CS489/698 Privacy, Cryptography, Network and Data Security

Authentication Protocols

Spring 2024, Monday/Wednesday 11:30am-12:50pm

A1 is due today!

- Late policy from today 3pm until May 31st 3pm.
  - No further help will be provided

Today’s Lecture – Authentication Protocols

- Symmetric Authentication
    - Needham-Schroeder
    - Kerberos
- Asymmetric Authentication (PKI)
    - DH
    - Certificates
- PAKEs
- Single Sign On
    - SAML
    - OAuth
- DNSSEC

Recall, Definition of Authentication

Authentication = { **Identification** + Verification }

Recall, Types of Authentication Tokens

- Something you know
  - Passwords, pins, etc

- Something you have
  - Mobile phones (SMS), RSA tokens, etc.

- Something you are
  - Fingerprints, retinal scans, etc.

- (Experimental) Something you do
  - Keystroke metrics, behavioral patterns, etc.

Today’s Focus

- Establishing Keys:
    - Typically, once authenticated, we give access to some service or message
    - Goal will typically be to establish a symmetric key between parties

Symmetric Crypto Authentication

Needham-Schroeder

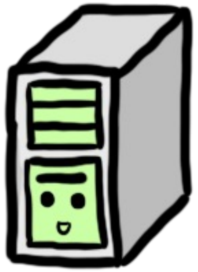
Needham-Schroeder Overview


Alice (A)


K_{AC}


Bob (B)


K_{BC}


Key Distribution Center (C)

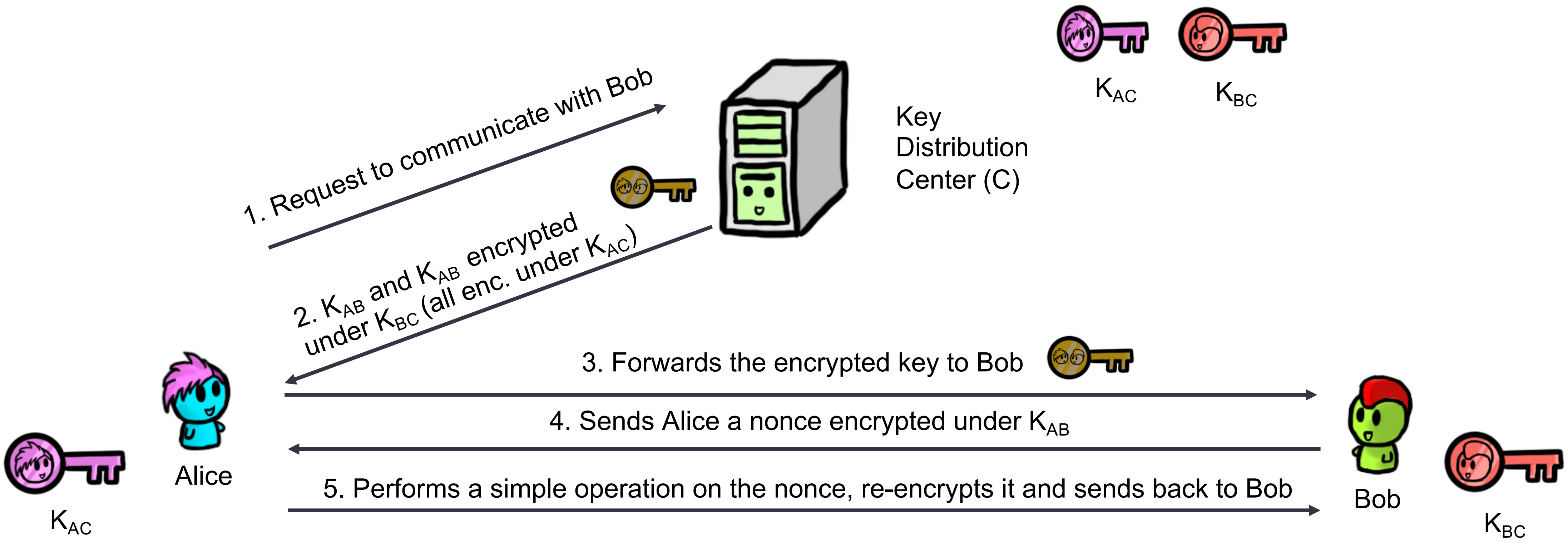

K_{AC}


K_{BC}

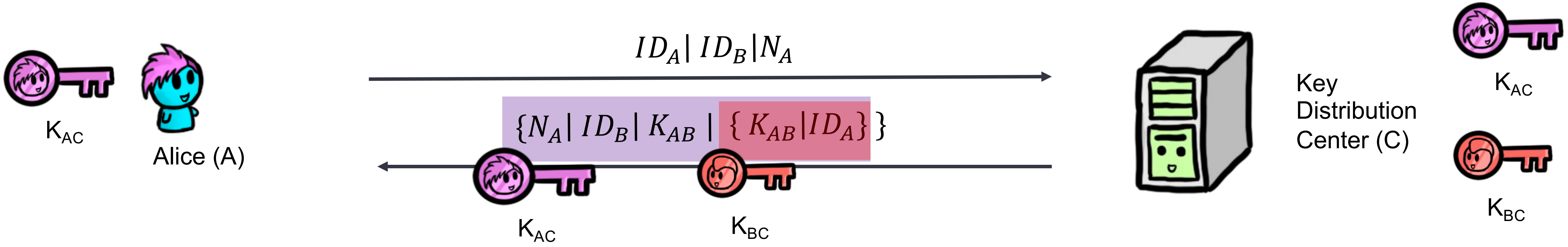
- Alice (A) wants to initiate communication with Bob (B)
- There's a Trusted Third Party (C) with pre established symmetric keys
- K_{AC} is a symmetric key known only to A and the Key Distribution Center (C)
  - K_{BC} is a symmetric key known only to B and C
- The server generates $K_{AB,}$ a symmetric key used in the session between A and B
  - Every time Alice wants to talk to Bob, a new symmetric K^{AB} key is provided


K_{AB}

Needham-Schroeder Flow

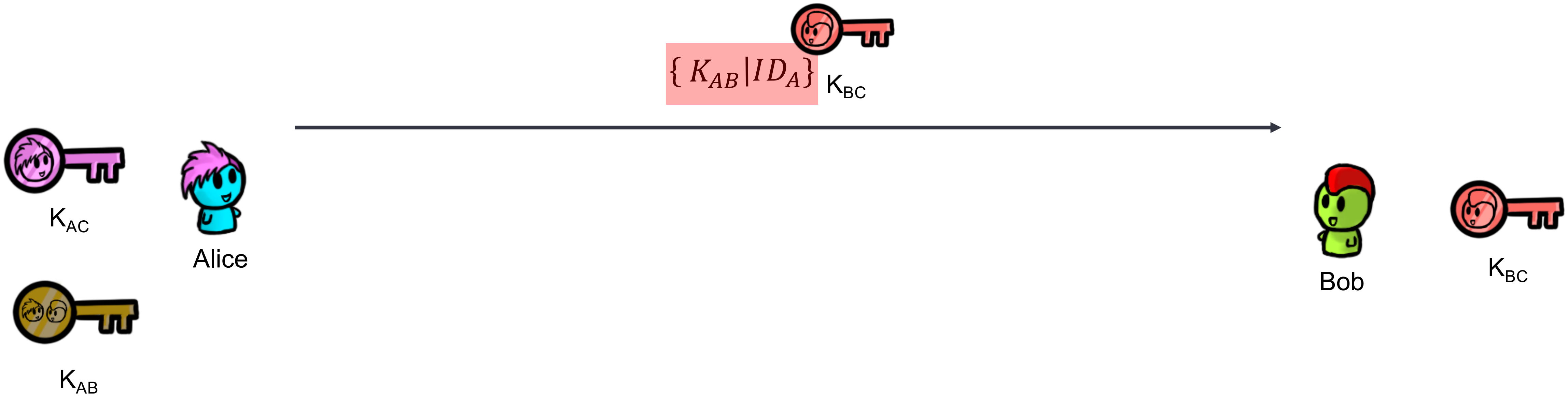


Breaking Down Needham-Schroeder - Step 1



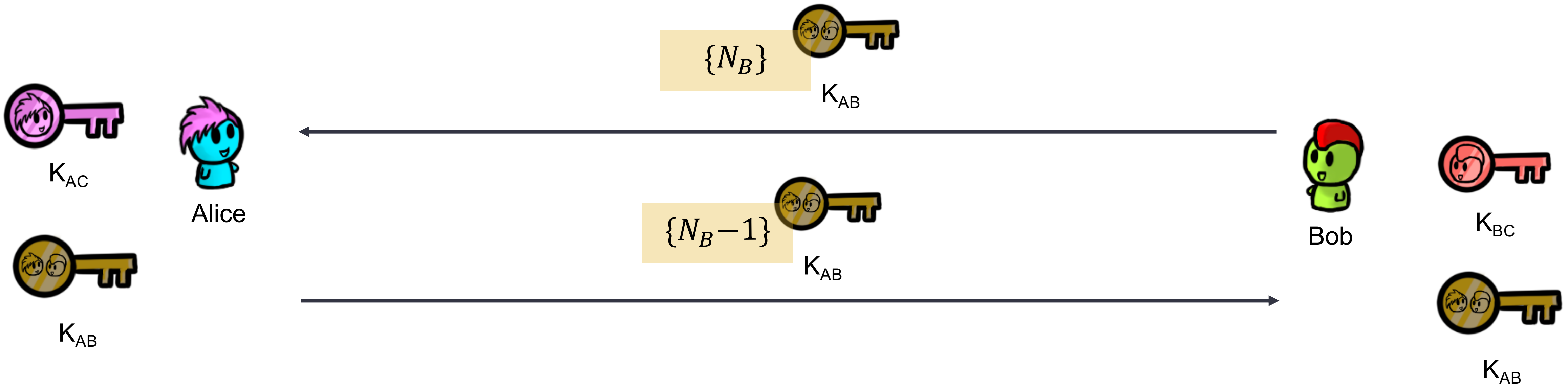
- First message in plaintext – Identifies Alice and Bob
- N_A is a nonce used to prevent reply attacks against Alice

Breaking Down Needham-Schroeder - Step 2



- Simply forward the encrypted K_{AB} to Bob

Breaking Down Needham-Schroeder - Step 3

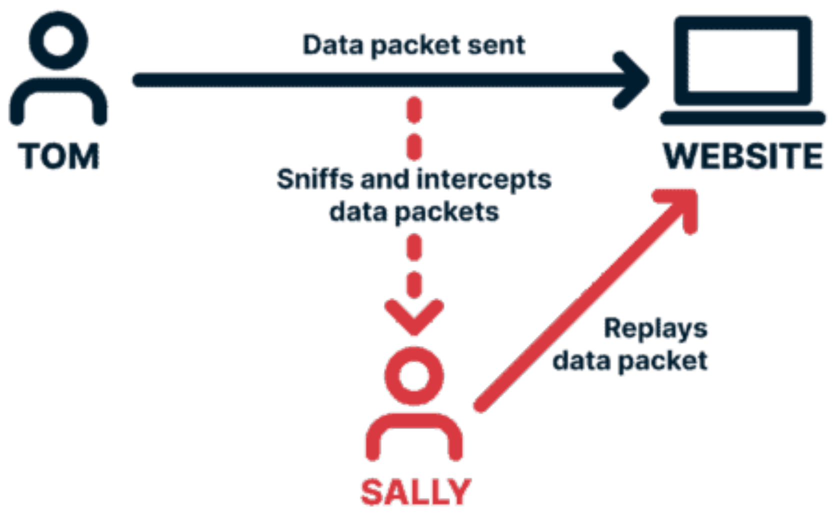


- Need to verify the keys
  - Bob challenges Alice to prove she knows K_{AB}
  - Remember that K_{AB} has been setup by the trusted 3rd party

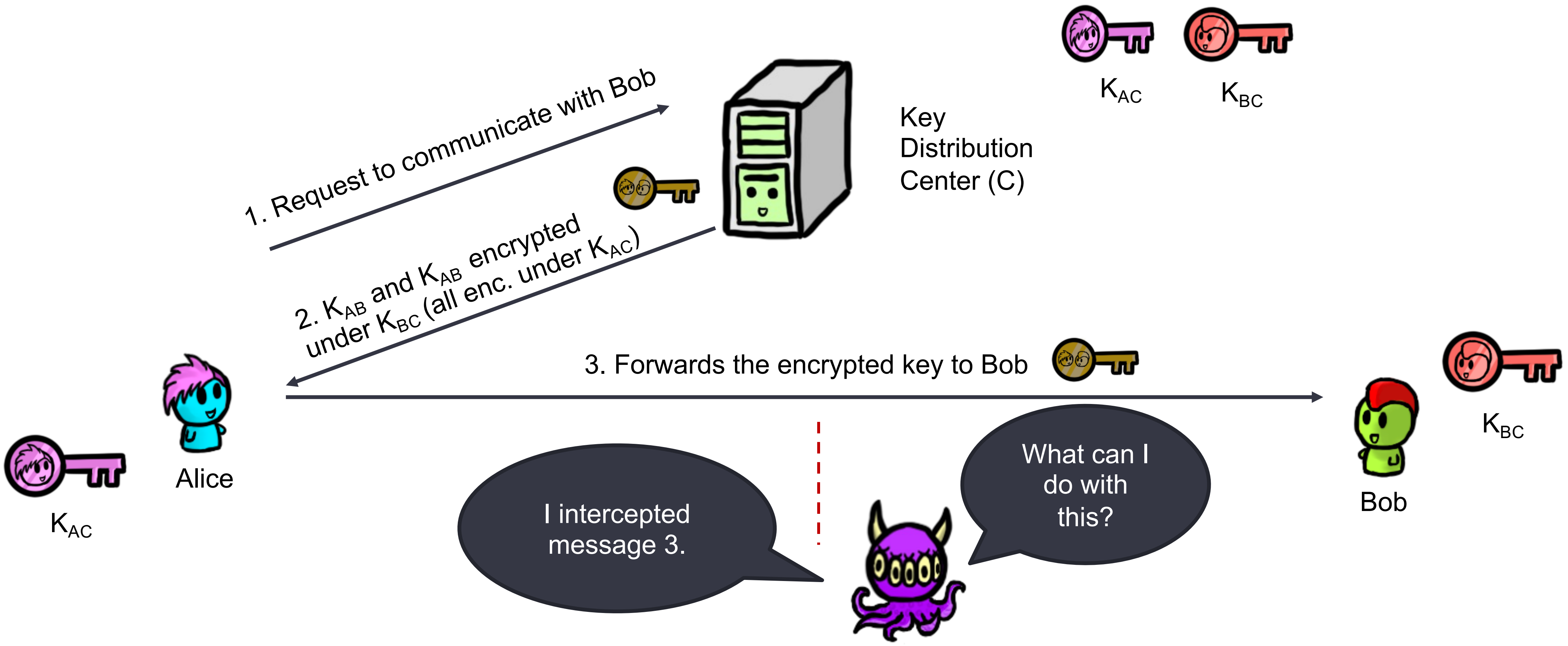
Is Needham-Schroeder Vulnerable to Replay Attacks?

- Replay attack:
  - Mallory intercepts a message meant for some other party
  - They later send this message again pretending to be some other party

- Example
  - Hashed password
  - Car unlocking



Yes, it is ☹



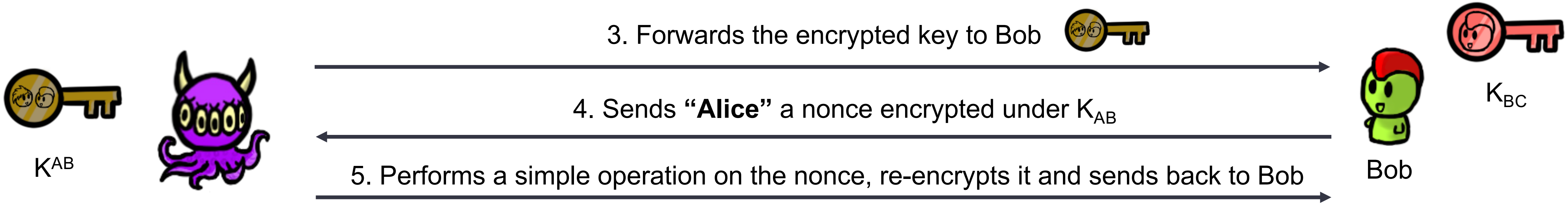
1. Request to communicate with Bob
2. K_AB and K_AB encrypted under K_BC (all enc. under K_AC)
Key Distribution Center (C)
K_AC
K_BC
3. Forwards the encrypted key to Bob
Alice
K_AC
Bob
K_BC
I intercepted message 3.
What can I do with this?

Needham-Schroeder is vulnerable to replay attacks

- 3 weeks later…

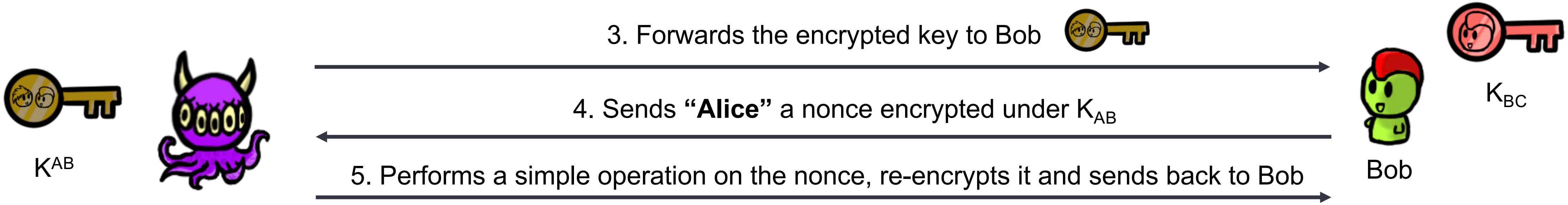
Needham-Schroeder is vulnerable to replay attacks

- 3 weeks later…



Needham-Schroeder is vulnerable to replay attacks

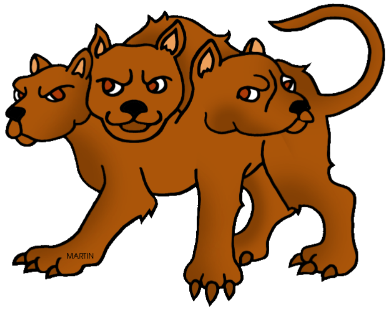
- 3 weeks later…



Bob will believe he is talking to Alice.

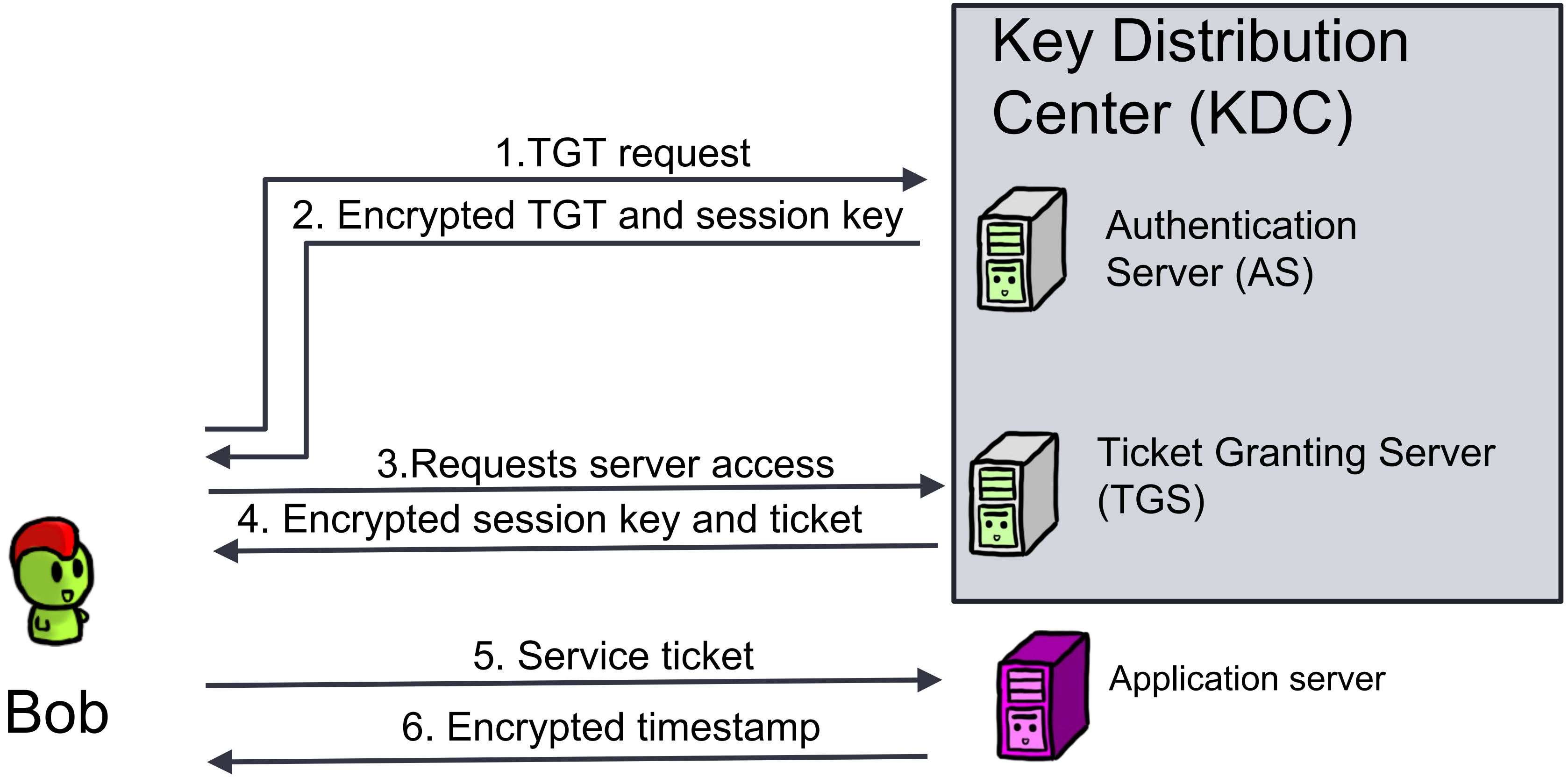
Kerberos

Kerberos

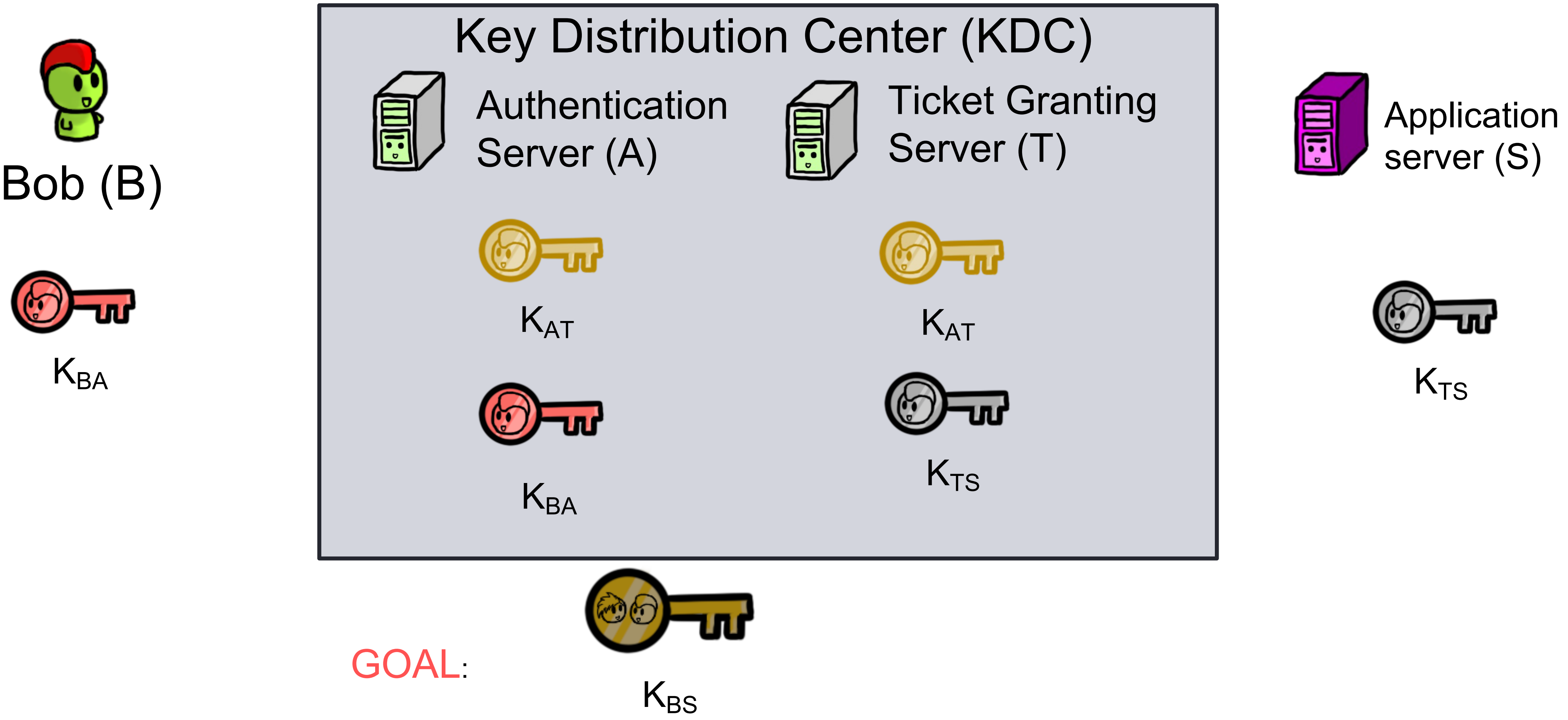


- Based on the Needham-Schroeder protocol
- Fixes the potential for a replay attack vulnerability by adding a timestamp
- Used in Windows Active Directory
- Effective Access Control
    - Each client only needs single key.
    - Each server also only needs a single key.
    - Mutual Authentication.

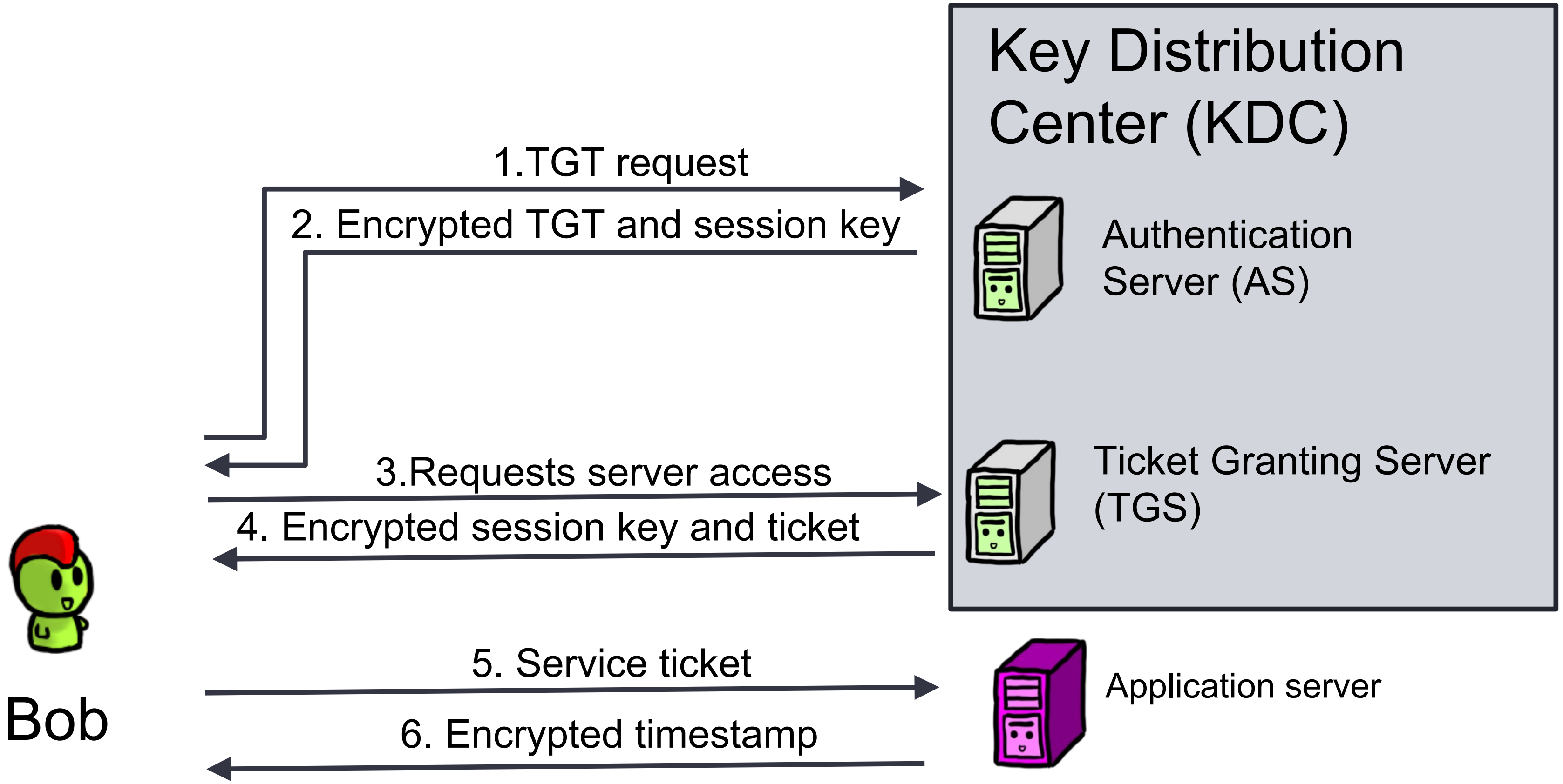
Kerberos Overview



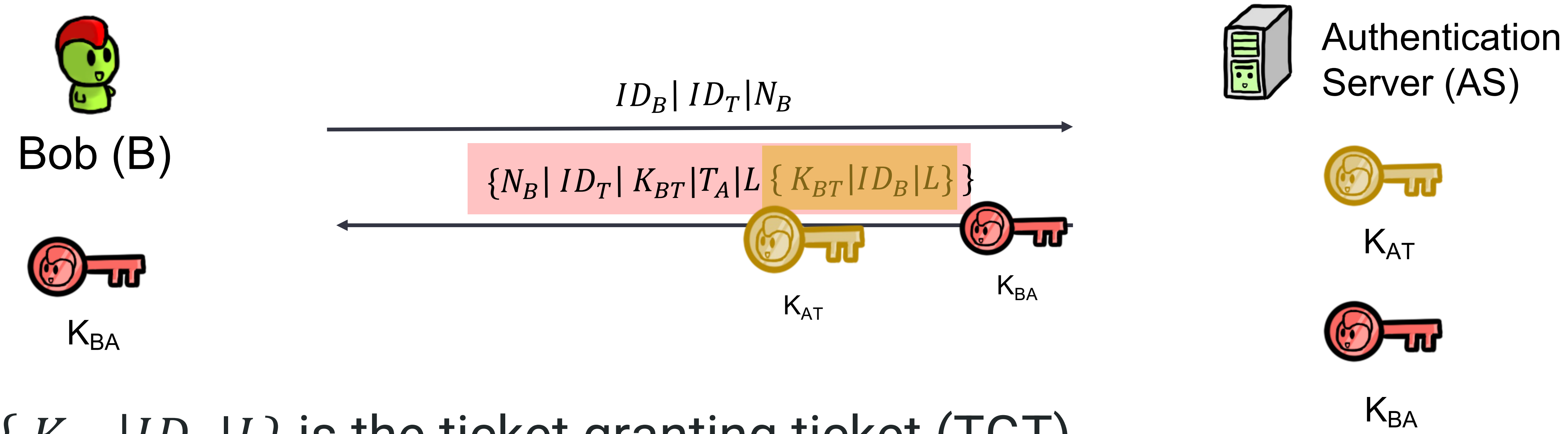
The Keys


Key Distribution Center (KDC)
Bob (B)
Authentication Server (A)
Ticket Granting Server (T)
Application server (S)
K_{BA}
K_{AT}
K_{AT}
K_{BA}
K_{TS}
K_{TS}
GOAL:
K_{BS}

Kerberos Overview

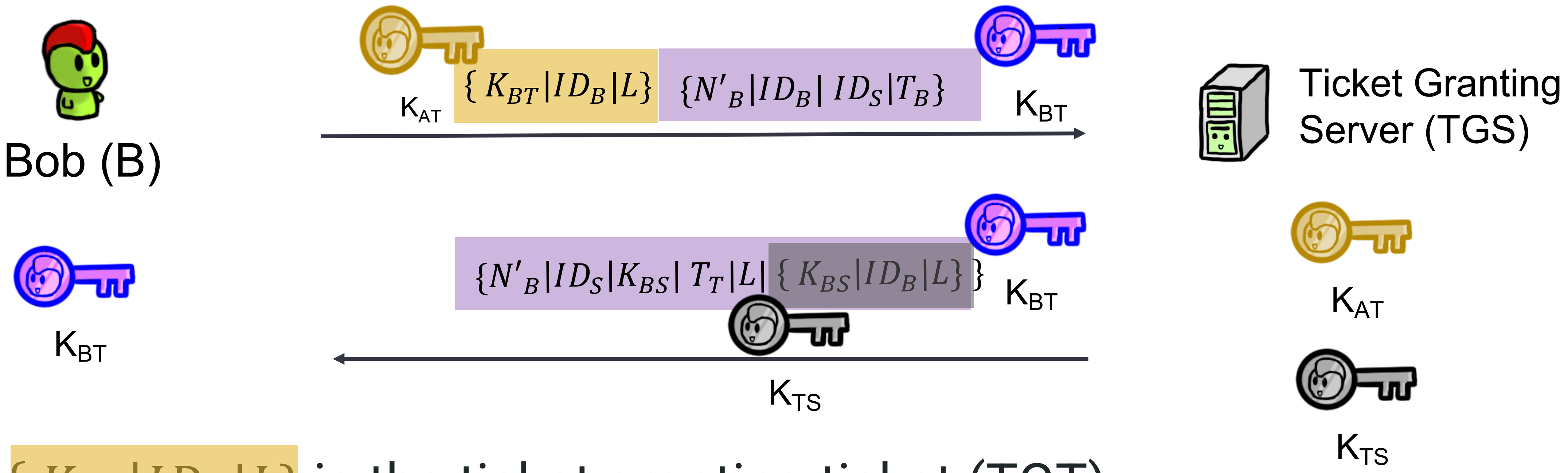


Breaking Down Kerberos – Part 1



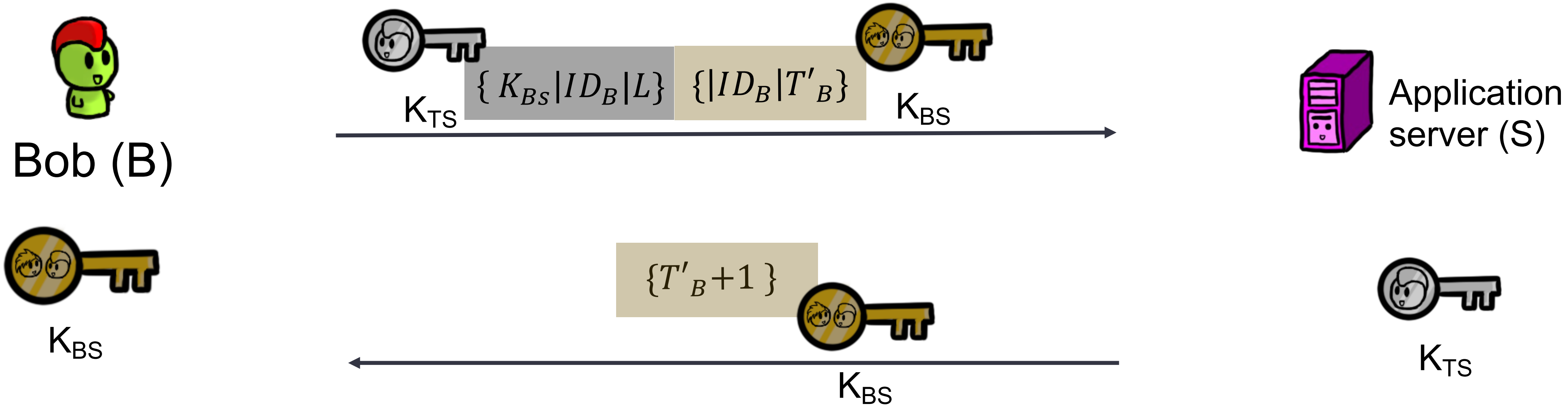
- $\{\,K_{BT}|ID_B|L\}$ is the ticket granting ticket (TGT)
- L is lifetime, T_A is the timestamp at A, N_B is a nonce

Breaking Down Kerberos – Part 2



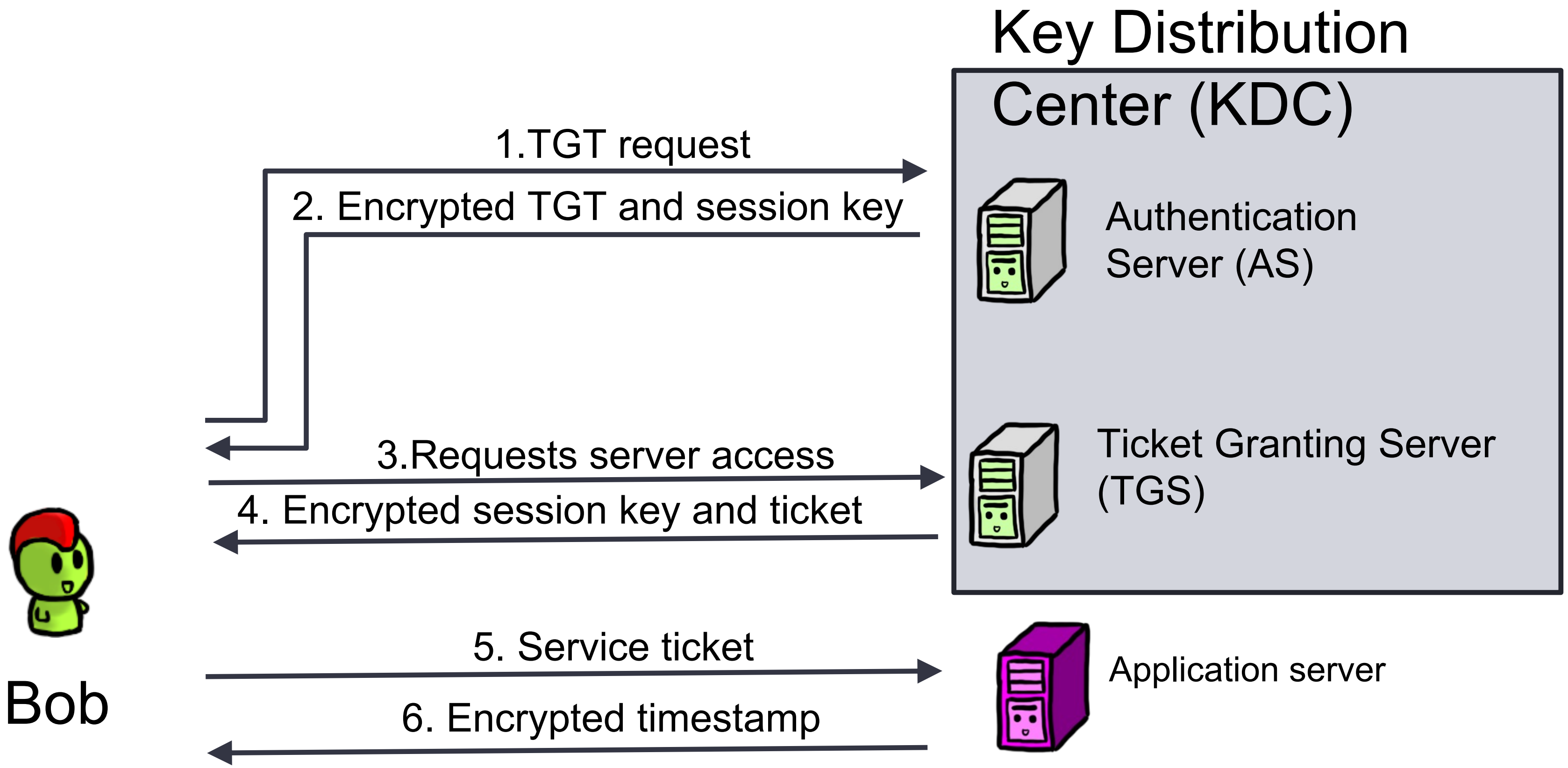
- $\{ K_{BT}|ID_B|L\}$ is the ticket granting ticket (TGT)
- $\{ K_{BS}|ID_B|L\}$ is the service ticket (ST)
- K_{BT} is a session key between Bob and the TGS

Breaking Down Kerberos – Part 3



- $\{ K_{BS}|ID_B|L\}$ is the service ticket (ST)
- K_{BS} is a session key between Bob and the Server

Kerberos Overview

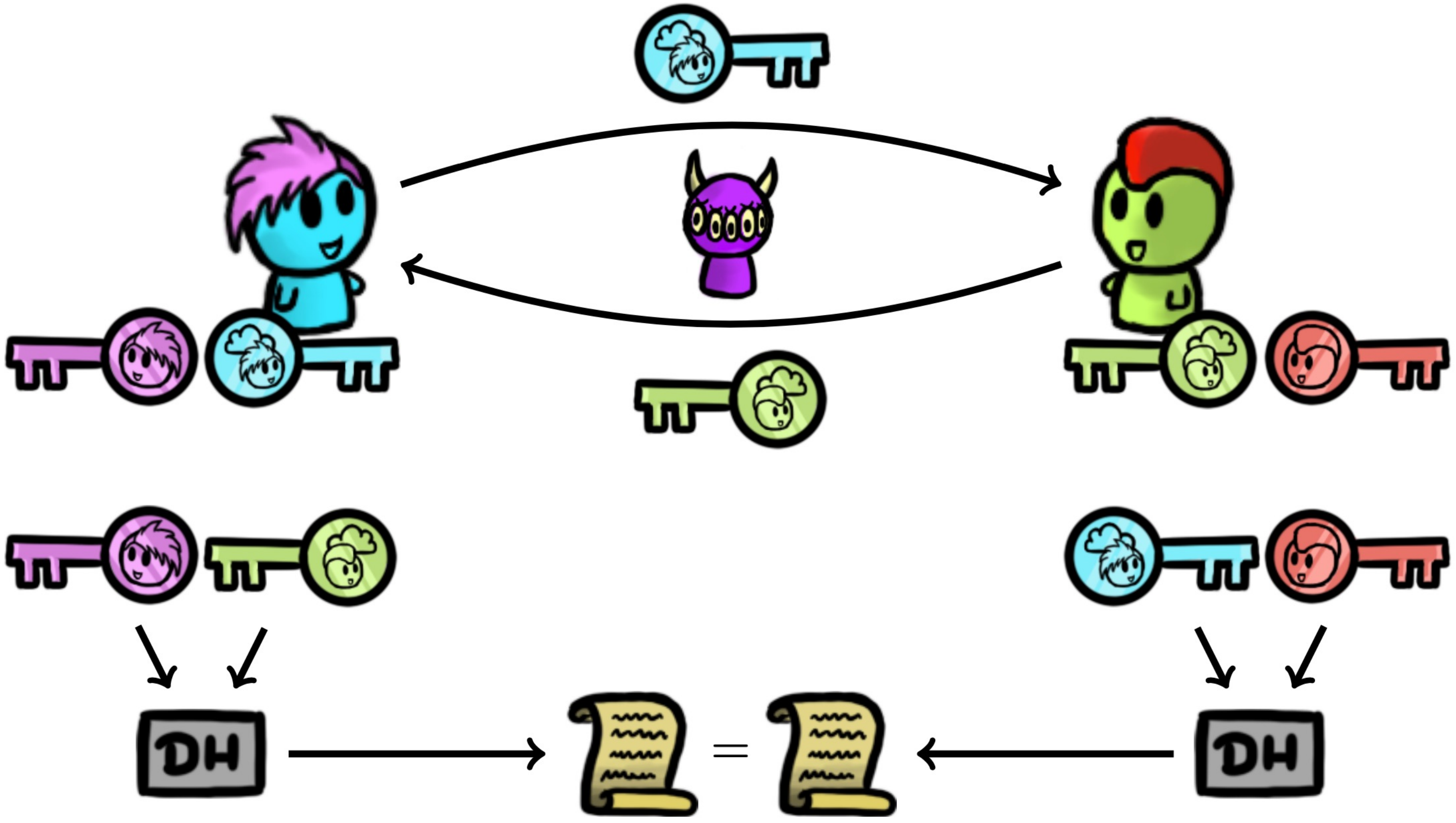


Reflect, why does Kerberos fix it

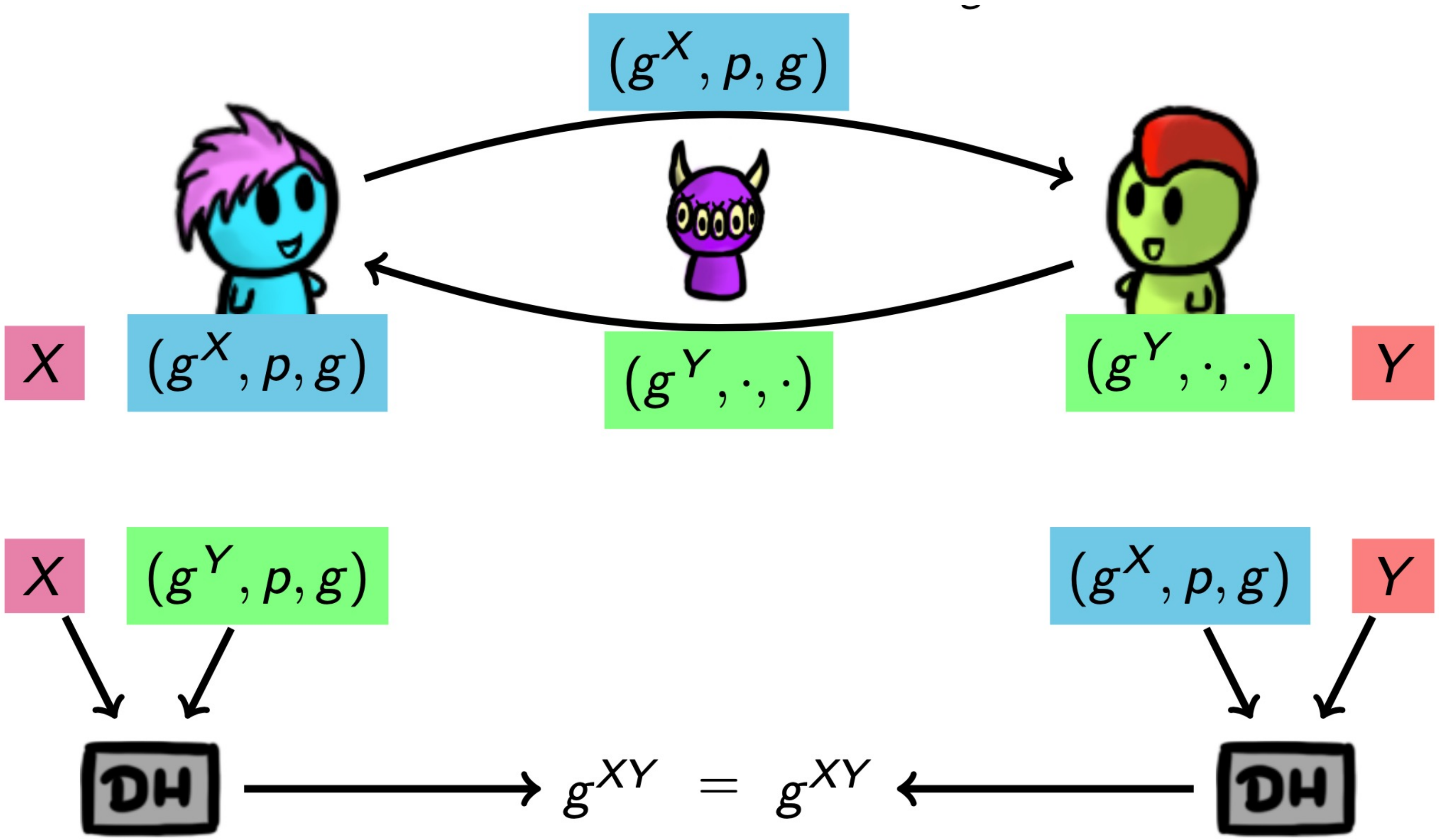
- Timestamps in previously insecure messages
- All tickets include a Lifetime (time they expire)

Asymmetric Crypto Authentication

Recall the Diffie-Hellman key exchange



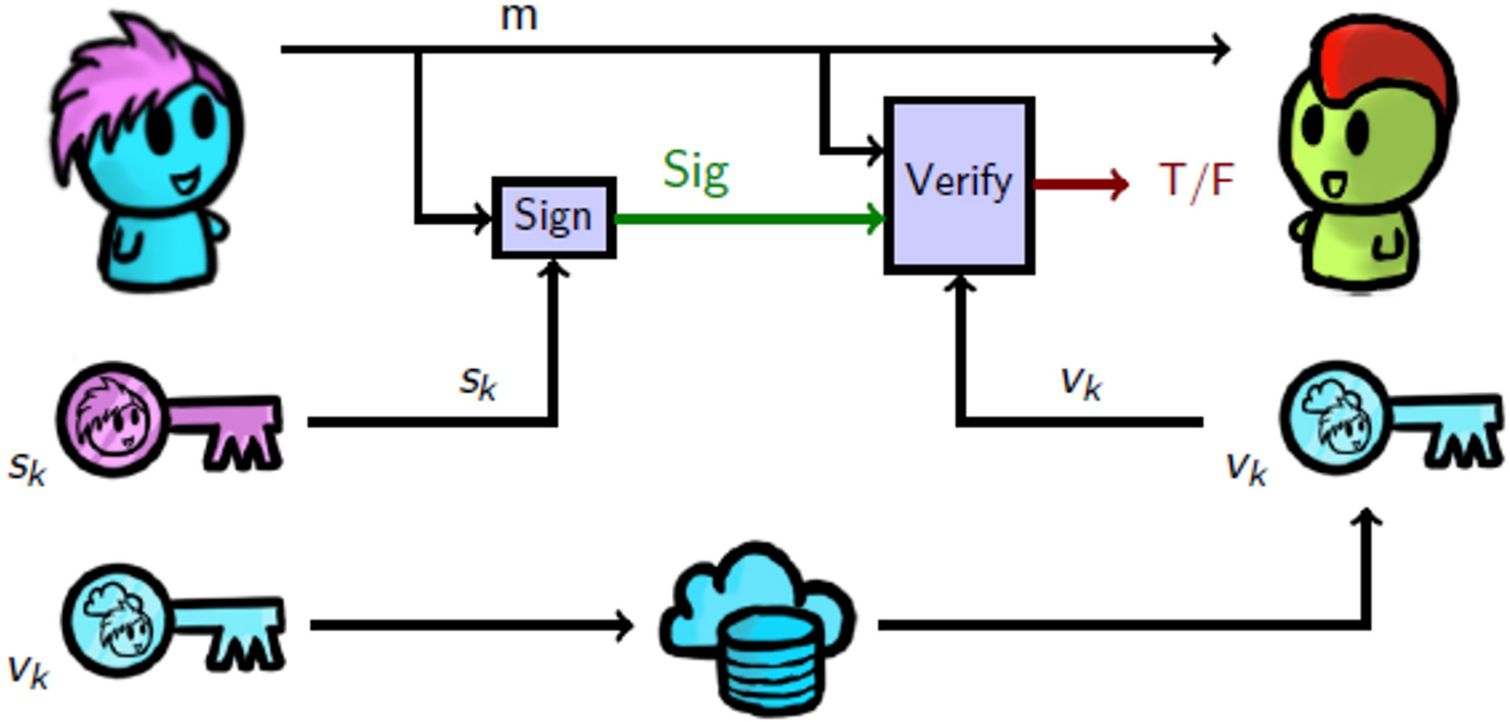
Diffie-Hellman key exchange – Altogether



What’s the Problem!

- Authentication!
- Need to verify the public keys!

Recall, Digital Signatures

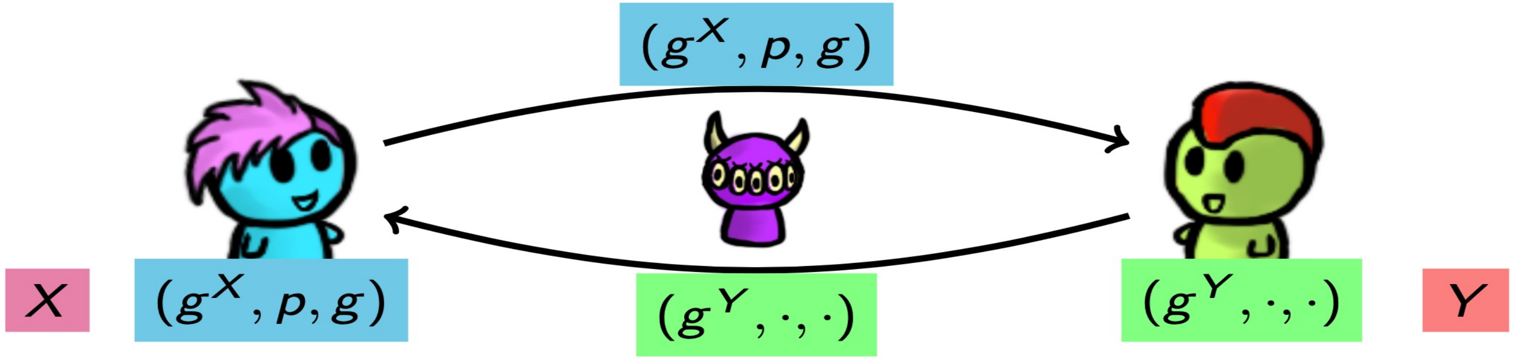


The Key Management Problem

Q: How can Alice and Bob be sure they're talking to each other?

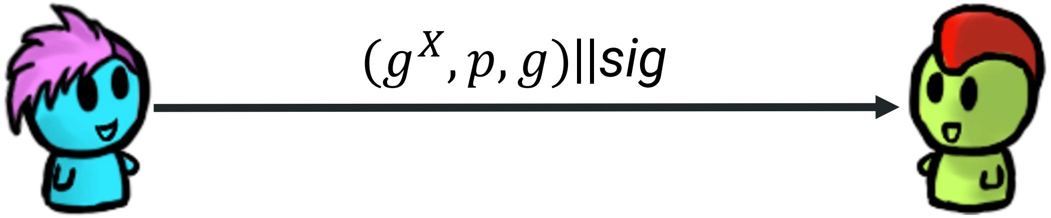
A: By having each other's verification key!

Before



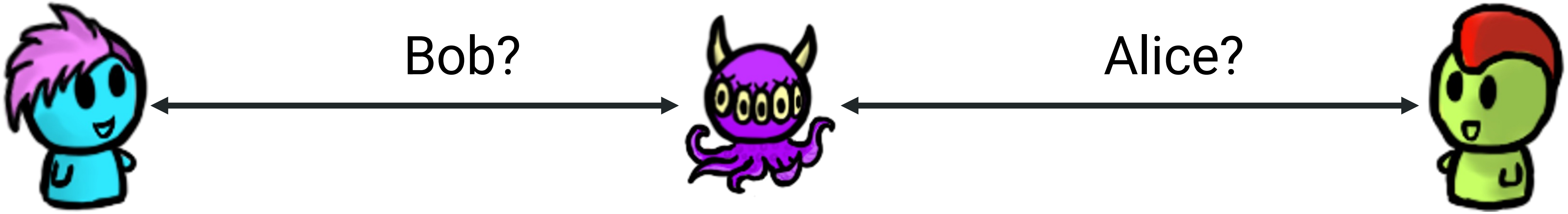
After

sig = $\text{Sign}_{sk}((g^X, p, g))$



$\text{Verify}_{vk}(sig, (g^X, p, g))$?

The Key Management Problem

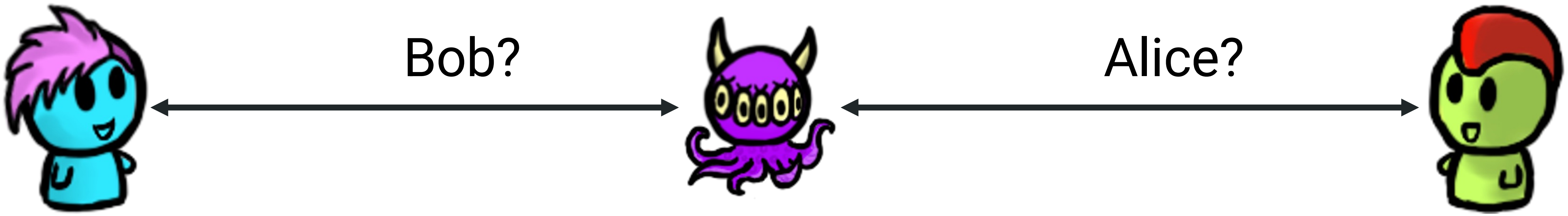


Q: How can Alice and Bob be sure they're talking to each other?

A: By having each other's verification key!

Q: But how do they get the keys…

The Key Management Problem…Solutions?



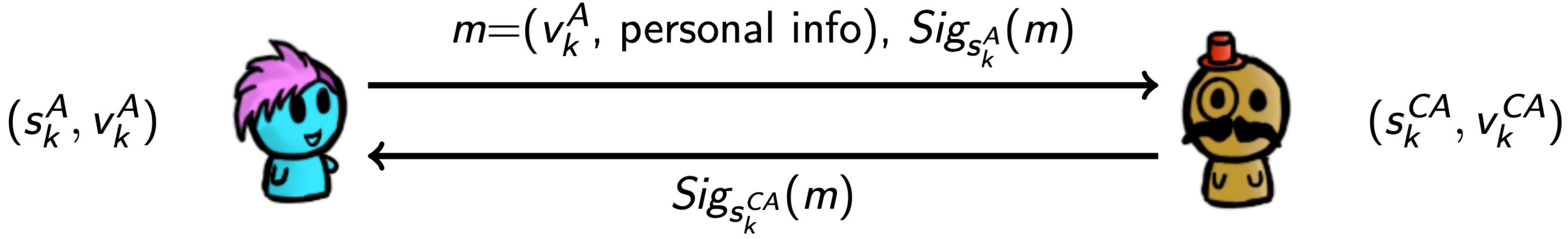
Q: But how do they get the keys…

A: Know it personally (manual keying e.g., SSH)

A: Trust a friend (web of trust e.g, PGP)

A: Trust some third party to tell them (CAs, e.g., TLS/SSL)

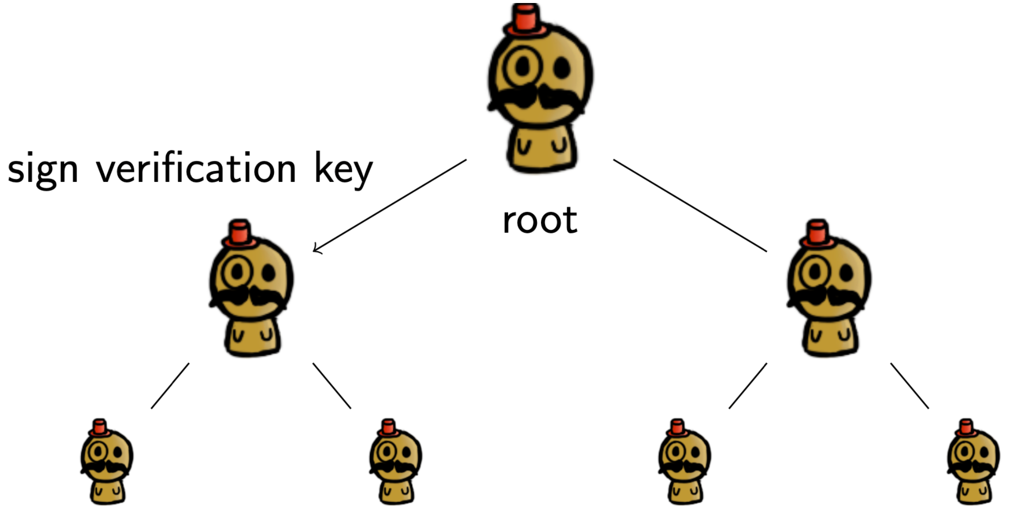
Certificate Authorities (CAs)



- A CA is a trusted third party who keeps a directory of people's (and organizations') verification keys
- Alice generates a (s_k^A, v_k^A) key pair, and sends the verification key and personal information, both signed with Alice's signature key, to the CA
- The CA ensures that the personal information and Alice's signature are correct
- The CA generates a certificate consisting of Alice's personal information, as well as her verification key. The entire certificate is signed with the CA's signature key

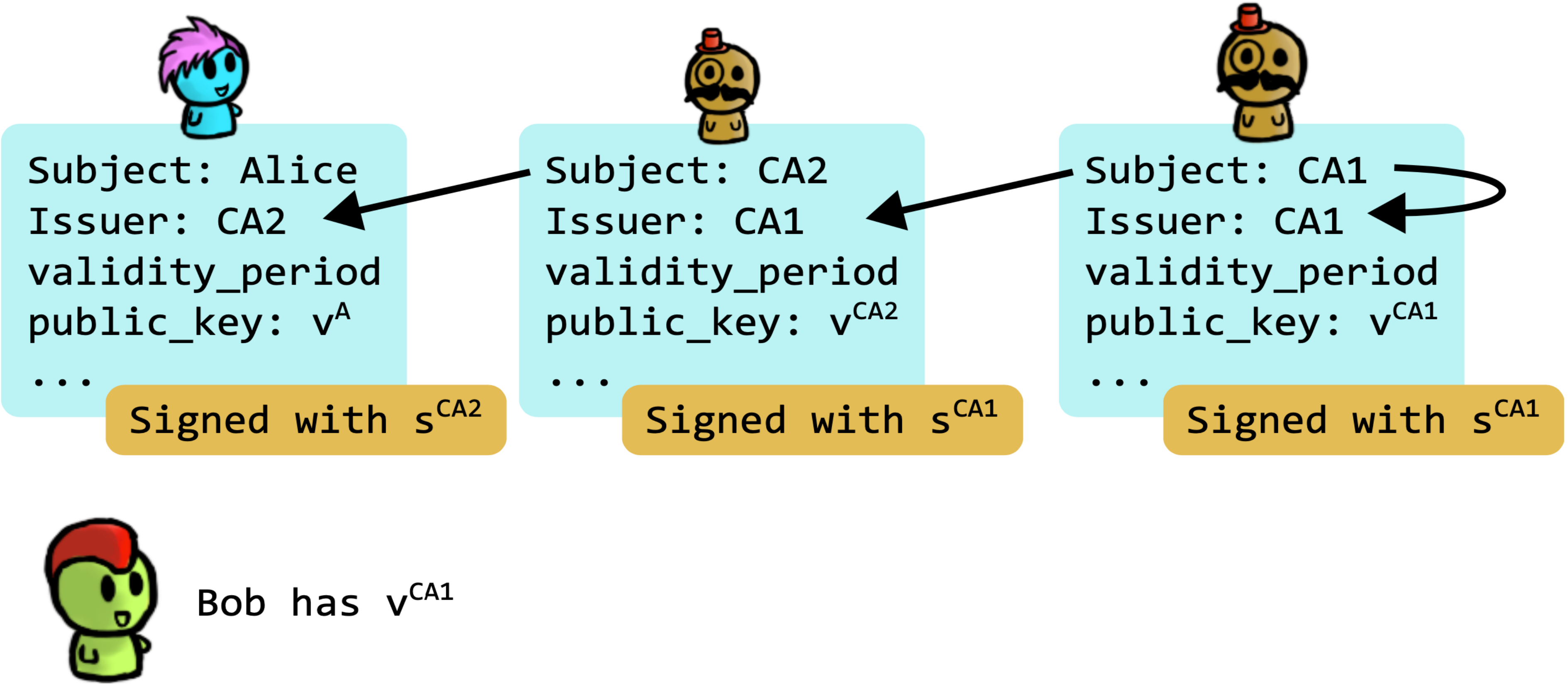
Certificate Authorities

- Everyone is assumed to have a copy of the CA's verification key (s_k^{CA}), so they can verify the signature on the certificate
- There can be multiple levels of certificate authorities; level n CA issues certificates for level n+1 CAs – Public-key infrastructure (PKI)
- Need to have only verification key of root CA to verify the certificate chain



Chain of Certificates

Alice sends Bob the following certificate to prove her identity. Bob can follow the chain of certificates to validate Alice's identity.

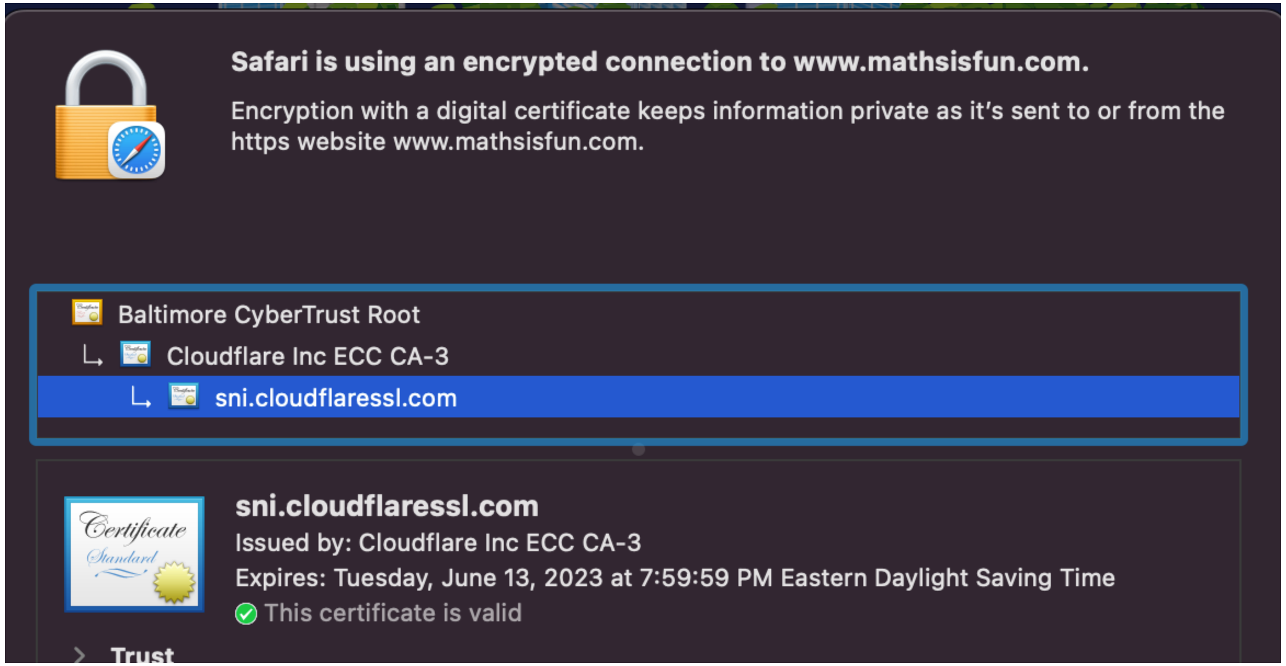


CAs on the web

- Root verification key installed on browser
- https://letsencrypt.org changed the game by offering free certificates
- Other common CAs:

Rank	Issuer	Usage	Market Share
1	IdenTrust	38.5%	43.6%
2	DigiCert Group	13.1%	14.5%
3	Sectigo (Comodo Cybersecurity)	12.1%	13.4%
4	GlobalSign	16.1%	16.7%
5	Let's Encrypt	5.8%	6.4%
6	GoDaddy Group	4.8%	5.3%

Examples


Safari is using an encrypted connection to www.mathsisfun.com.
Encryption with a digital certificate keeps information private as it's sent to or from the https website www.mathsisfun.com.
Baltimore CyberTrust Root
Cloudflare Inc ECC CA-3
sni.cloudflaressl.com
sni.cloudflaressl.com
Issued by: Cloudflare Inc ECC CA-3
Expires: Tuesday, June 13, 2023 at 7:59:59 PM Eastern Daylight Saving Time
This certificate is valid
Trust

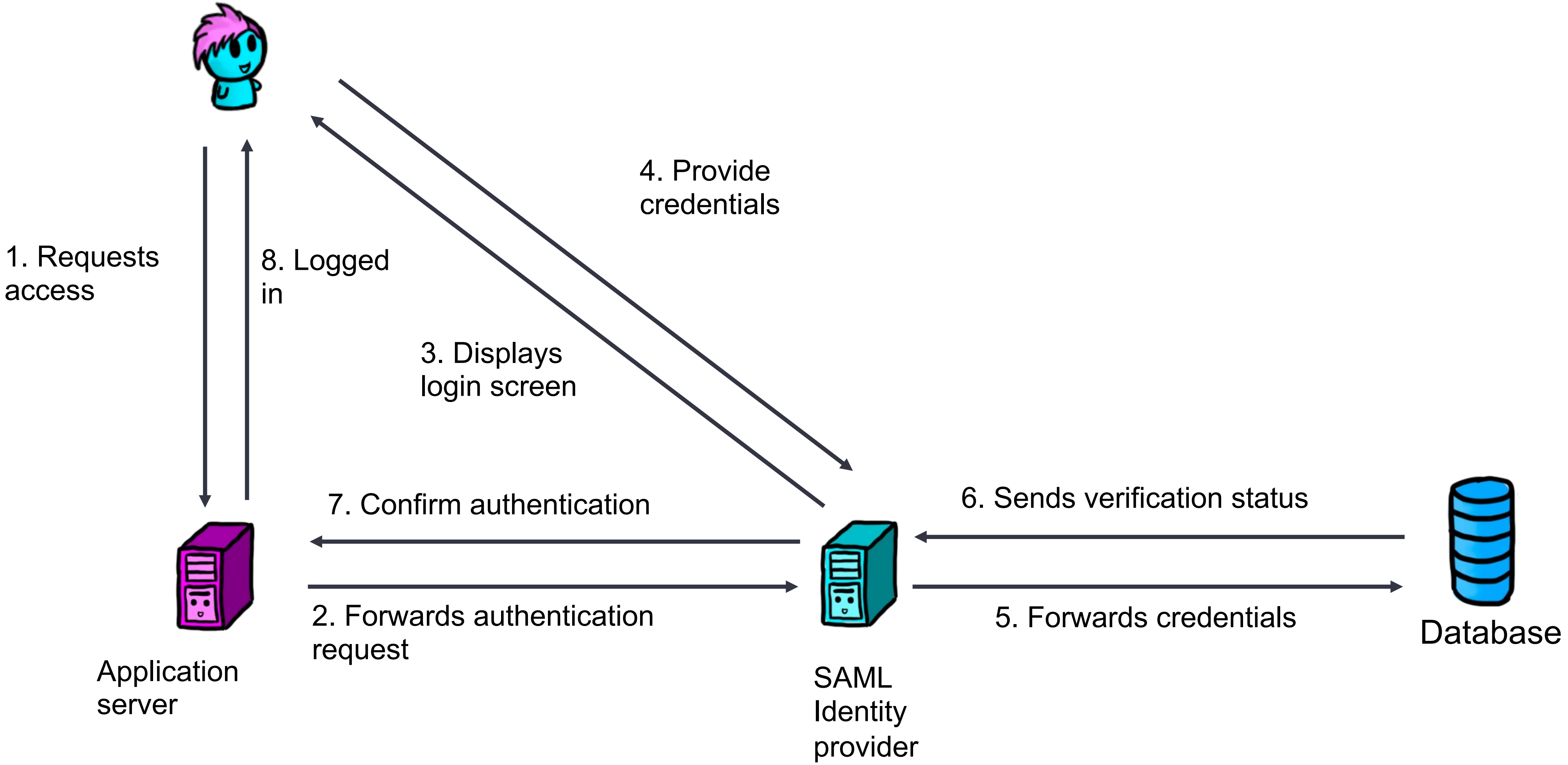

Details
Subject Name
Country or Region US
State/Province California
Locality San Francisco
Organization Cloudflare, Inc.
Common Name sni.cloudflaressl.com
Issuer Name
Country or Region US
Organization Cloudflare, Inc.
Common Name Cloudflare Inc ECC CA-3
Serial Number 0D 62 A9 13 F8 92 16 F7 74 7D 82 56 83 B4 C1 93
Version 3
Signature Algorithm ECDSA Signature with SHA-256 (1.2.840.10045.4.3.2)
Parameters None
Not Valid Before Sunday, June 12, 2022 at 8:00:00 PM Eastern Daylight Saving Time
Not Valid After Tuesday, June 13, 2023 at 7:59:59 PM Eastern Daylight Saving Time
Public Key Info
Algorithm Elliptic Curve Public Key (1.2.840.10045.2.1)
Parameters Elliptic Curve secp256r1 (1.2.840.10045.3.1.7)
Public Key 65 bytes : 04 74 C2 77 87 04 8D BD E0 C7 C8 8B CF 13 B8 F5 18 40 7E 98 1F C2 F7 9E 4A 66 23 5E C8 C8 93 33 75 CC C2 ED 56 1F AB DA 31 D5 5D 1A AB 39 60 9B 2B E9 91 02 62 8C B2 4D 28 F4 91 07 A8 26 01 44 2D
Key Size 256 bits
Key Usage Encrypt, Verify, Derive
Signature 70 bytes : 30 44 02 20 7A 62 4A 32 ...

Key Management - SSO

Security Assertion Markup Language (SAML)

- Uses secure tokens (encrypted, digitally signed XML-certificates) instead of credentials
- Allows users to access multiple applications with trusted information with a single log in - single sign-on (SSO)
- Can use whatever authentication protocol you choose
- Primarily a standard for how these communications are formatted

Security Assertion Markup Language (SAML)



Security Assertion Markup Language (SAML)

- Advantages:
  - Authentication is centralized
  - Loose coupling of directories
  - User errors such as forgotten, weak or leaked password are avoided
  - Improves user experience (single-sign on for multiple applications)
  - XML-based protocol
    - Widely used and known
- Disadvantages
  - Complex to implement
    - Errors
    - Lengthened timelines
  - If down, can remove access from multiple systems

DNSSEC

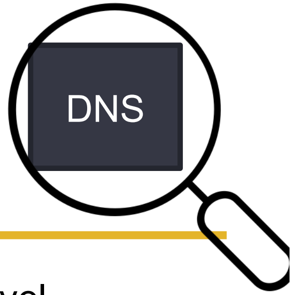
Source: Jason Goertzen and Miti Mazmudar

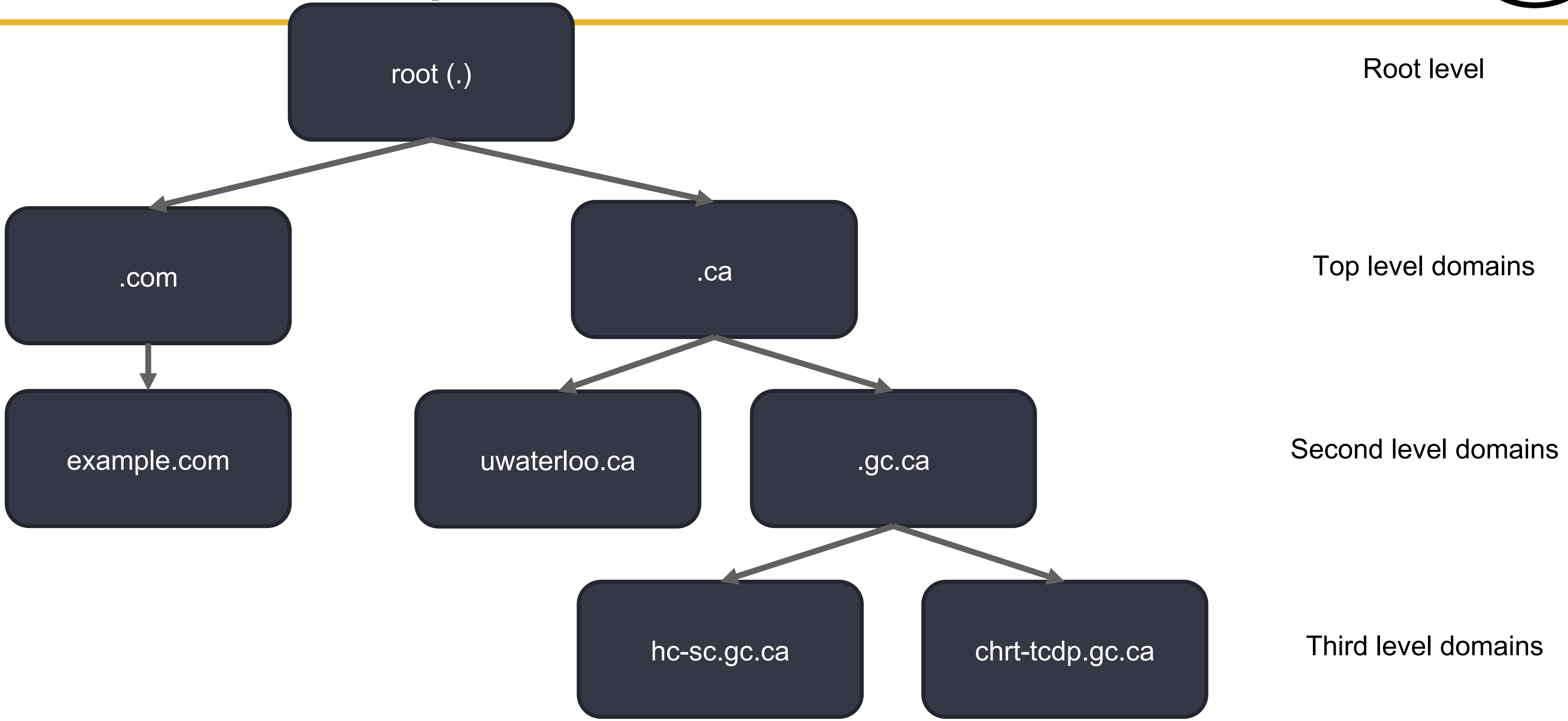
Recall, what is DNS?

- The internet uses IP addresses to determine where to send messages
- IP addresses are difficult for people to remember!
- The Domain Name System is responsible to translating something easy for a human to remember into IP addresses

```
example.com -> 93.184.216.34
```

DNS is broken up into zones





Domain Name System (DNS) - *dig* command

```
; <<>> DiG 9.16.15 <<>> crysp.uwaterloo.ca
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34154
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1280
;; QUESTION SECTION:
;crysp.uwaterloo.ca.            IN      A

;; ANSWER SECTION:
crysp.uwaterloo.ca.     4552    IN      A       129.97.167.73

;; Query time: 0 msec
;; SERVER: 192.168.0.1#53(192.168.0.1)
;; WHEN: Wed May 19 15:10:46 EDT 2021
;; MSG SIZE  rcvd: 63
```

`dig crysp.uwaterloo.ca`

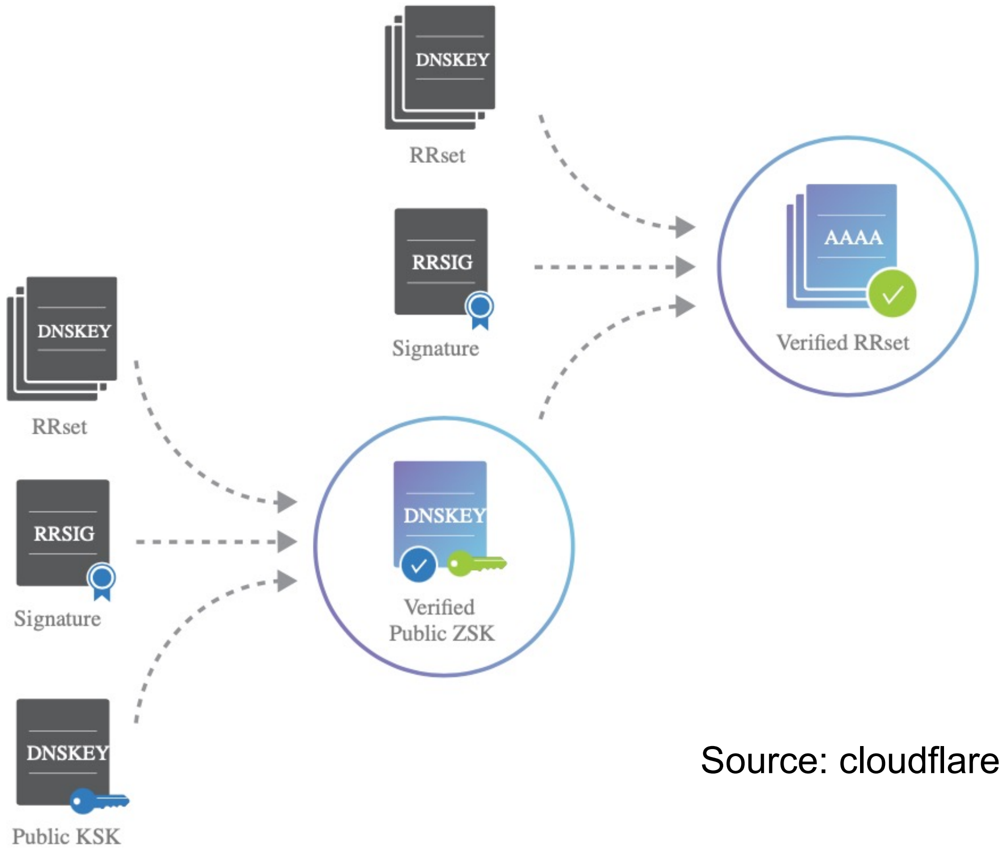
Securing DNS

Use **digital signatures** to make sure a correct and unmodified message is received from the correct entity!

- New records added to DNSSEC signed zone
- Record sets (RRSets) are signed, instead of individual records
- Have two keys:
  - **Key Signing Key (KSK):** kept in trusted hardware, hard to change
  - **Zone Signing Key (ZSK):** changed more often, smaller, used for records

Verification Procedure

- Assume you trust the public **KSK** held by a “trust anchor”
- Use it to verify the RRset containing a given **ZSK**
- Then use **ZSK** to verify the records



Source: cloudflare blog

How do we maintain key integrity?

Construct a chain of trust!

- The root verification **KSK** must be manually configurated on the machine making the request
- When the root **ZSK** is queried use the trust anchor to verify key and its signature (https://www.cloudflare.com/dns/dnssec/root-signing-ceremony/) 
- Each zone's parent zone contains a "Delegate signer" (DS) record which is used to verify zone's **KSK**
  - Hash of **KSK**

The verification process

- **Light blue:** Because of our trust anchor, we trust the KSK of the root (1). The root's KSK signs its ZSK, so now we trust the root's ZSK (2-3).
- **Dark blue:** We trust the root's ZSK. The root's ZSK signs .edu's KSK (4-5), so now we trust .edu's KSK.
- **Light green:** We trust the .edu's KSK (6). .edu's KSK signs .edu's ZSK, so now we trust .edu's ZSK (7-8).
- **Dark green:** We trust .edu's ZSK. .edu's ZSK signs berkeley.edu's KSK (9-10), so now we trust berkeley.edu's KSK.
- **Light orange:** We trust the berkeley.edu's KSK (11). berkeley.edu's KSK signs berkeley.edu's ZSK, so now we trust berkeley.edu's ZSK (12-13).
- **Dark orange:** We trust berkeley.edu's ZSK. berkeley.edu's ZSK signs the final answer record (14-15), so now we trust the final answer.

https://textbook.cs161.org/network/dnssec.html