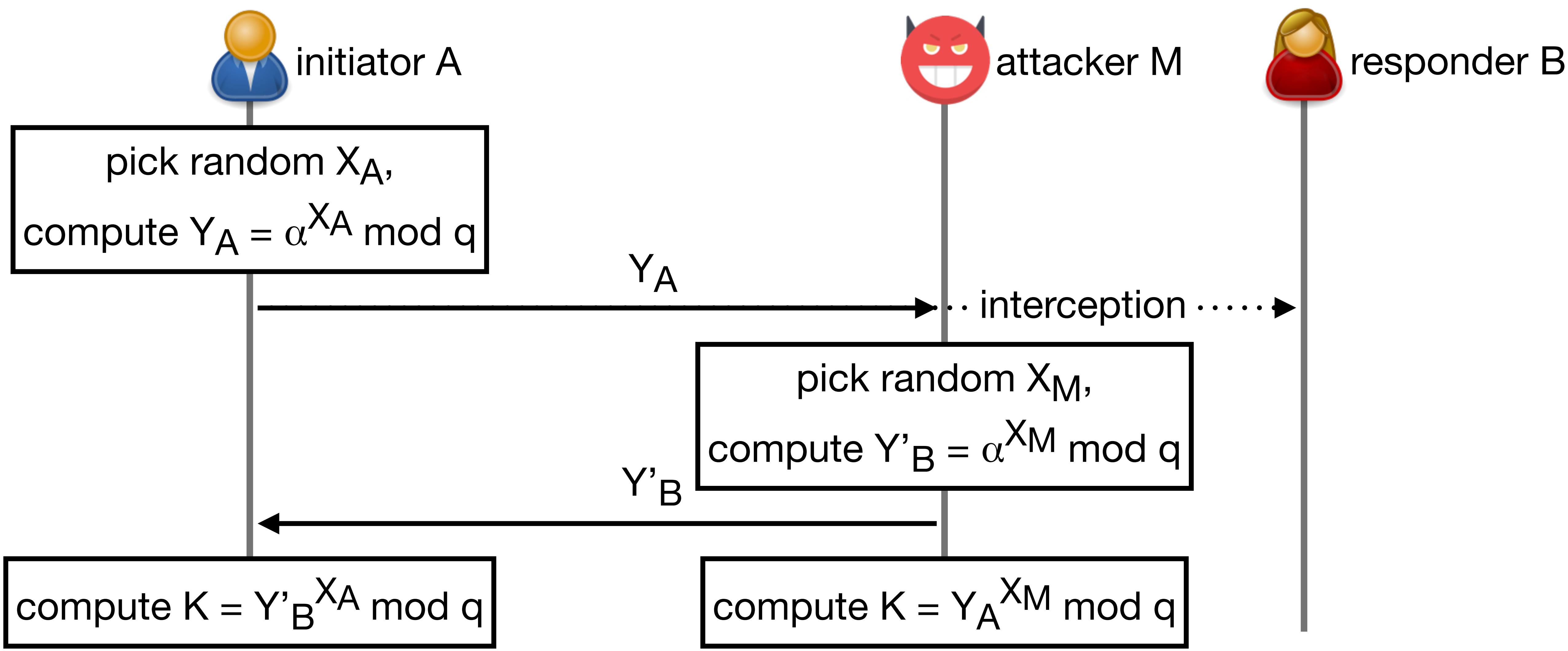
- Public: let **q** be a large prime number, and α be a primitive root of **q**



$$K = {Y_B}^{X_A} = (\alpha^{X_B})^{X_A} = \alpha^{X_A X_B} = (\alpha^{X_A})^{X_B} = {Y_A}^{X_B} = K \bmod q$$

- Secure against eavesdropping since computing X_A or X_B is hard
- Elliptic Curve D-H (ECDH): same principle, more efficient

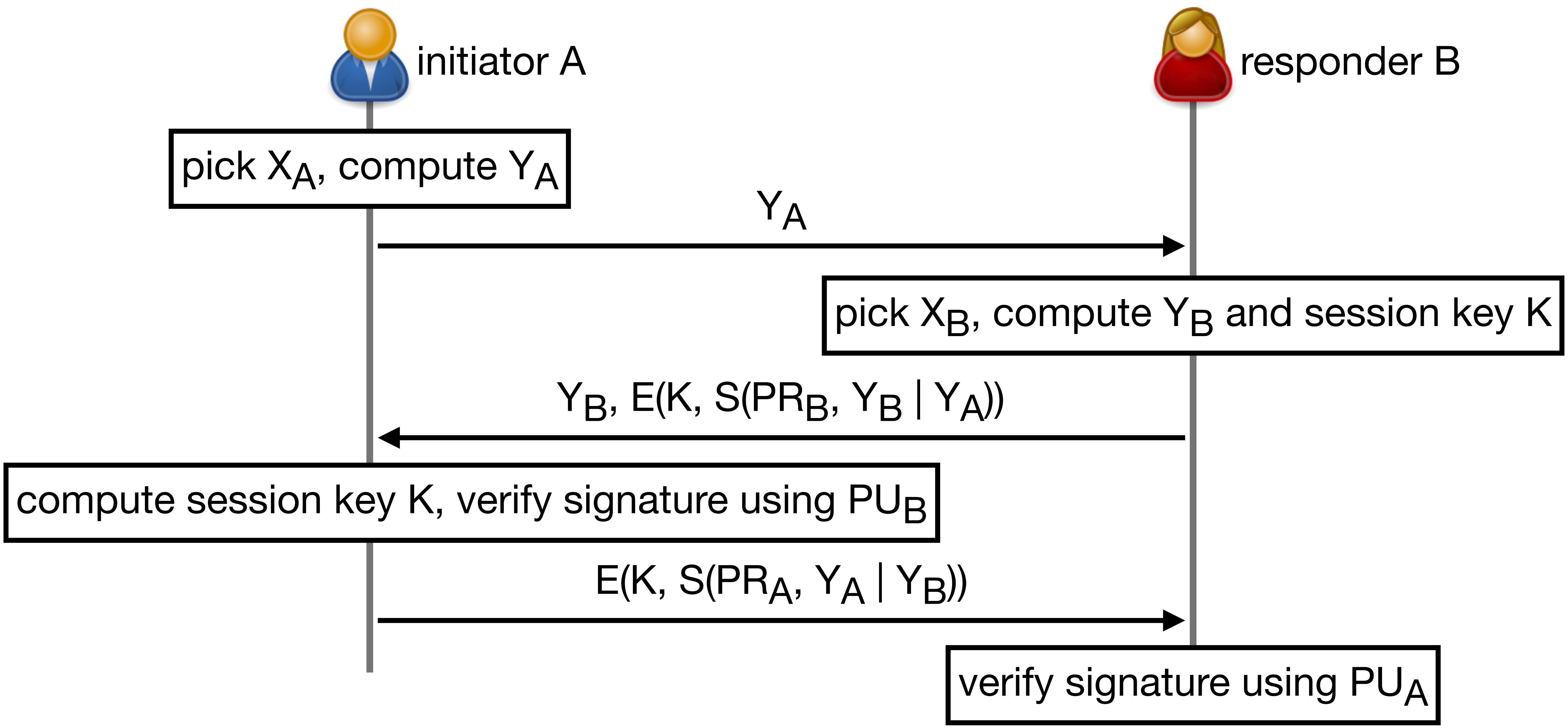
Diffie-Hellman Key Exchange
Man-in-the-Middle Attack



- A thinks that it shares a secret key with B, but it actually shares the key with attacker M

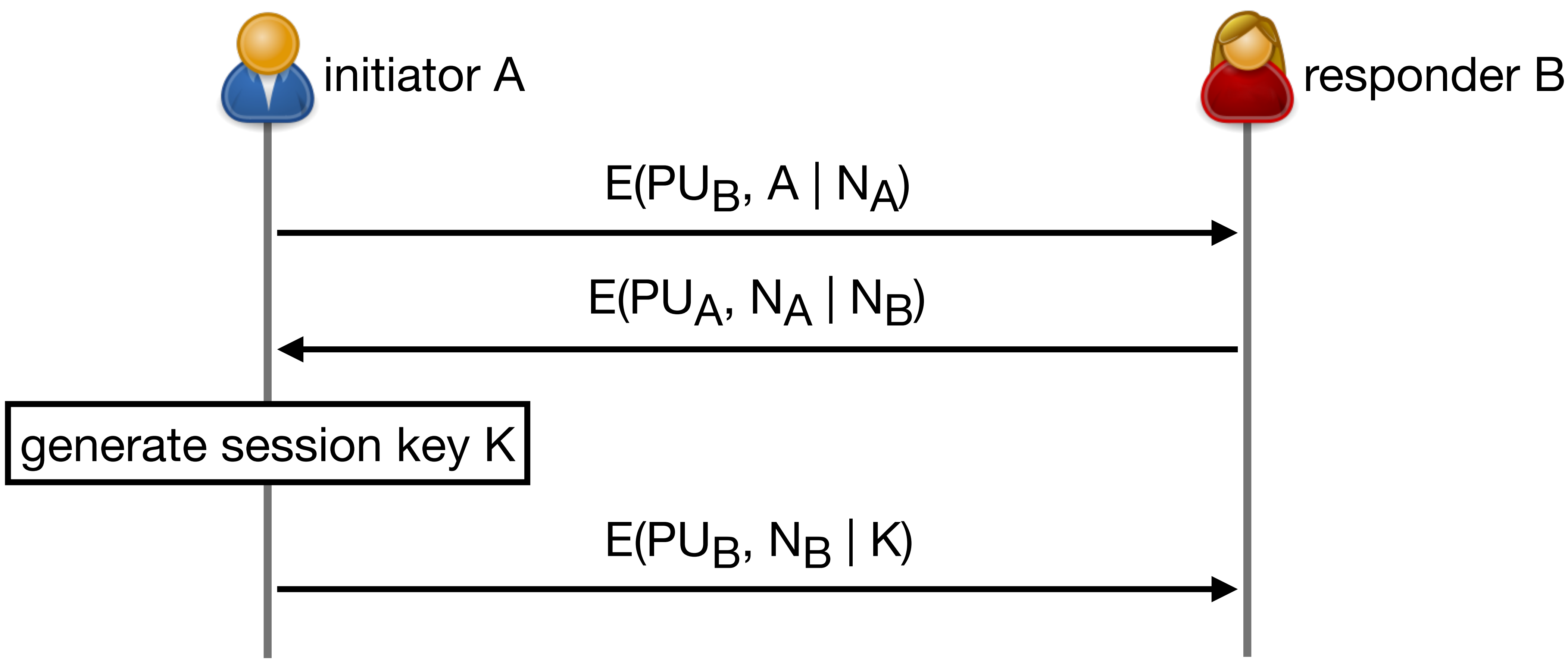
Station-to-Station Protocol

- Assume that **A** knows **B**'s public key $\mathbf{PU_B}$ and **B** knows **A**'s public key $\mathbf{PU_A}$
- Digital signature: **S(PR, M)** is message **M** signed using private key **PR**



Key Distribution Using Public-Key Encryption

- *Example*:

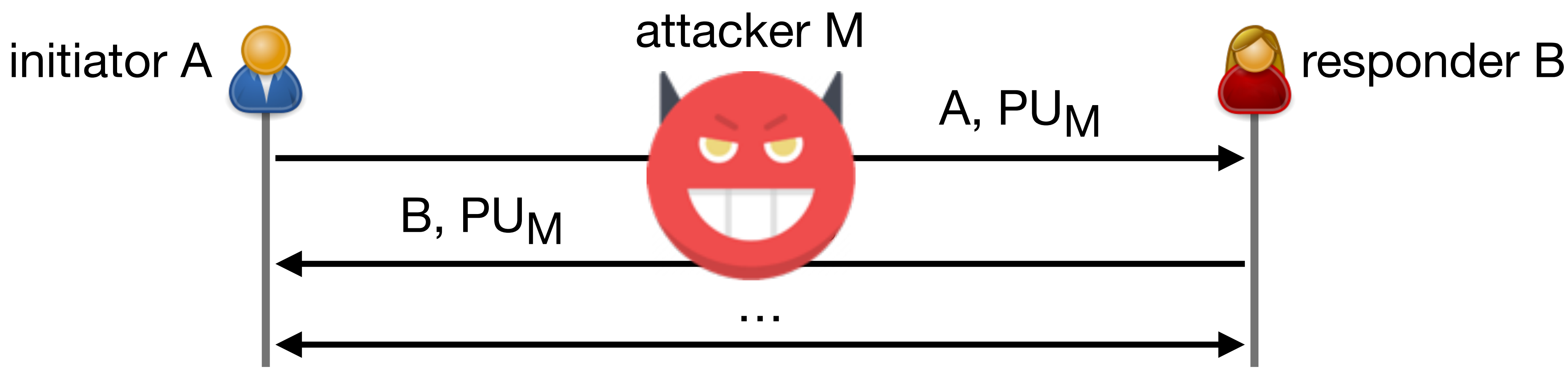


- Assumes that communication parties know each other's public keys

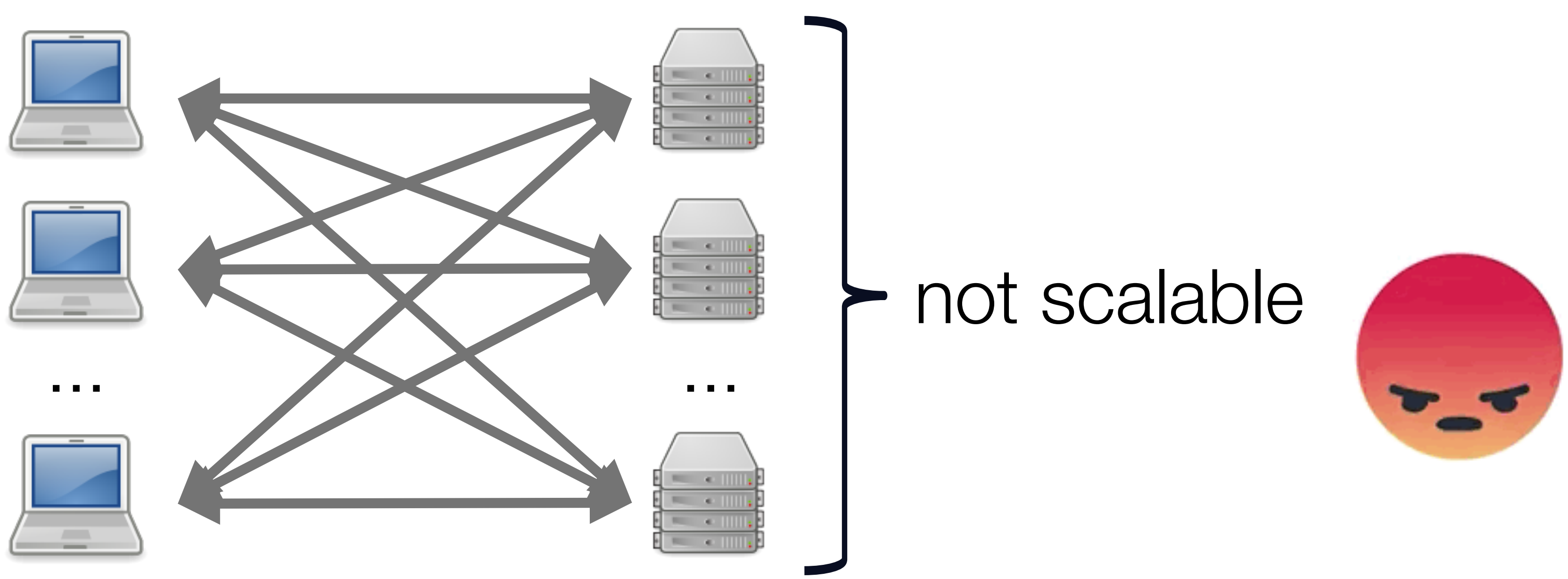
Distributing Public Keys

How to distribute public keys?

- Naïve public-key distribution



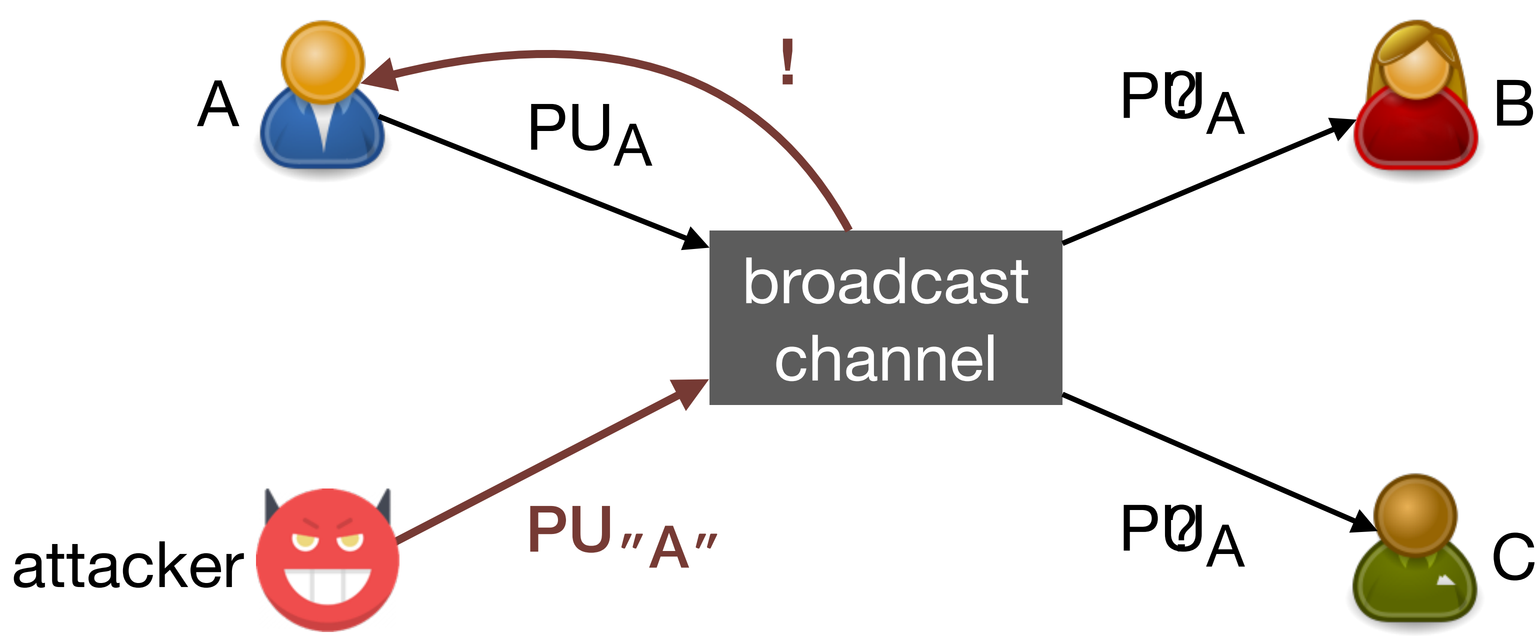
- Manual public-key distribution (*e.g.*, physical transfer) for all pairs



Distribution of Public Keys

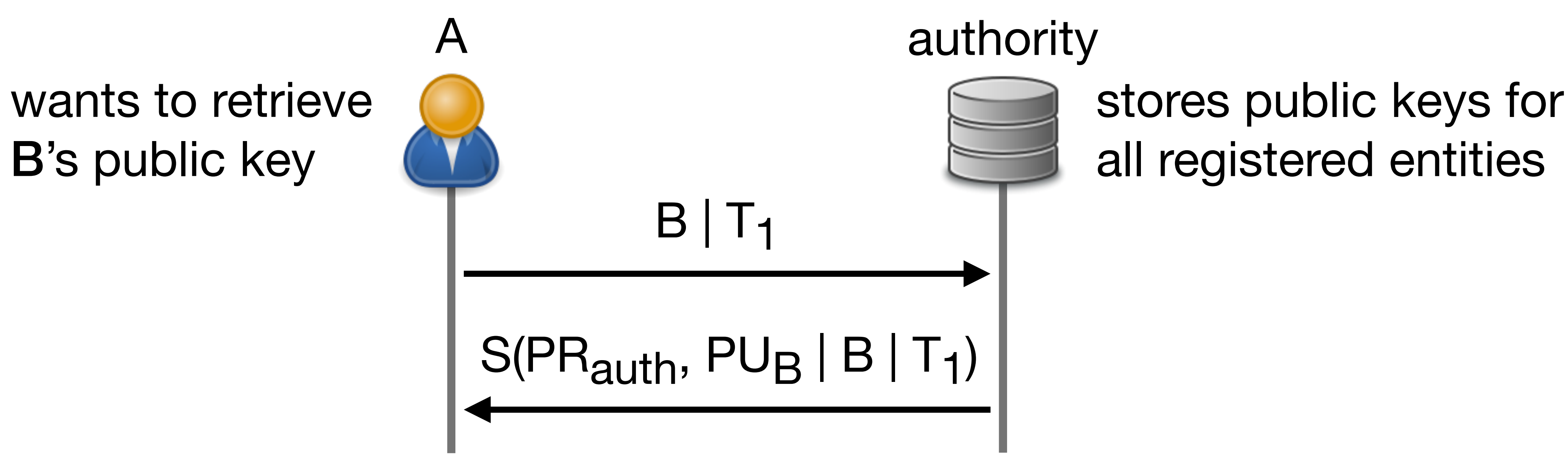
Public Announcement

- Announce the public key on some **broadcast channel**
  - *example*: announce on some public forum, such as an e-mail list or social media
  - public keys used for PGP (Pretty Good Privacy) are often distributed in this way
- *Weakness*: **anyone can forge** such an announcement
  - impersonated entity can detect the attack and notify others
  - but until then, the forger can impersonate the victim



Public-Key Authority

- Each participant knows the **public key PU_{auth} of an authority** that maintains a publicly available **directory of public keys**
- *Weaknesses*
  - authority is a **single point of failure**
  - authority must be **online**

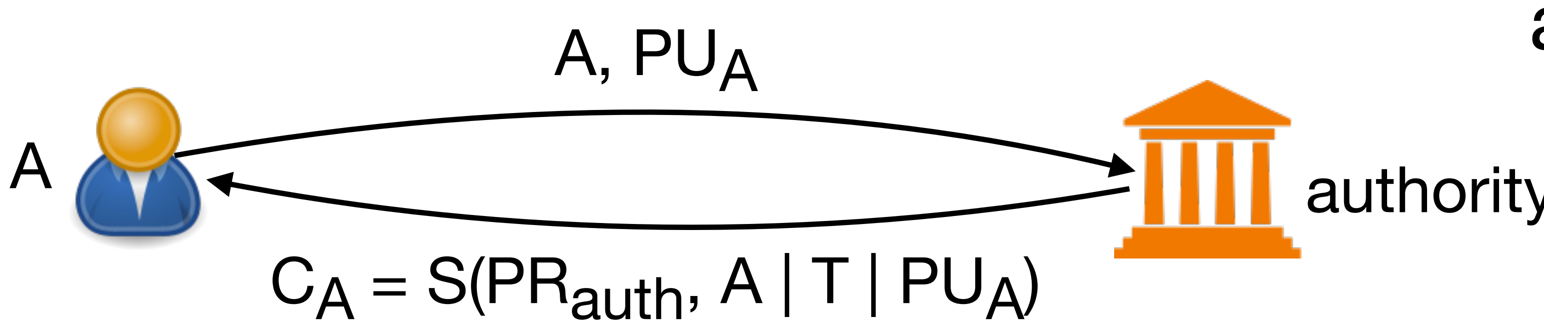


Public-Key Certificates

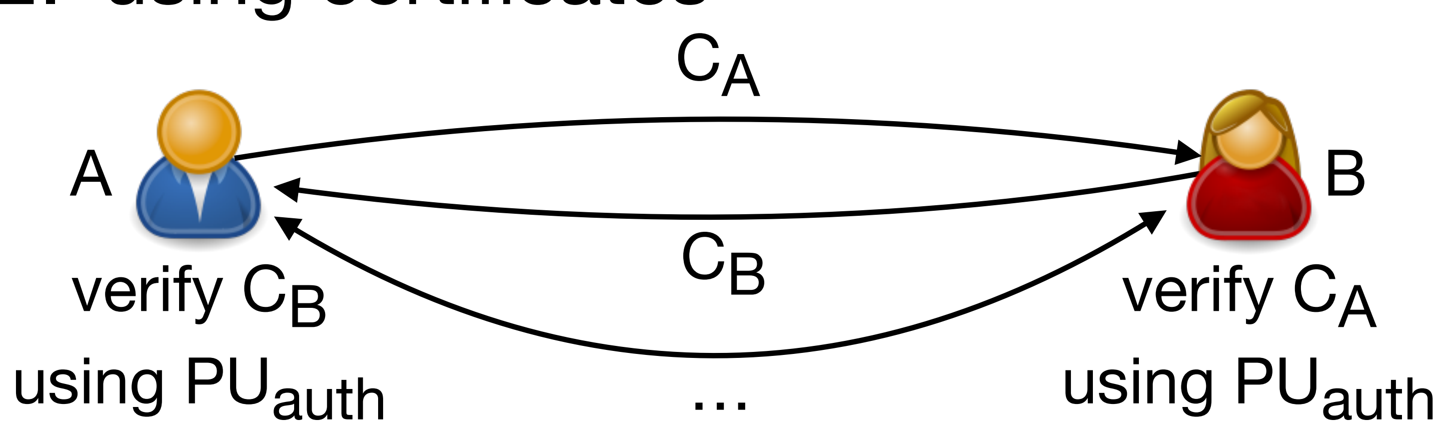
- **Certificates**
  - enable participants to **exchange keys without contacting the authority**
  - in a way that is as **secure** as if the keys were obtained from the authority
- Two phases:

1. requesting / issuing a certificate

through a **secure authenticated channel** (for example, in person)



2. using certificates



proves the **authenticity of public keys** without using the authority

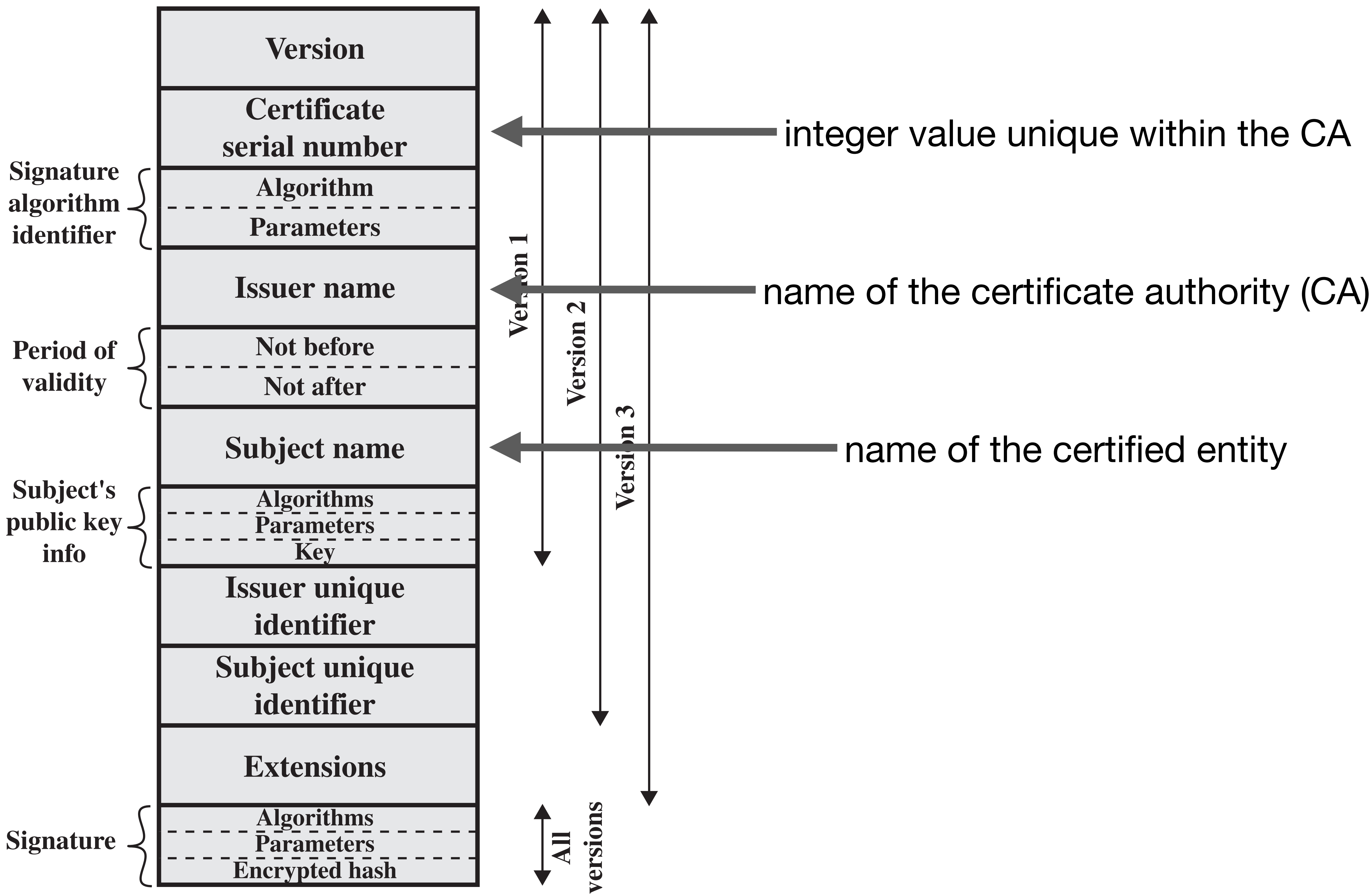
Public-Key Certificate Properties

- Certificate = $\underbrace{\text{owner's name, public key, timestamp, …}}_{\text{signed by the certificate authority}}$
- Requirements
  1. any participant can read a certificate to determine the name and public key of the certificate's owner
  2. only the certificate authority can create certificates
  3. any participant can verify that a certificate originated from the authority
  4. any participant can verify that a certificate is recent
- Problem: compromised private key
  - if an attacker has learned the private key PR_A of an entity A, then the attacker can use A's certificate C_A to impersonate A

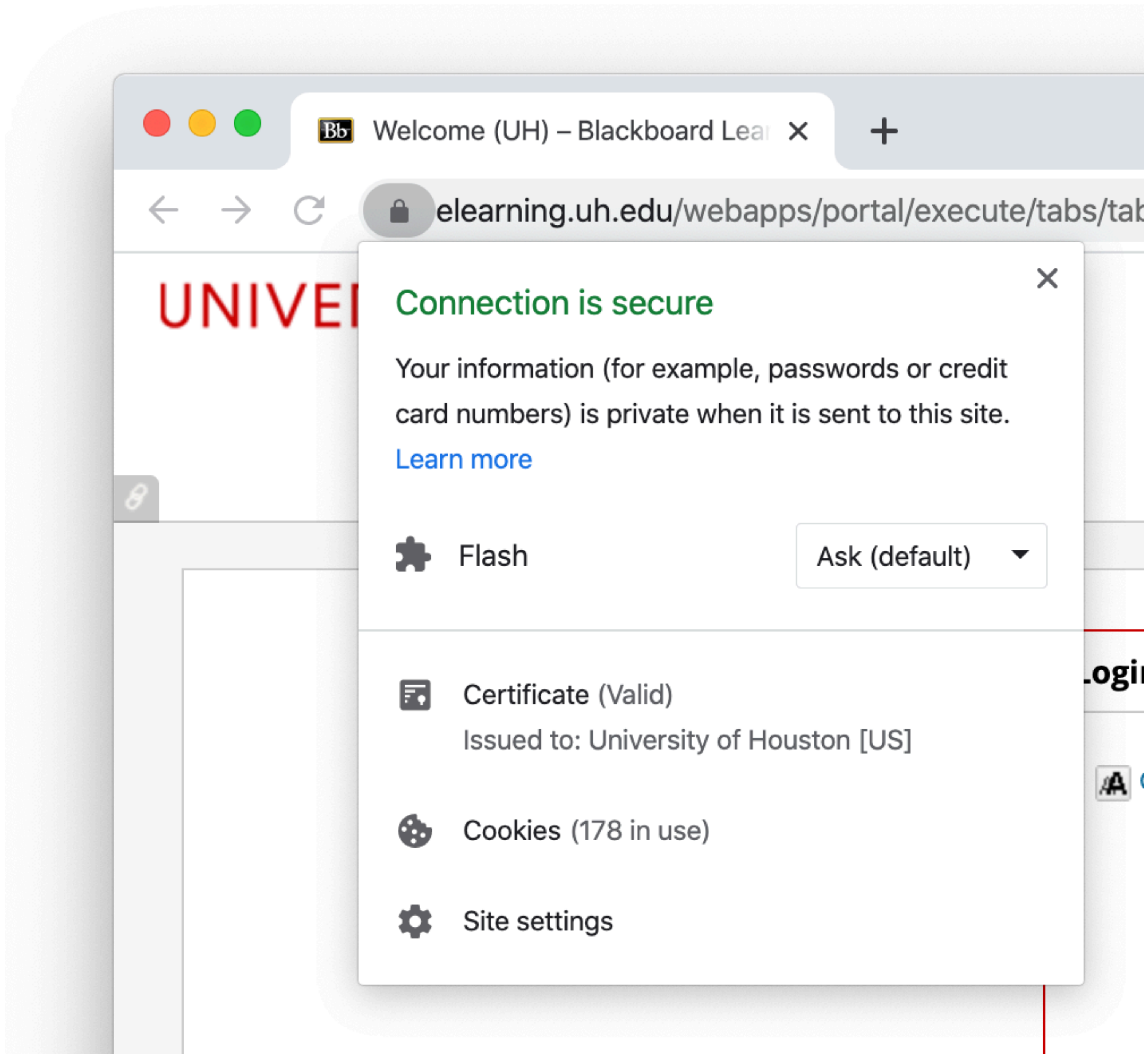
X.509 Certificates

- X.509 standard
  - ITU-T standard for public-key certificates and related functions
  - first published in 1988, updated multiple times
  - does not dictate specific algorithms (*e.g.*, for signature)
- Very widely used
  - SSL/TLS
  - IPSec
  - S/MIME
  - …

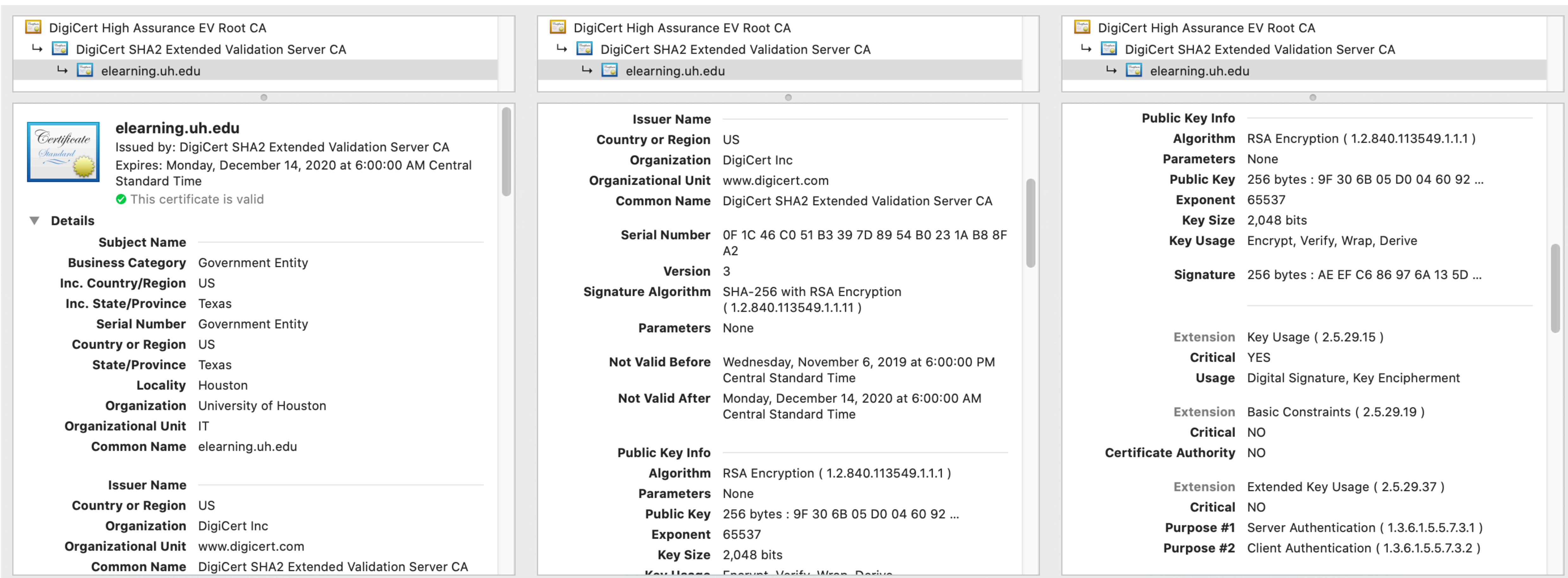
X.509 Format


Version
Certificate serial number
Signature algorithm identifier
Algorithm
Parameters
Issuer name
Period of validity
Not before
Not after
Subject name
Subject's public key info
Algorithms
Parameters
Key
Issuer unique identifier
Subject unique identifier
Extensions
Signature
Algorithms
Parameters
Encrypted hash
Version 1
Version 2
Version 3
All versions
integer value unique within the CA
name of the certificate authority (CA)
name of the certified entity

X.509 Certificate in Practice



X.509 Certificate in Practice



Certificate Authorities in Practice

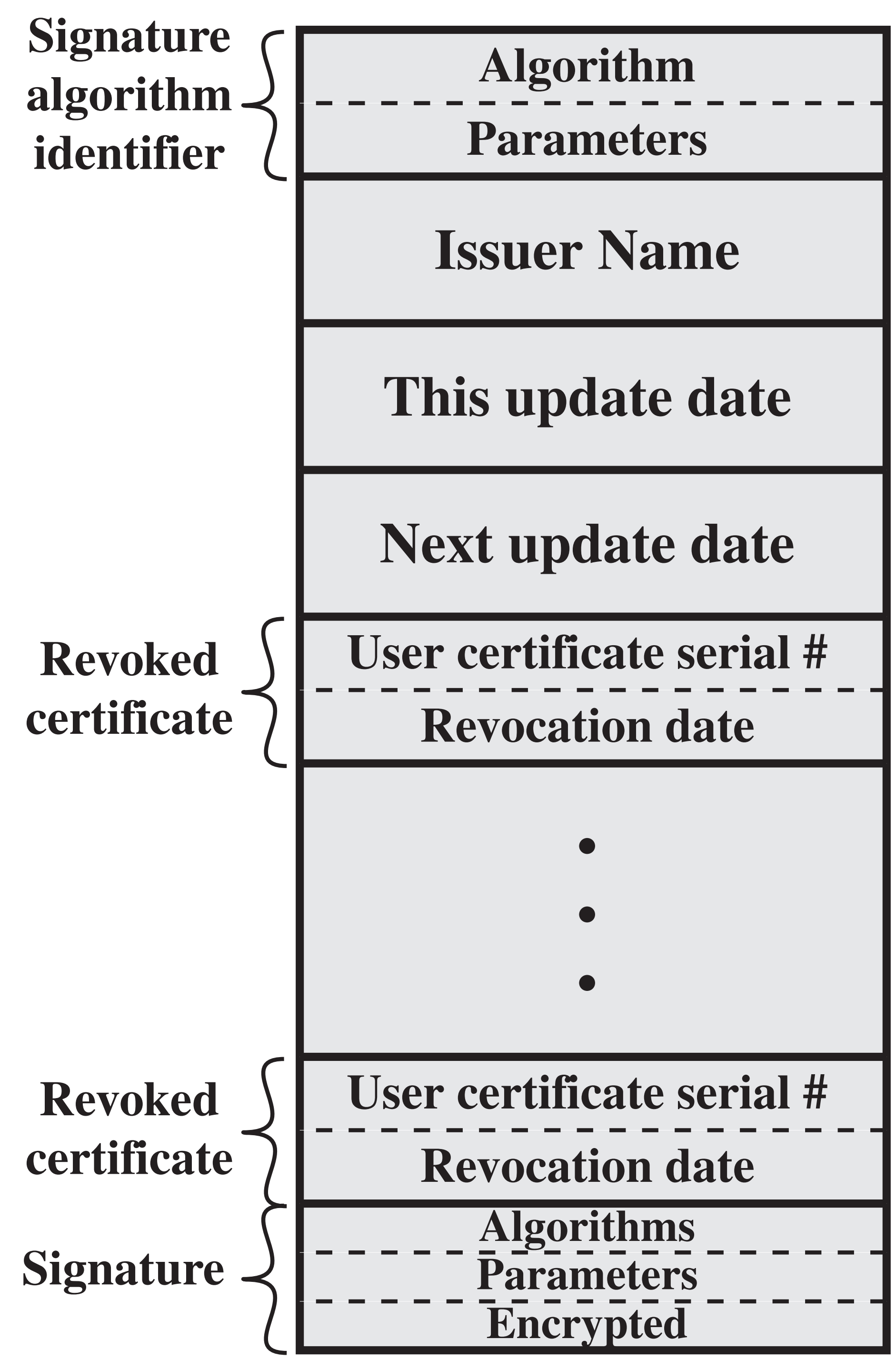
- Operating systems and web browsers typically come with a **list of trusted certificate authorities** (*e.g.*, Microsoft Root Certificate Program, Mozilla Root Certificate Program)
  - these CAs are **trusted by the developers** (*e.g.*, they follow security standards)
  - users can add to or remove CAs from this list
- Common types of CAs
  - **commercial:** charges a fee for issuing a certificate

CA	Market share (2021 September)
IdenTrust	54%
DigiCert	19%
Sectigo	17%
…	

  - **governmental**
  - **private non-profit** (*e.g.*, CAcert)

Certificate Revocation (the Ugly Part)

- Revocation is necessary if
  - private key of the owner is compromised
  - owner is no longer certified
  - authority's certificate is compromised
- Each CA maintains a
  Certificate Revocation List (CRL)
  - signed and published by the CA
  - when checking the validity of a certificate, one must check if it is on the CRL
- For efficiency, clients cache the list
  → revoked certificates may be accepted until the cache expires



Conclusion

- Distributing symmetric (i.e., secret) keys
  - **decentralized** → not scalable
  - **centralized** (*e.g.*, extended Needham-Schroeder, Kerberos)
  - **public-key cryptography** (*e.g.*, Diffie-Hellman key exchange)
- Distributing public keys
  - public announcement, public-key authority
  - **public-key certificates**
    - requesting on a secure channel (*e.g.*, in person)
    - certificate proves the authenticity of a public key
    - revocation lists signed and published by the CA

Next lecture:

Security Protocols

Certificate Chains

- Entities that do not trust a common CA are not able to directly verify each other’s certificates
- X.509 certificate chain
  - trust can be established through a chain of CAs who trust each other
  - *example*:
    C wants to verify B’s certificate
    → X«W» W«V» V«Y» Y«Z» Z«B»