CS459/698 Privacy, Cryptography, Network and Data Security

Authentication Protocols

Fall 2025, Tuesday/Thursday 8:30-9:50am

A1 is due today!

- Late policy from today 3pm until Oct 2 3pm.
    - No further help will be provided

Today’s Lecture – Authentication Protocols

- Symmetric Authentication
  - Needham-Schroeder
  - Kerberos
- Asymmetric Authentication (PKI)
  - DH
  - Certificates
- DNSSEC

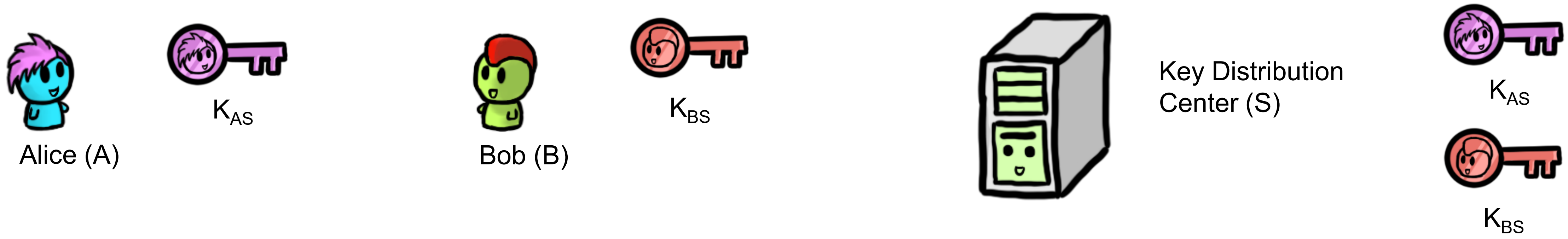
Today’s Focus

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

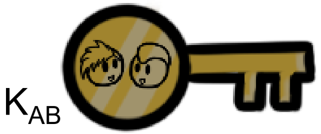
Symmetric Crypto Authentication

Needham-Schroeder

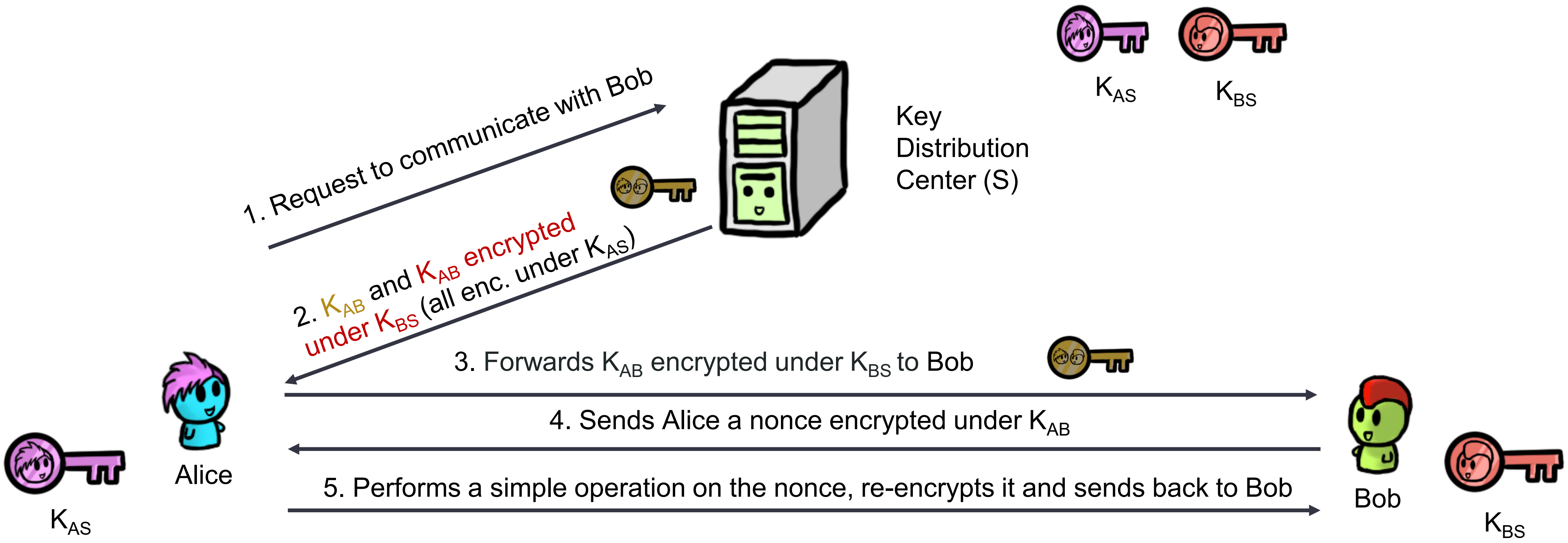
Needham-Schroeder Overview (1978)



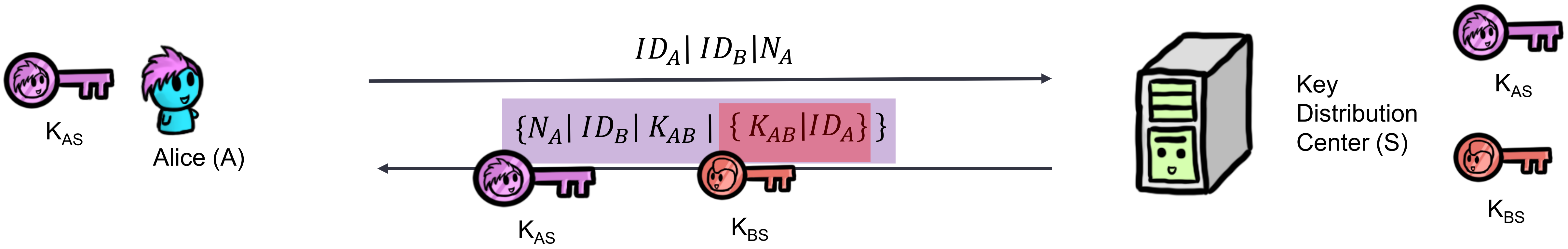
- Alice (A) wants to initiate communication with Bob (B)
- There is a trusted Key Distribution Center (S) with pre-established symmetric keys
- K_{AS} is a symmetric key known only to A and S
  - K_{BS} is a symmetric key known only to B and S
- S generates $K_{AB,}$ a symmetric key used in the session between A and B
  - Every time Alice wants to talk to Bob, a new symmetric K_{AB} key is provided



Needham-Schroeder Flow

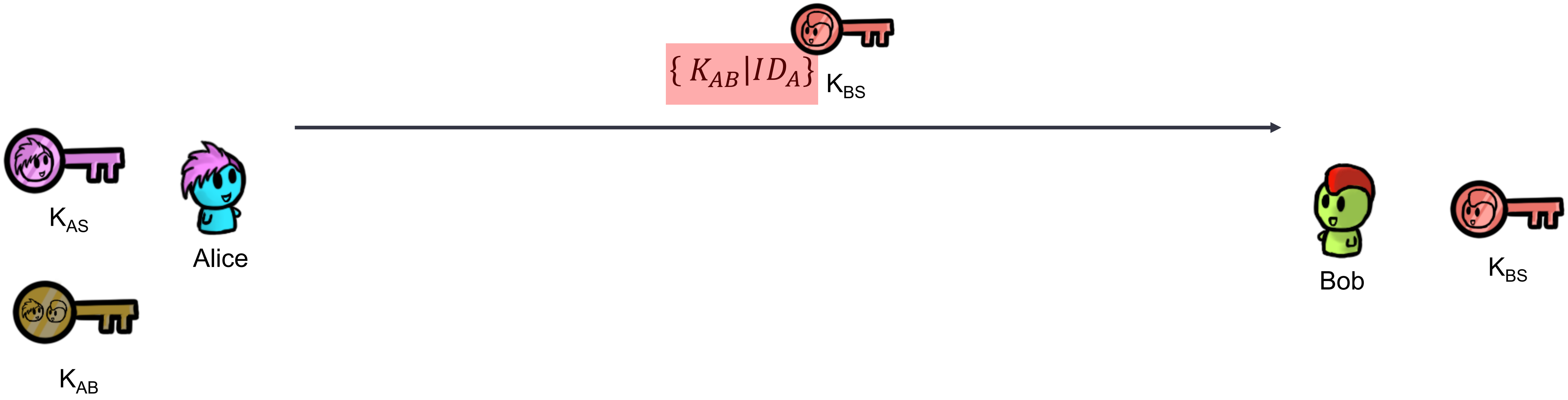


Breaking Down Needham-Schroeder - Step 1



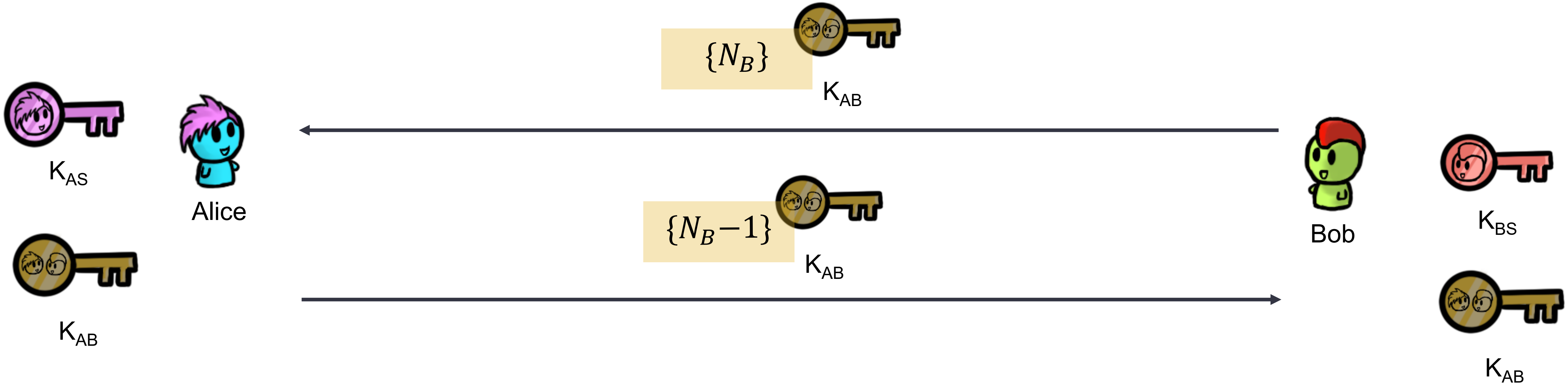
- First message in plaintext – Identifies Alice and Bob
- N_A is a nonce used to prevent reply attacks against Alice

Breaking Down Needham-Schroeder - Step 2



- Simply forward the encrypted K_{AB} to Bob

Breaking Down Needham-Schroeder - Step 3

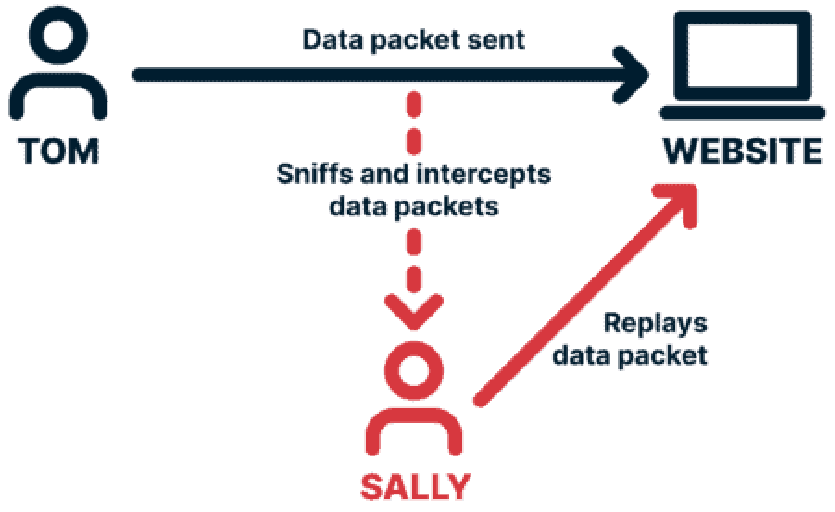


- Need to verify the keys
  - Bob challenges Alice to prove she knows K_{AB}
  - Remember that K_{AB} has been set up by trusted 3rd party S

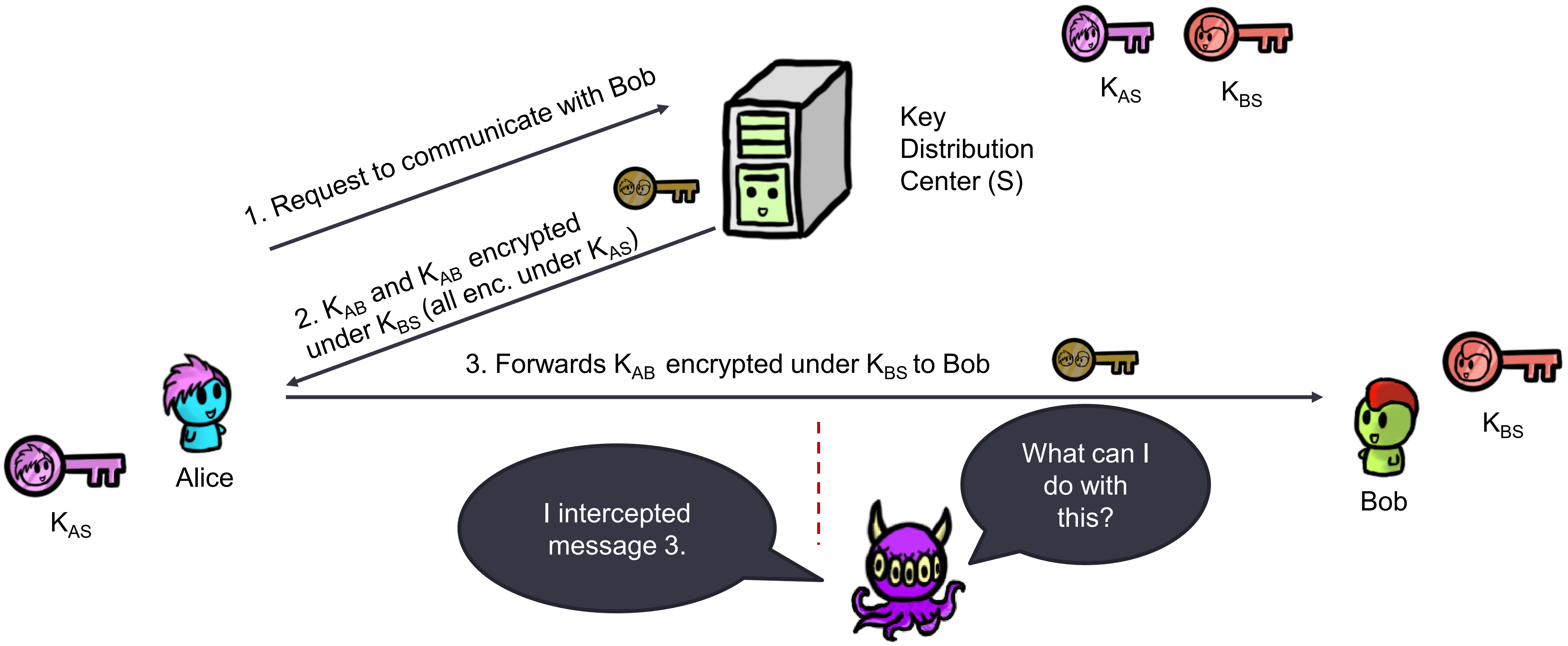
Is Needham-Schroeder Vulnerable to Replay Attacks?

- 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 ☹



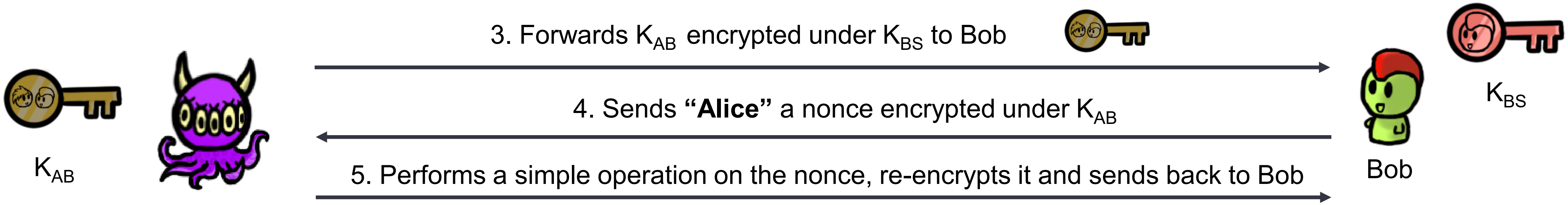
1. Request to communicate with Bob
2. K_{AB} and K_{AB} encrypted under K_{BS} (all enc. under K_{AS})
Key Distribution Center (S)
K_{AS}
K_{BS}
3. Forwards K_{AB} encrypted under K_{BS} to Bob
Alice
K_{AS}
Bob
K_{BS}
I intercepted message 3.
What can I do with this?

Needham-Schroeder is vulnerable to replay attacks

- 3 weeks later…

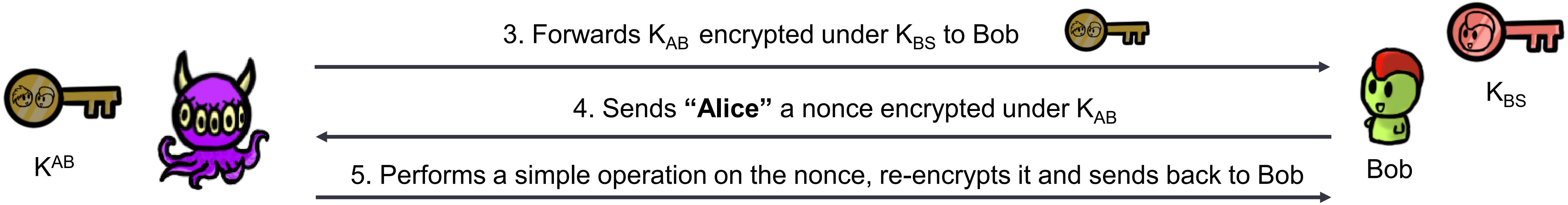
Needham-Schroeder is vulnerable to replay attacks

- 3 weeks late



Needham-Schroeder is vulnerable to replay attacks

- 3 weeks later…

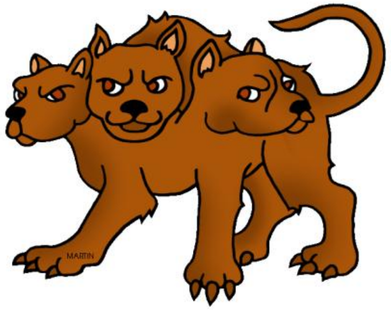


Bob will believe he is talking to Alice.

Symmetric Crypto Authentication

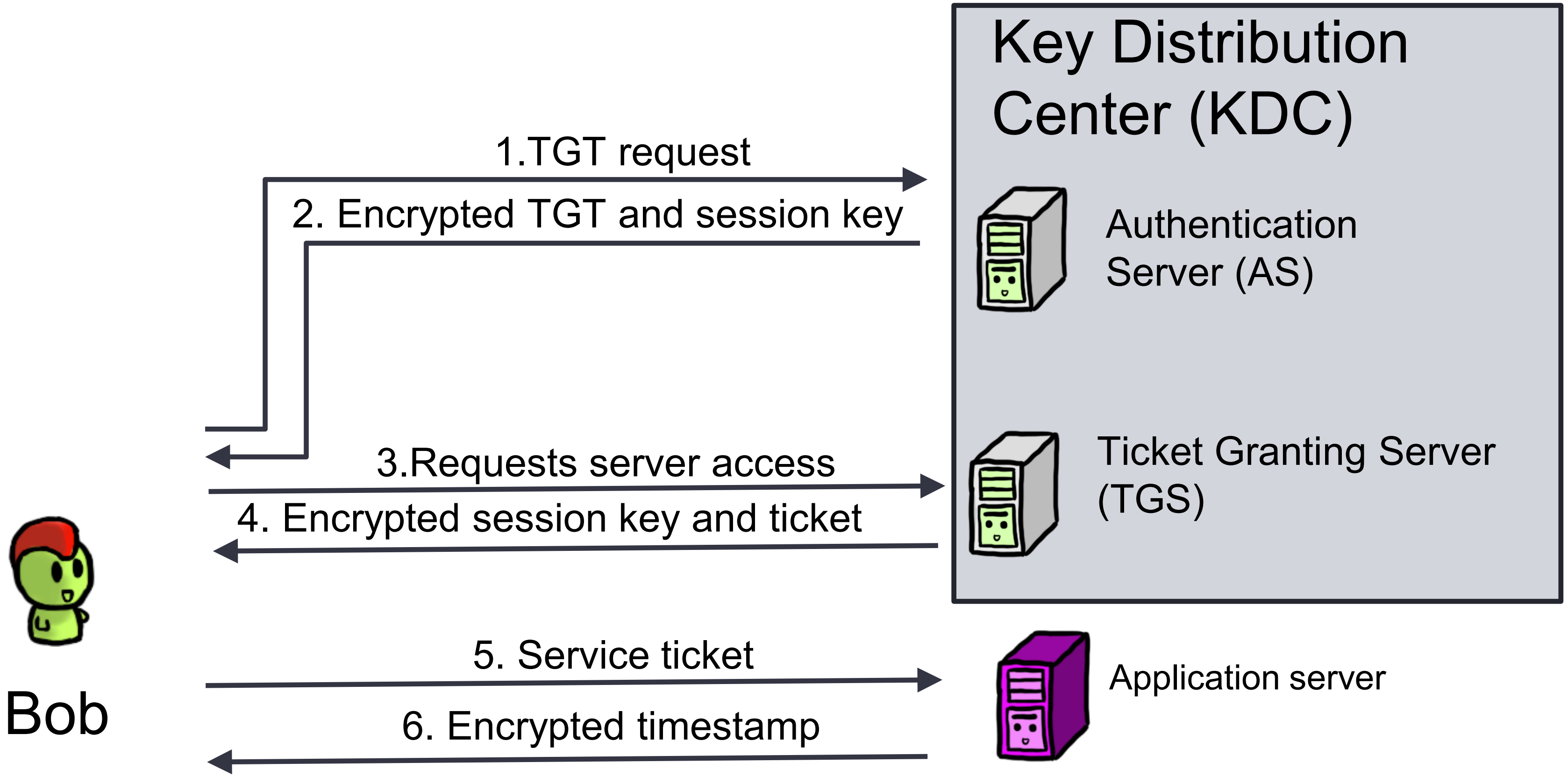
Kerberos

Kerberos (1988)

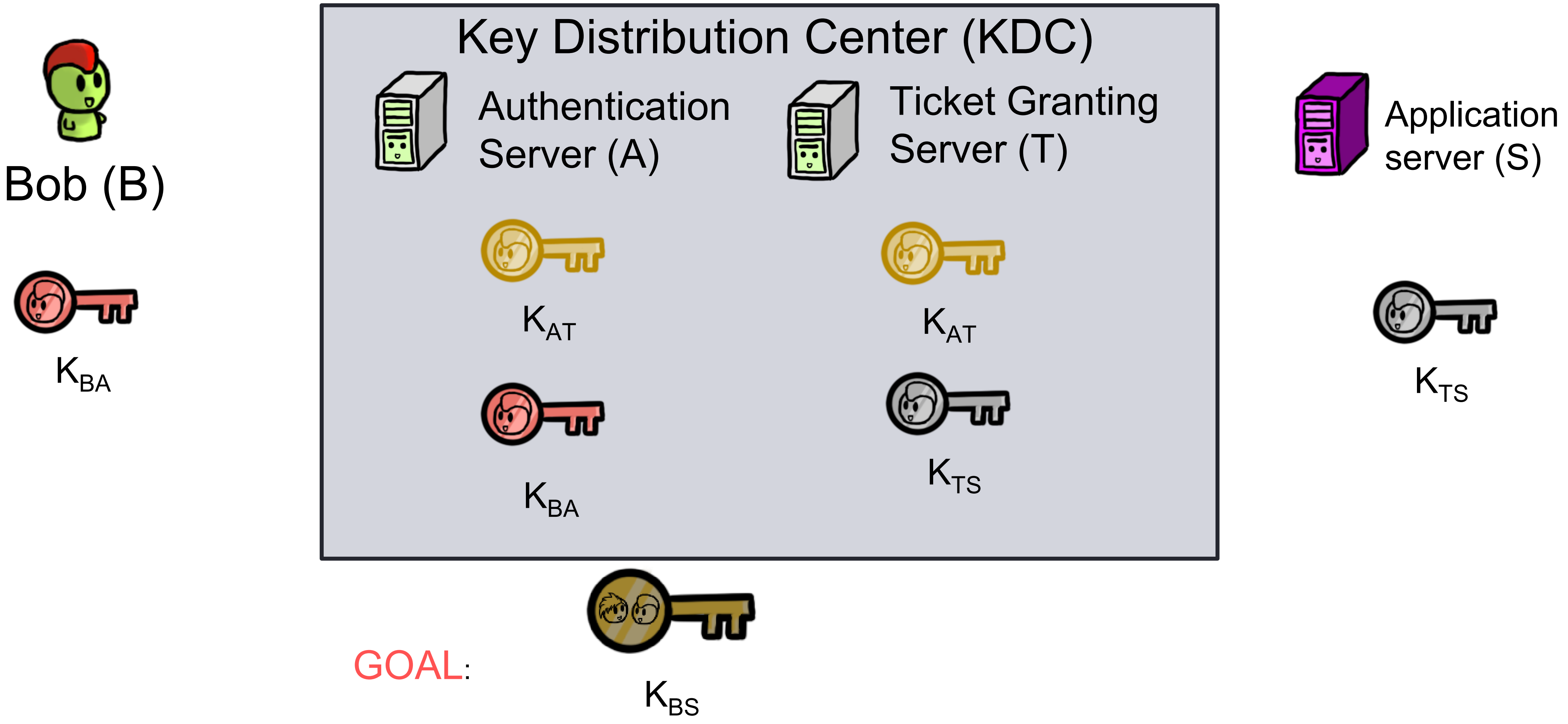


- Based on the Needham-Schroeder protocol
- Fixes the potential for a replay attack
  - By adding a timestamp!
- Used in Windows Active Directory
  - Enables administrators to manage permissions and access to network resources
- Effective Access Control
  - Each client only needs single key.
  - Each server also only needs a single key.
  - Mutual Authentication.

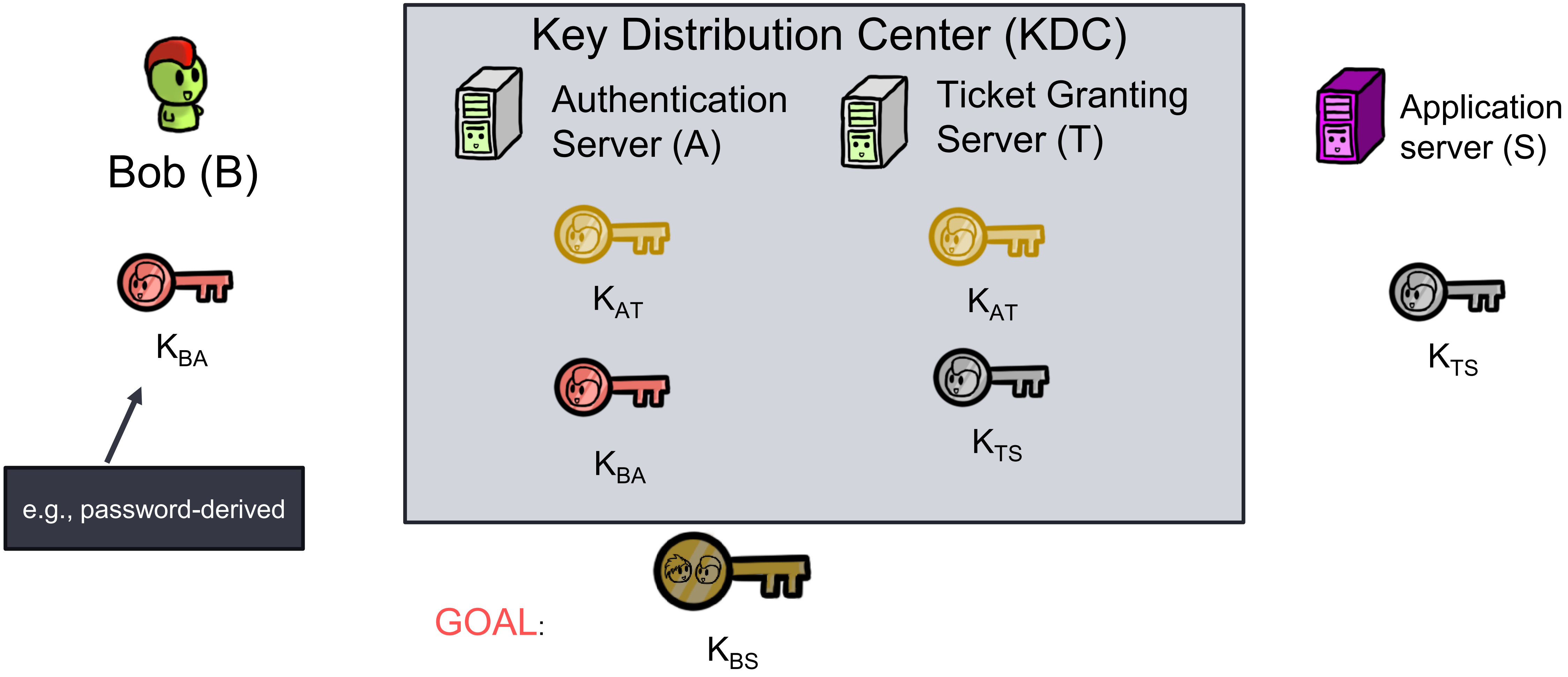
Kerberos Overview



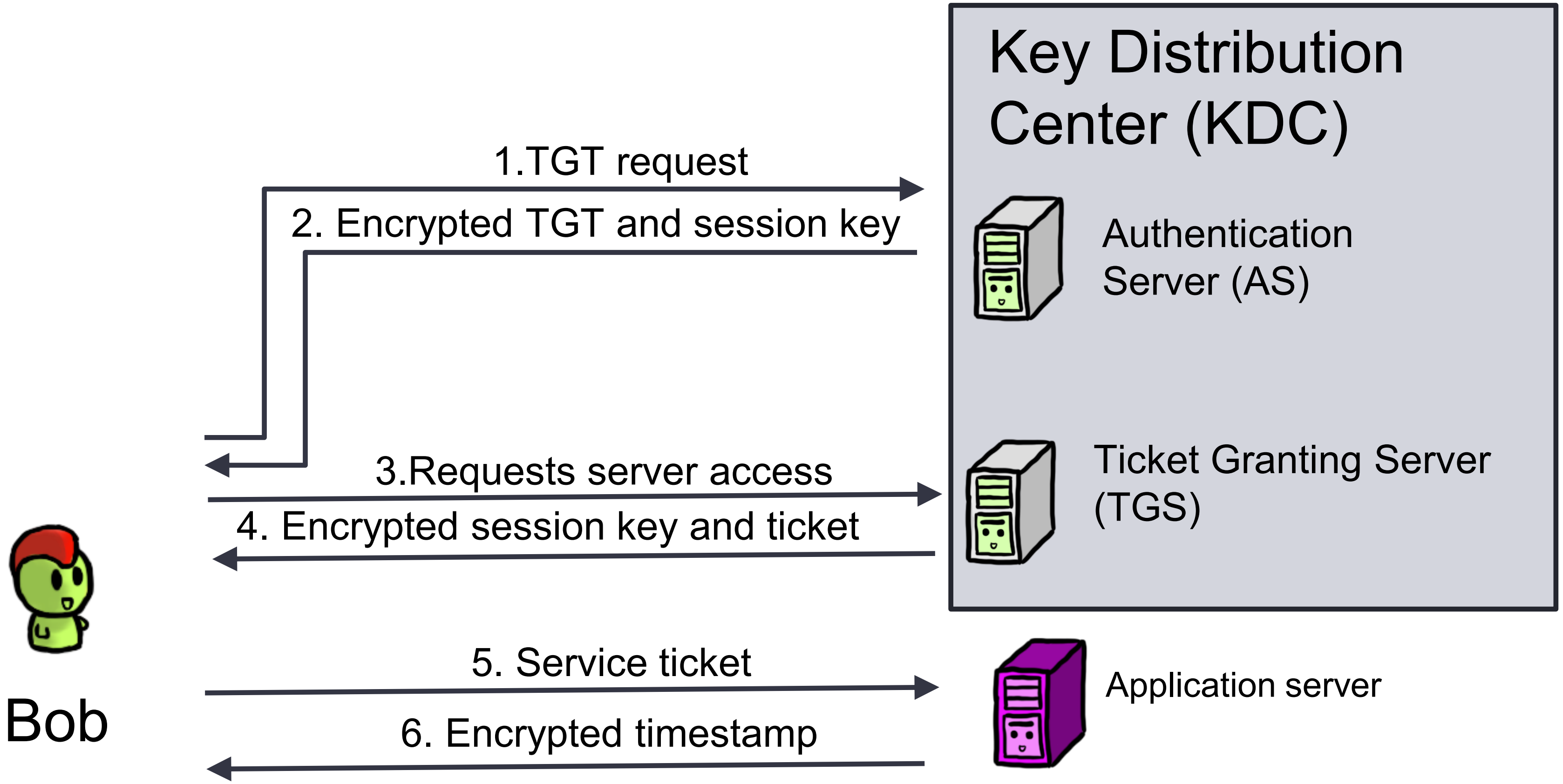
The Keys


Bob (B)
K_BA
Key Distribution Center (KDC)
Authentication Server (A)
K_AT
K_BA
Ticket Granting Server (T)
K_AT
K_TS
Application server (S)
K_TS
GOAL:
K_BS

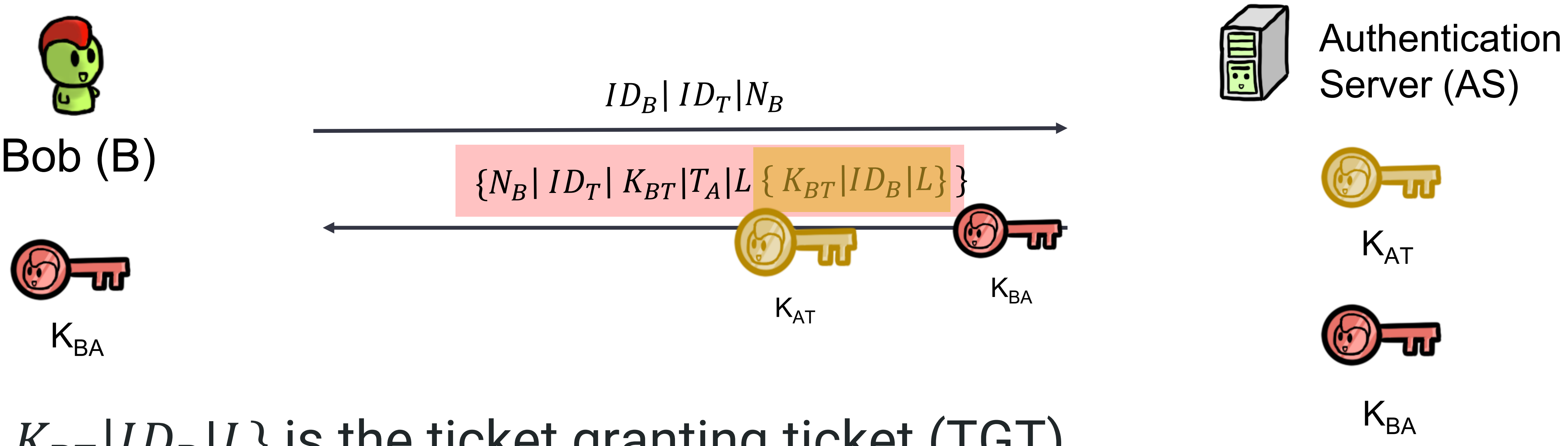
The Keys


Bob (B)
K_BA
e.g., password-derived
Key Distribution Center (KDC)
Authentication Server (A)
Ticket Granting Server (T)
K_AT
K_BA
K_AT
K_TS
Application server (S)
K_TS
GOAL:
K_BS

Kerberos Overview

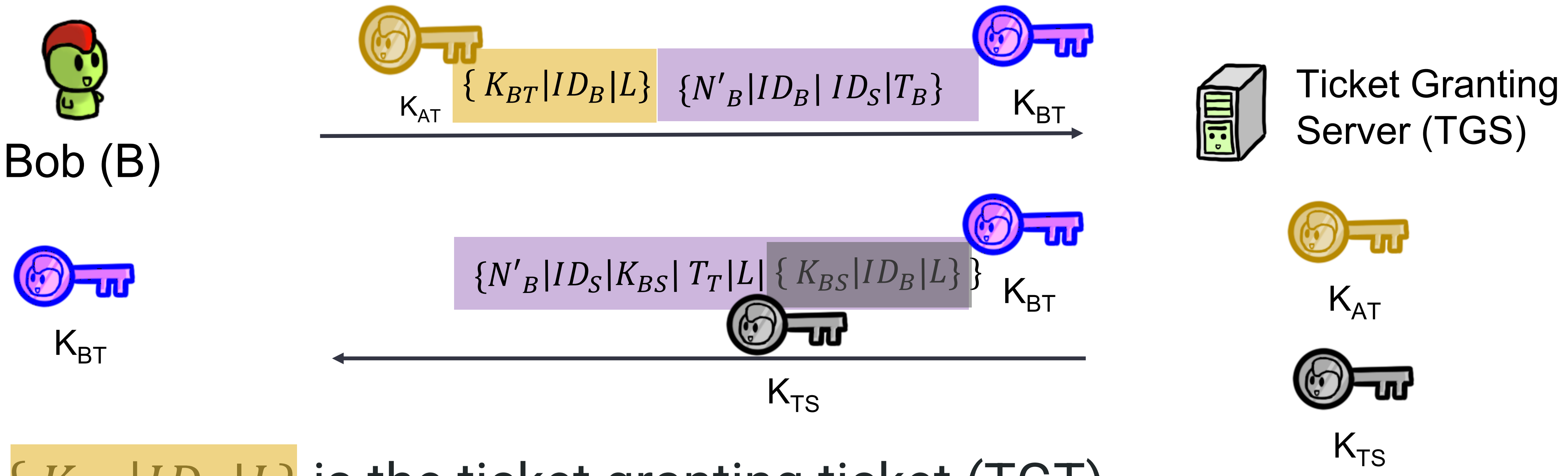


Breaking Down Kerberos – Part 1



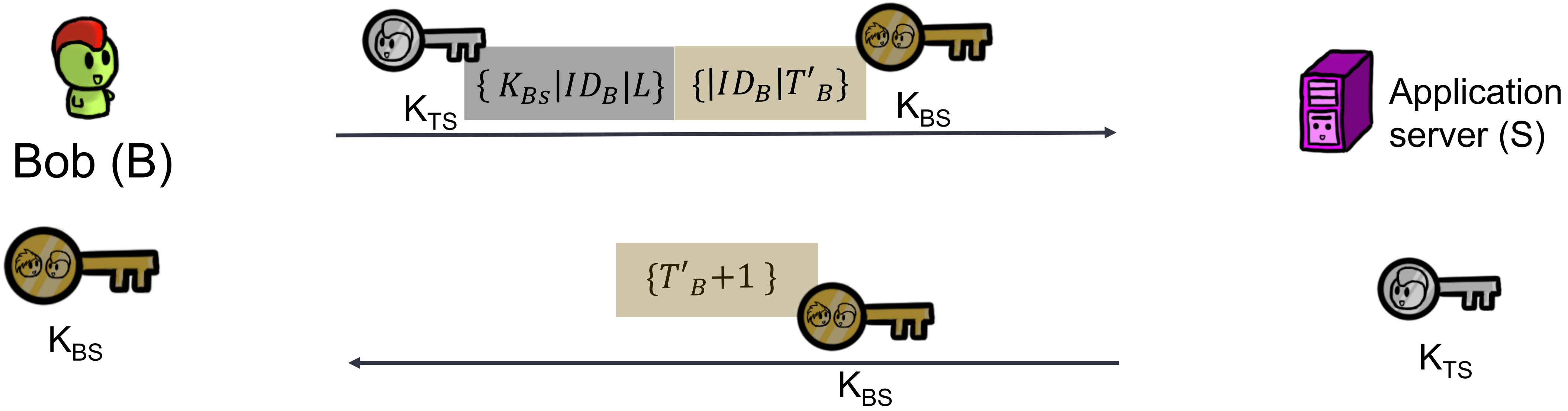
- $\{K_{BT}|ID_B|L\}$ is the ticket granting ticket (TGT)
- L is lifetime, T_A is the timestamp at A, N_B is a nonce

Breaking Down Kerberos – Part 2



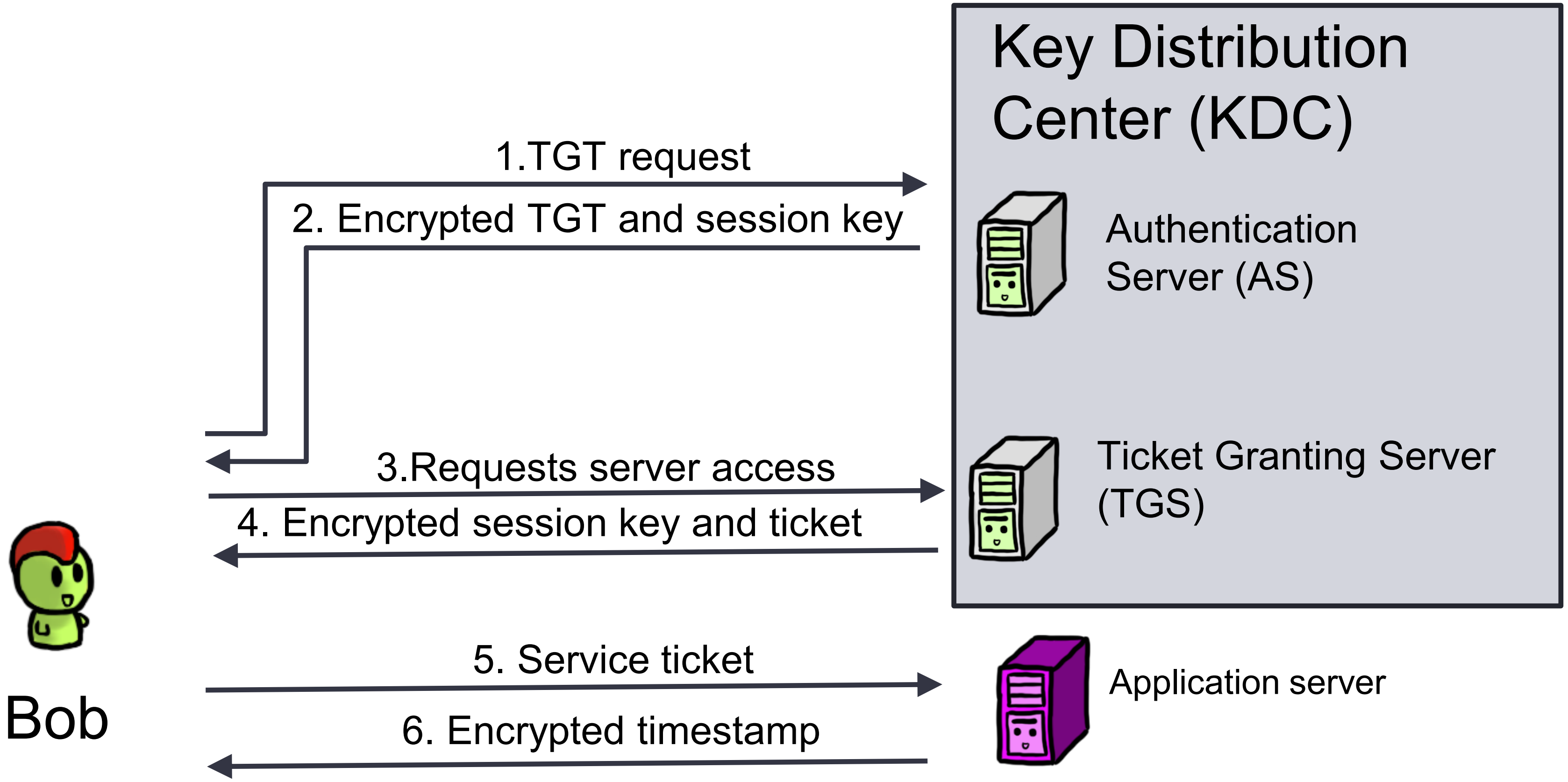
- $\{K_{BT}|ID_B|L\}$ is the ticket granting ticket (TGT)
- $\{K_{BS}|ID_B|L\}$ is the service ticket (ST)
- K_{BT} is a session key between Bob and the TGS

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

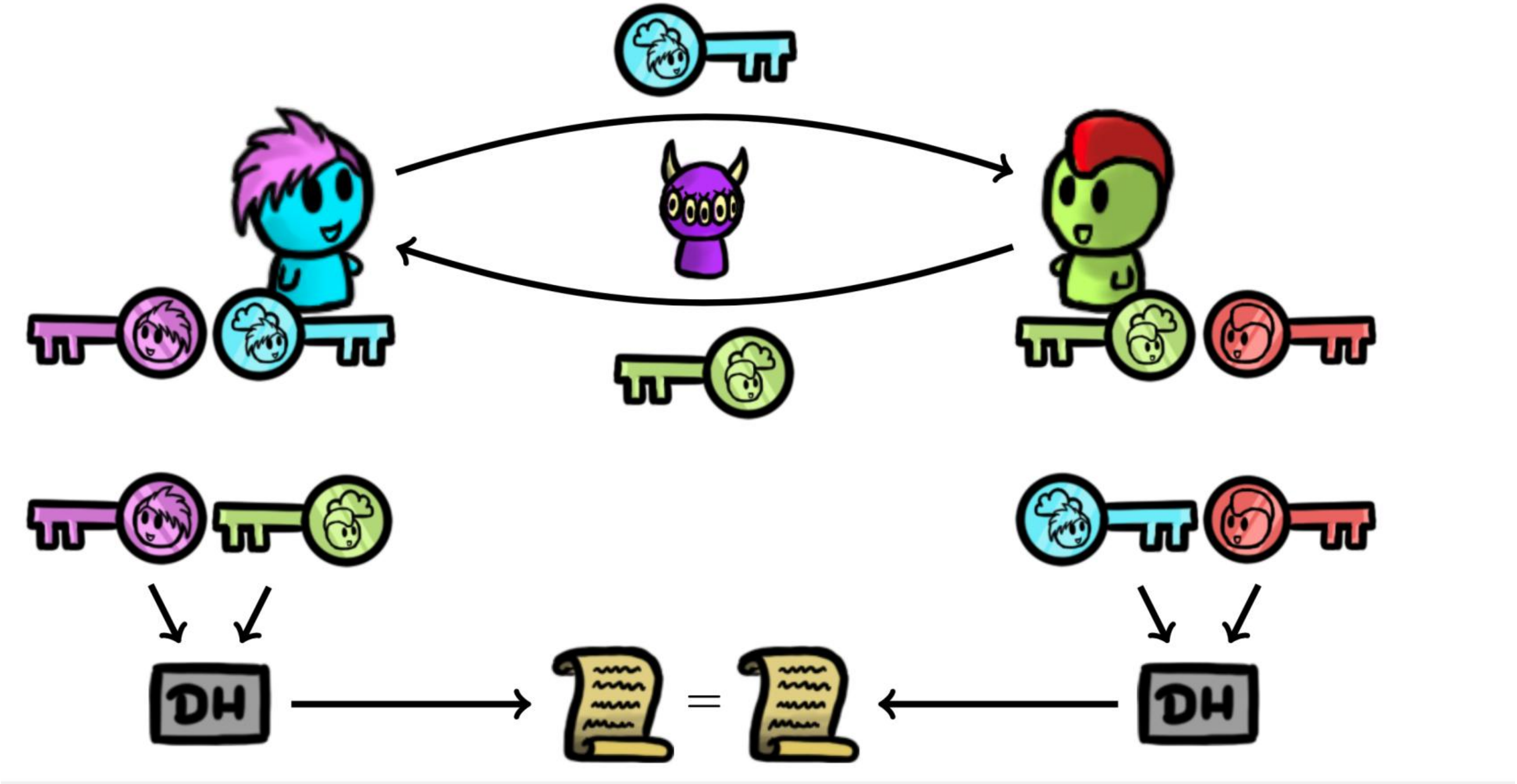


Why does Kerberos help us?

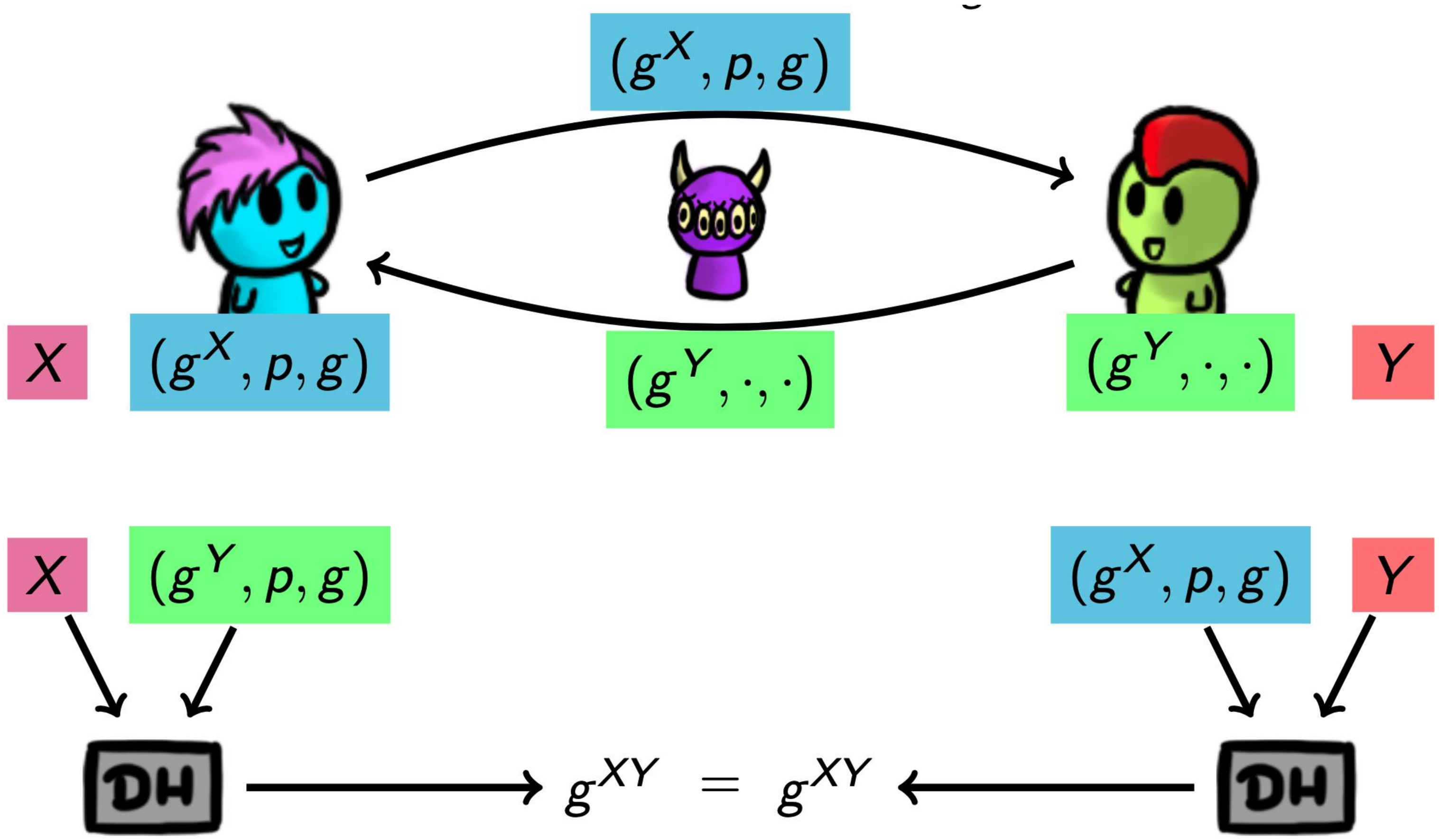
- Timestamps included in previously insecure messages
- All tickets include a Lifetime (time at which they expire)

Asymmetric Crypto Authentication

Recall the Diffie-Hellman key exchange



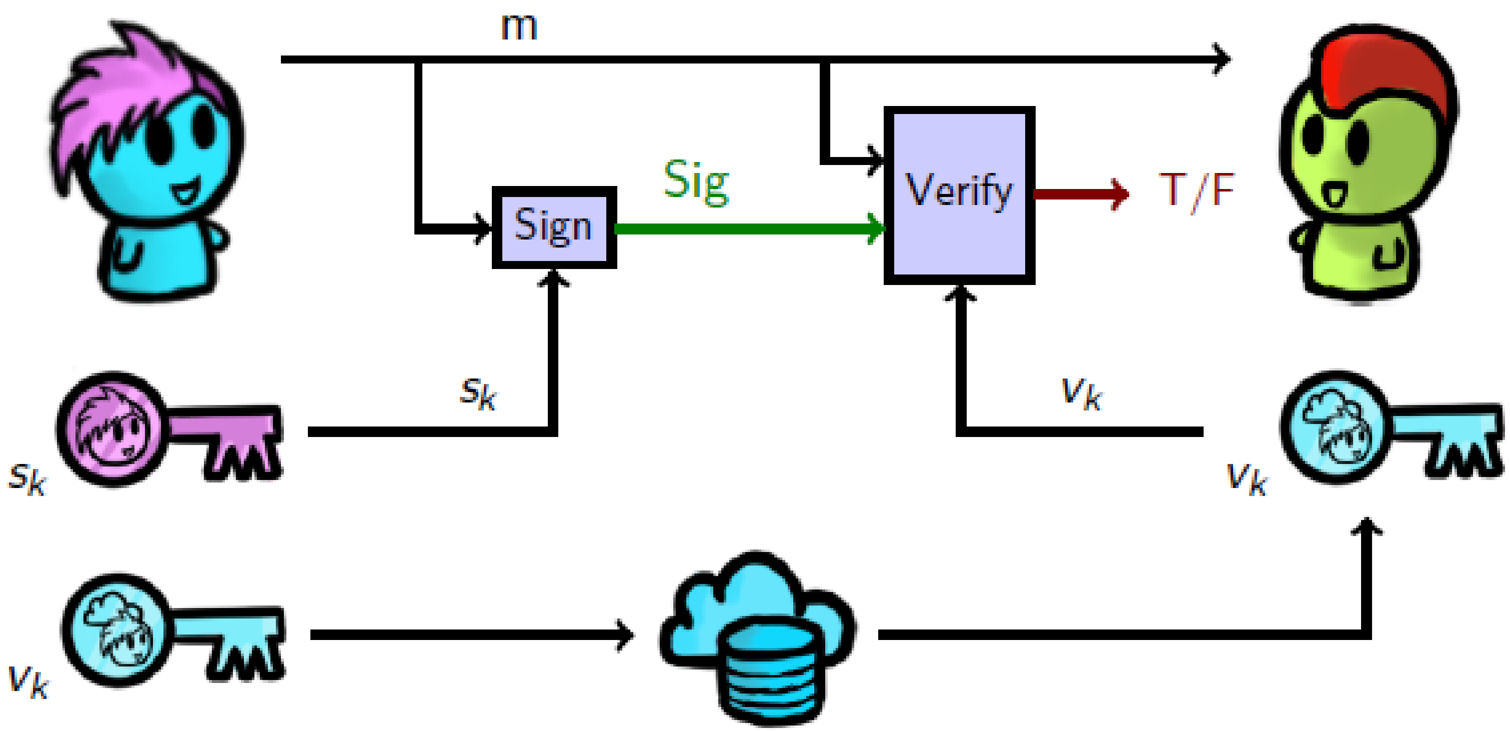
Diffie-Hellman key exchange – Altogether



What's the Problem!

- Authentication!
- Need to verify the public keys!

Recall, Digital Signatures

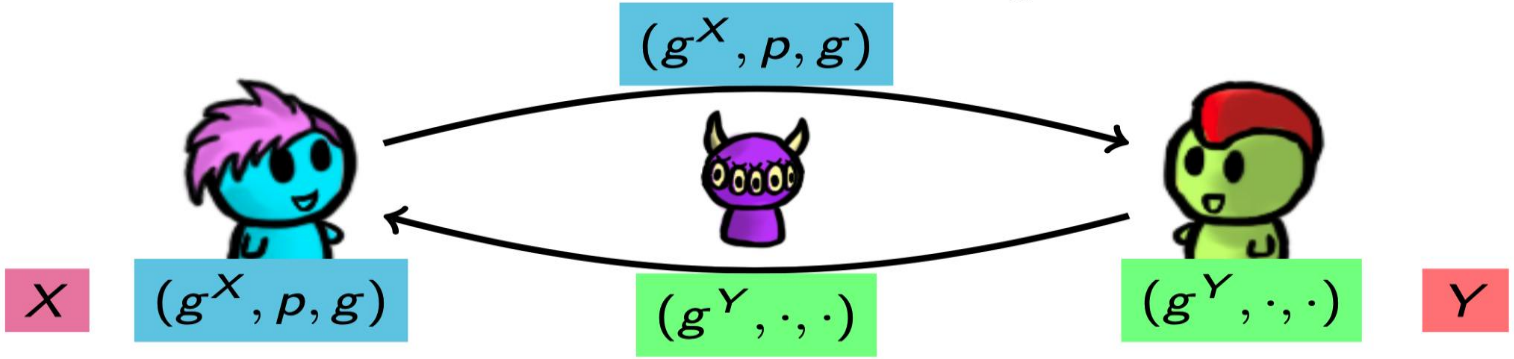


The Key Management Problem

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

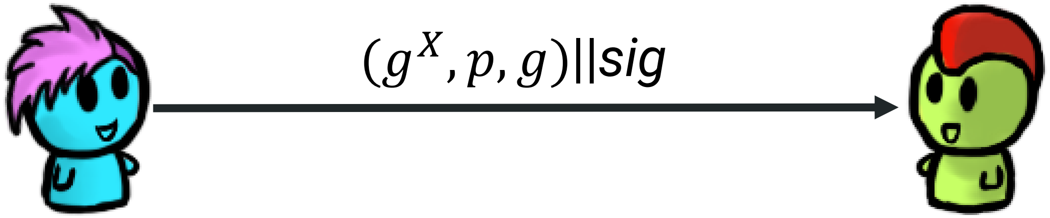
A: By having each other's verification key!

Before



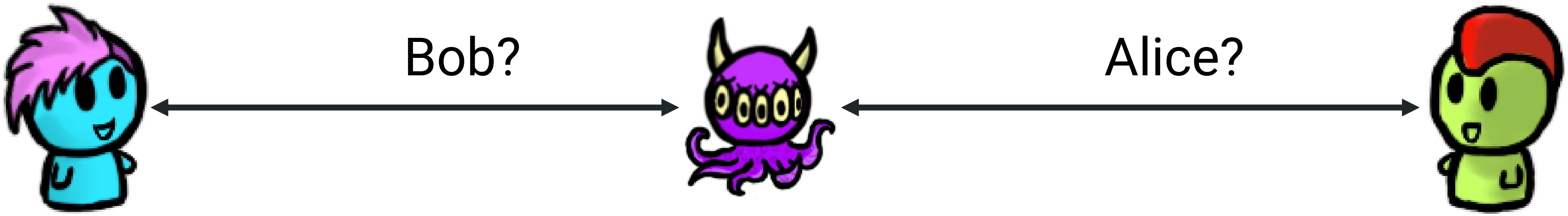
After

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



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

The Key Management Problem

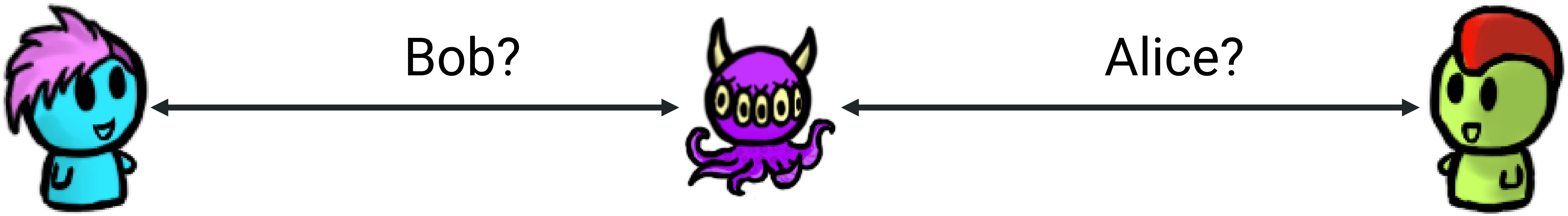


Q: How can Alice and Bob be sure they’re talking to each other?

A: By having each other’s verification key!

Q: But how do they get the keys…

The Key Management Problem…Solutions?



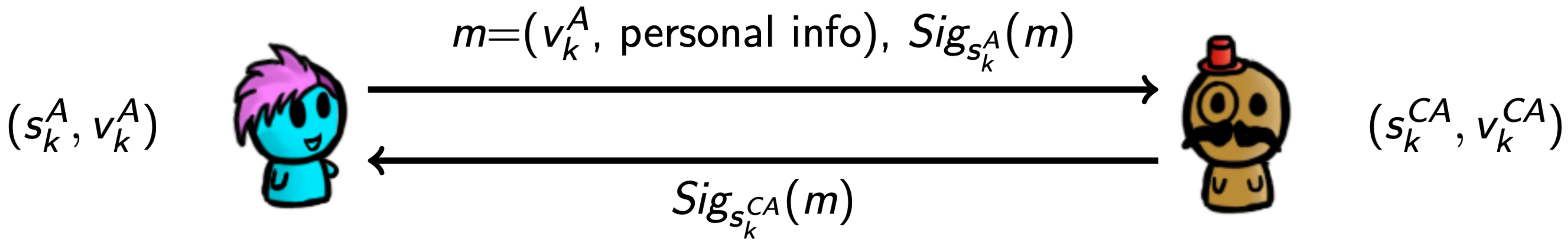
Q: But how do they get the keys…

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)

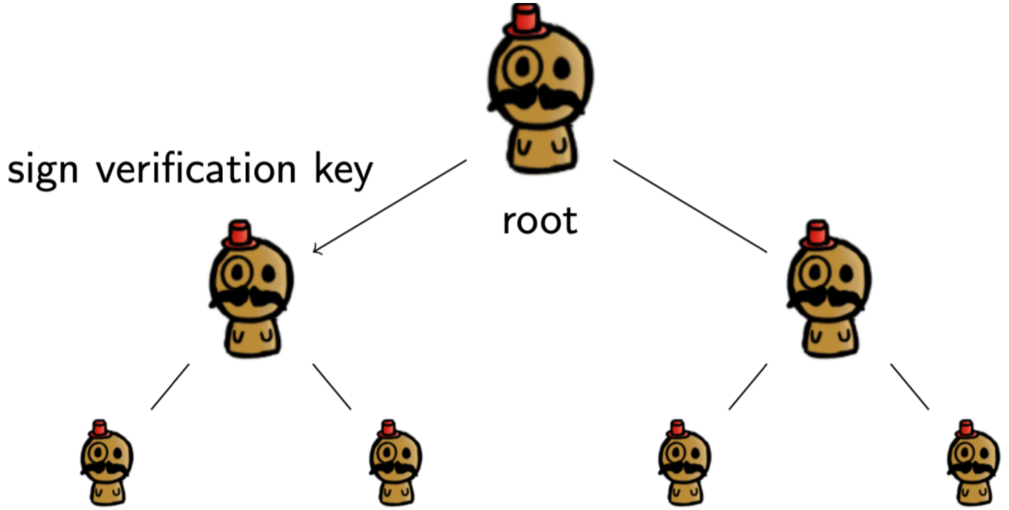
Certificate Authorities (CAs)



- 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

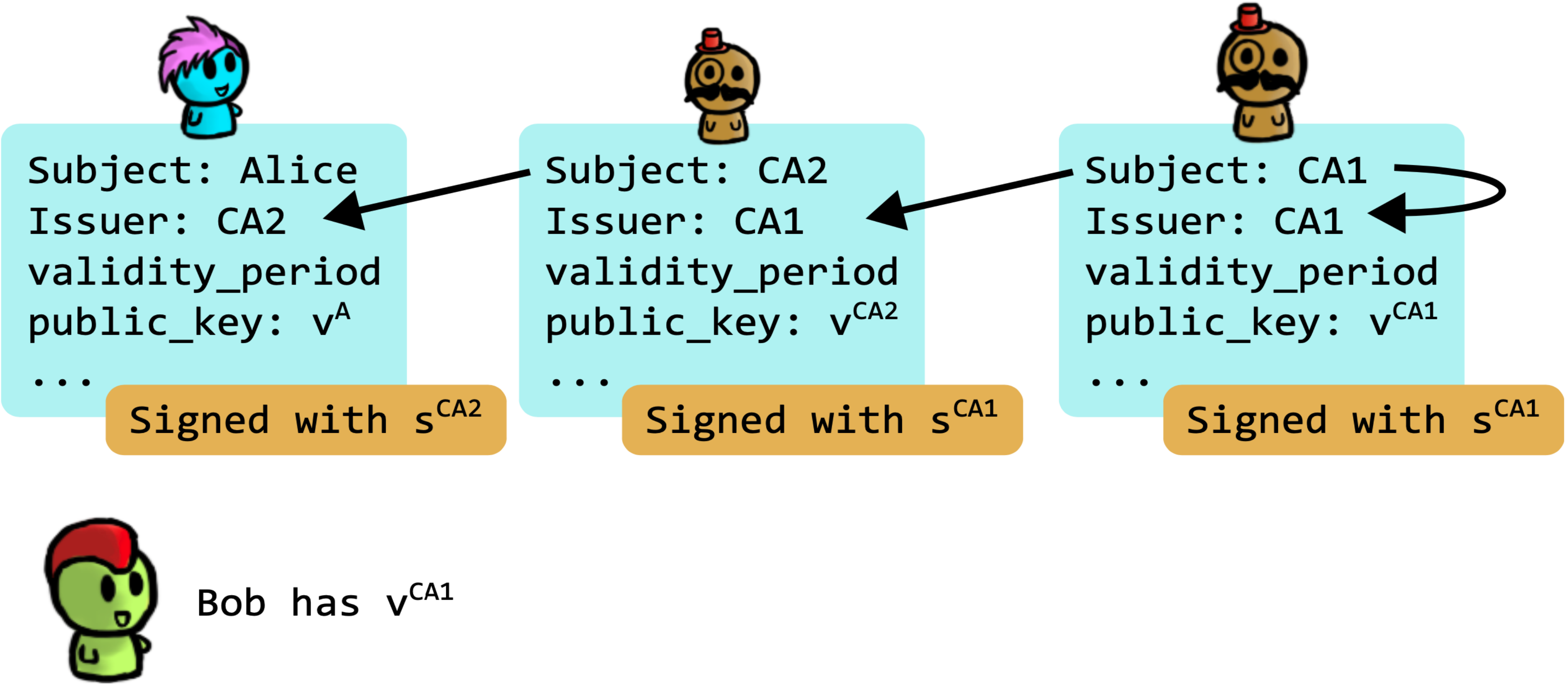
Certificate Authorities

- Everyone is assumed to have a copy of the CA's verification key (v_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



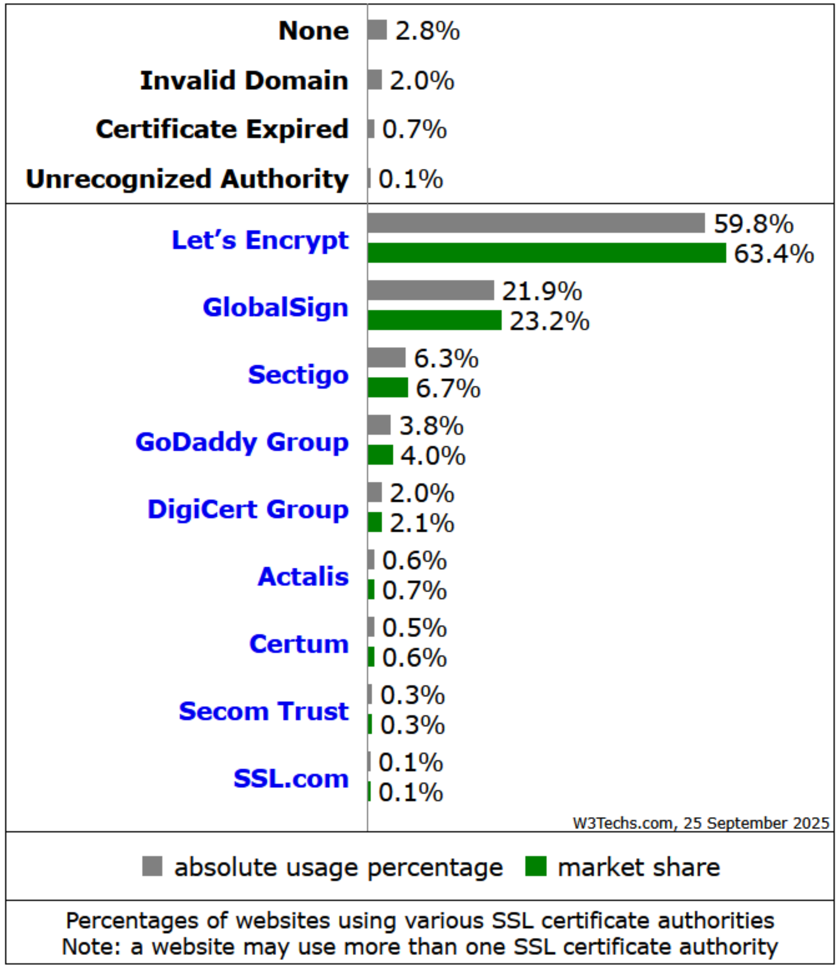
Chain of Certificates

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 keys installed on browser
- https://letsencrypt.org changed the game by offering free certificates



Examples

mathsisfun.com	WE1	GTS Root R4 (built-in root)

Subject Name	
Common Name	mathsisfun.com
Issuer Name	
Country	US
Organization	Google Trust Services
Common Name	WE1
Validity	
Not Before	2025-07-30, 7:52:55 a.m. (Eastern Daylight Saving Time)
Not After	2025-10-28, 8:52:53 a.m. (Eastern Daylight Saving Time)
Subject Alt Names	
DNS Name	mathsisfun.com
DNS Name	*.mathsisfun.com
Public Key Info	
Algorithm	Elliptic Curve
Key size	256 bits
Curve	P-256
Public Value	04:F6:6B:39:B7:11:A8:E5:5C:FA:53:99:30:83:99:DF:F8:1B:28:B0:0D:E2:42:BE:6A:0D:81:79:42:C5:49:22:29:11:DE:79:E4:6D:27:51:AC...
Miscellaneous	
Download	PEM (cert), PEM (chain)
Serial Number	11:64:F9:2F:D6:45:ED:26:0D:BE:07:C9:62:C8:D1:63
Signature Algorithm	ECDSA with SHA-256
Version	3
Fingerprints	
SHA-256	82:B4:43:E1:42:0C:CB:A7:91:E7:3B:4E:FC:37:7A:23:57:AC:BB:7C:15:55:5E:55:7E:1A:76:F4:3B:4F:A7:C8
SHA-1	A7:EC:D3:66:E1:26:2B:5D:B9:6B:9C:E3:C3:9A:3B:30:C3:8F:58:BD

mathsisfun.com	WE1	GTS Root R4 (built-in root)

Subject Name	
Country	US
Organization	Google Trust Services
Common Name	WE1
Issuer Name	
Country	US
Organization	Google Trust Services LLC
Common Name	GTS Root R4
Validity	
Not Before	2023-12-13, 4:00:00 a.m. (Eastern Daylight Saving Time)
Not After	2029-02-20, 9:00:00 a.m. (Eastern Daylight Saving Time)
Public Key Info	
Algorithm	Elliptic Curve
Key size	256 bits
Curve	P-256
Public Value	04:6F:CD:3A:FE:67:57:47:4C:21:03:85:40:C2:47:5D:BB:58:47:0F:40:C1:5C:17:85:C6:19:37:E7:D5:7C:ED:86:4B:9B:81:D9:D7:1A:13:A...
Miscellaneous	
Download	PEM (cert), PEM (chain)
Serial Number	7F:F3:19:77:97:2C:22:4A:76:15:5D:13:B6:D6:85:E3
Signature Algorithm	ECDSA with SHA-384
Version	3
Fingerprints	
SHA-256	1D:FC:16:05:FB:AD:35:8D:8B:C8:44:F7:6D:15:20:3F:AC:9C:A5:C1:A7:9F:D4:85:7F:FA:F2:86:4F:BE:BF:96
SHA-1	10:8F:BF:79:4E:18:EC:53:47:A4:14:E4:37:0C:C4:50:6C:29:7A:B2

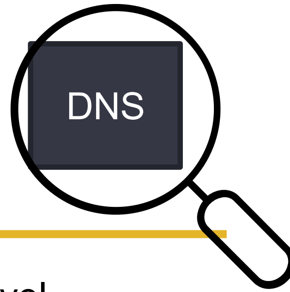
DNSSEC

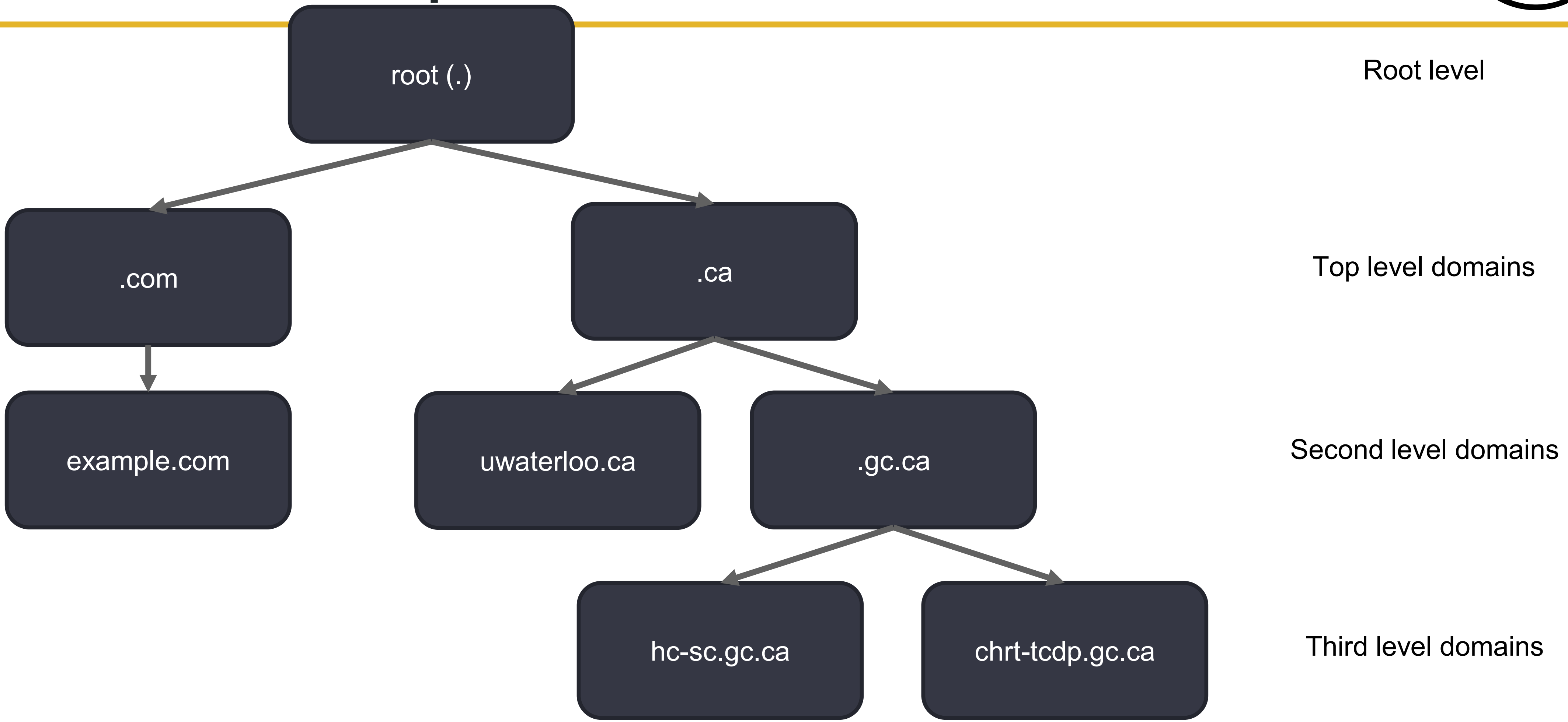
What is DNS?

- 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

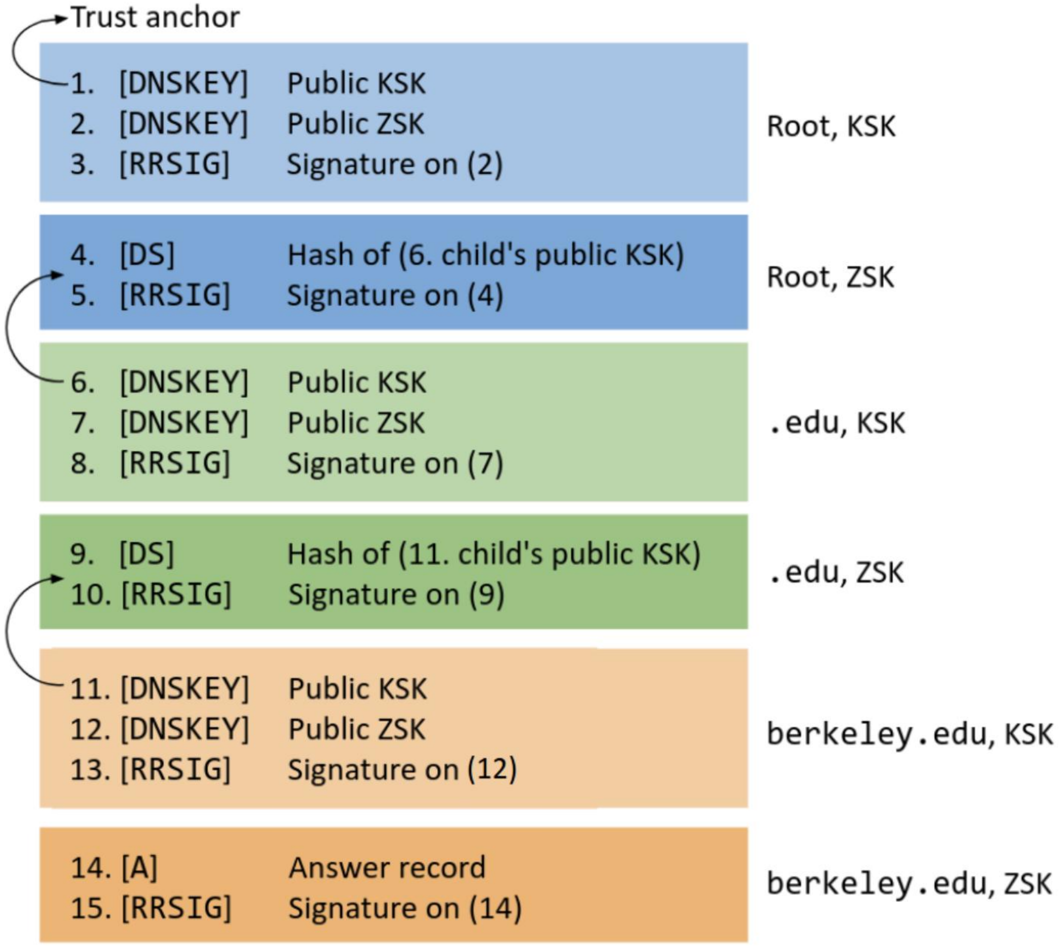
Use **digital signatures** to make sure a correct and unmodified message is received from the correct entity!

- New records added to DNSSEC signed zone
- Record sets (RRSets) are signed, instead of individual records
- Have two keys:
    - **Key Signing Key (KSK):** used for signing a zone's verification KSK and ZSK, kept in trusted hardware, hard to change, results in long signatures
    - **Zone Signing Key (ZSK):** used for signing a zone's RRSets, changed more often, results in short signatures

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

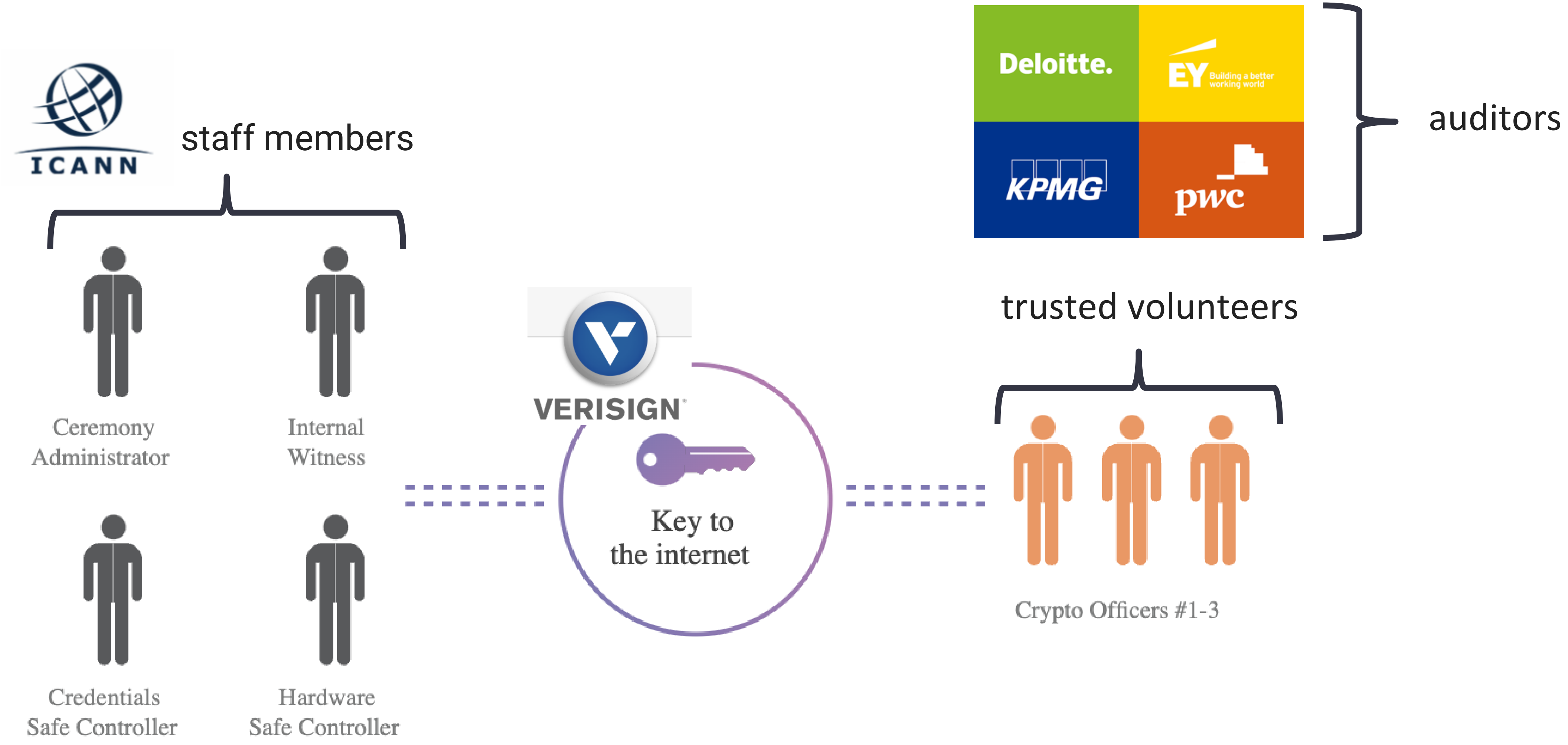


How do we maintain key integrity?

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/learning/dns/dnssec/root-signing-ceremony/)
- Each zone's parent zone contains a "Delegate signer" (DS) record, which is used to verify the zone's **KSK**
  - Essentially, a hash of **KSK**

Who’s involved?



DNSSEC Root Signing Ceremony

- For signing the root DNS public keying information
    - There are two geographically distinct locations that safeguard the root key-signing key: **El Segundo, CA** and **Culpeper, VA**