CS489/689 Privacy, Cryptography, Network and Data Security

Authentication Protocols

A2 is out!

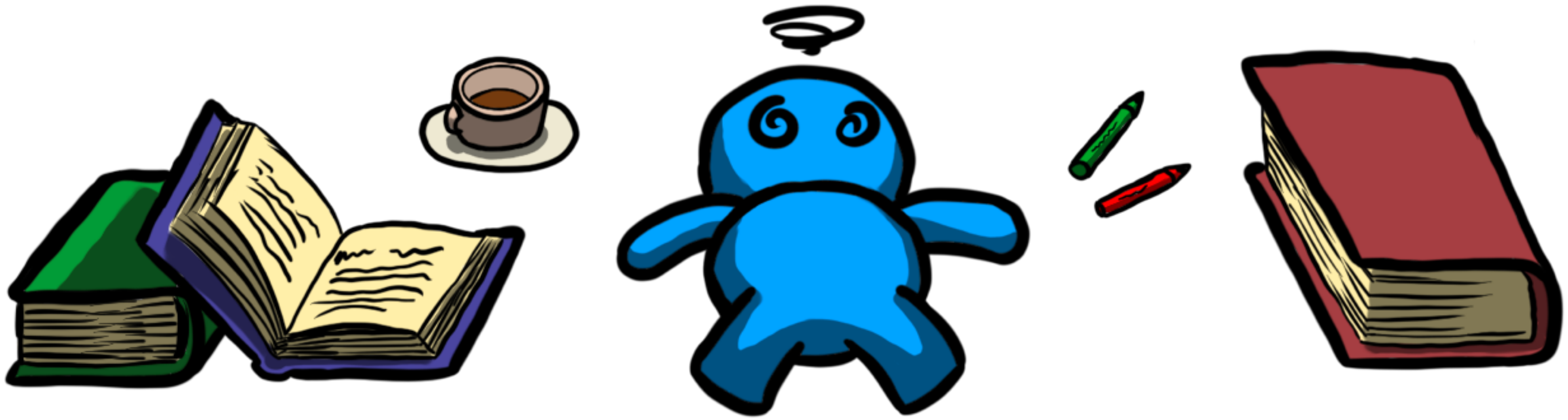
- Written Q1,2,5 after today
  - Q3 by Feb 14th Q5 by Feb 16th
- Programming Q1 and Q3 already
  - Q2 and Q4 by Feb 9th
- Office hours Mondays 12:30 DC3333

Grad Student Projects

- Proposals Due **<u>Feb 27th</u>** (not Feb 21st)
- Come talk to us during office hours or make an appointment.

Exam Schedule is out!

- Wednesday April 19th at 7:30PM – 10PM
- Location TBA



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.

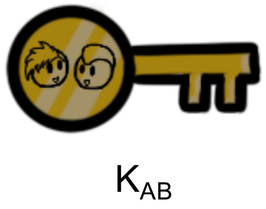
Today’s Focus

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

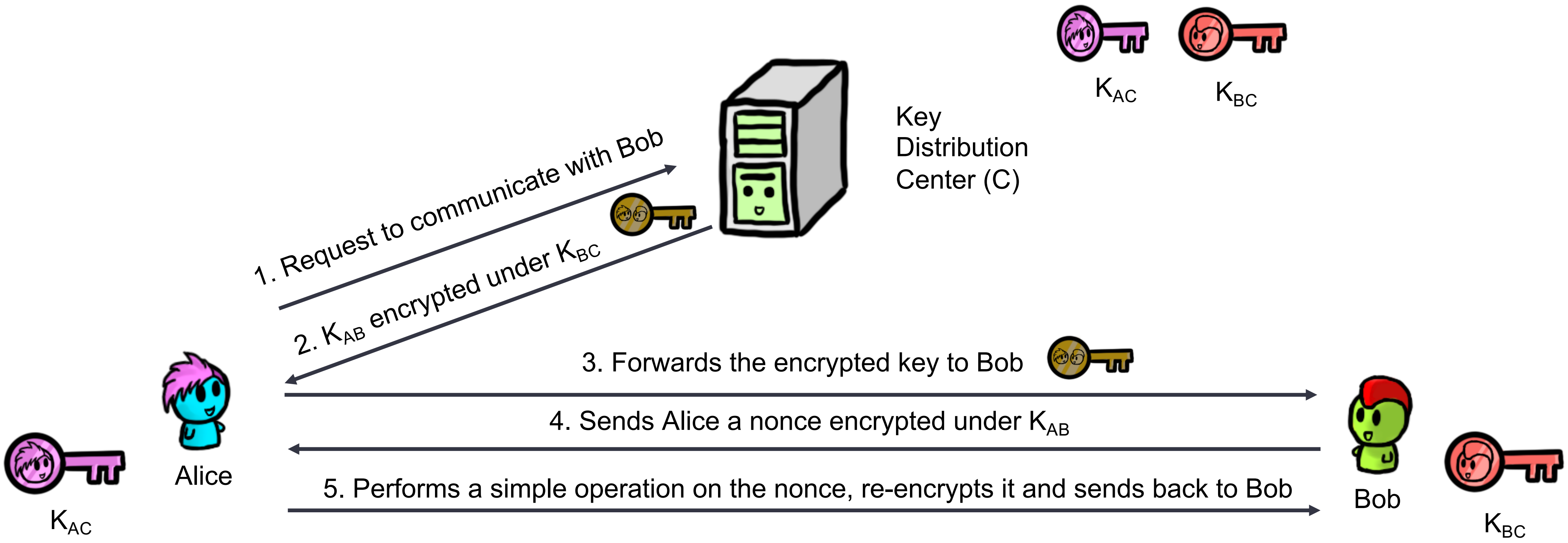
Symmetric Crypto Authentication

Needham-Schroeder Overview

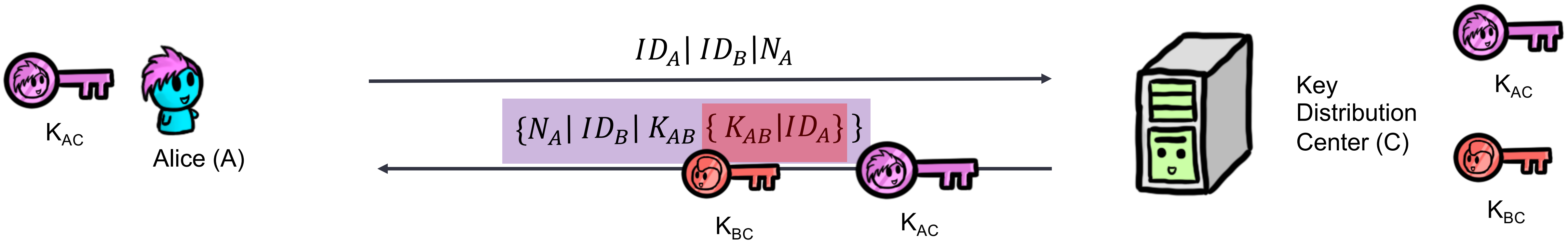
- Alice (A) wants to initiate communication with Bob (B)
- A Trusted Third Party (C) with pre established symmetric keys
- K_{AC} is a symmetric key known only to A and the Key Distribution Center (C)
  - K_{BC} is a symmetric key known only to B and C
- GOAL: Generate $K_{AB,}$ a symmetric, generated key used in the session between A and B



Needham-Schroeder Flow

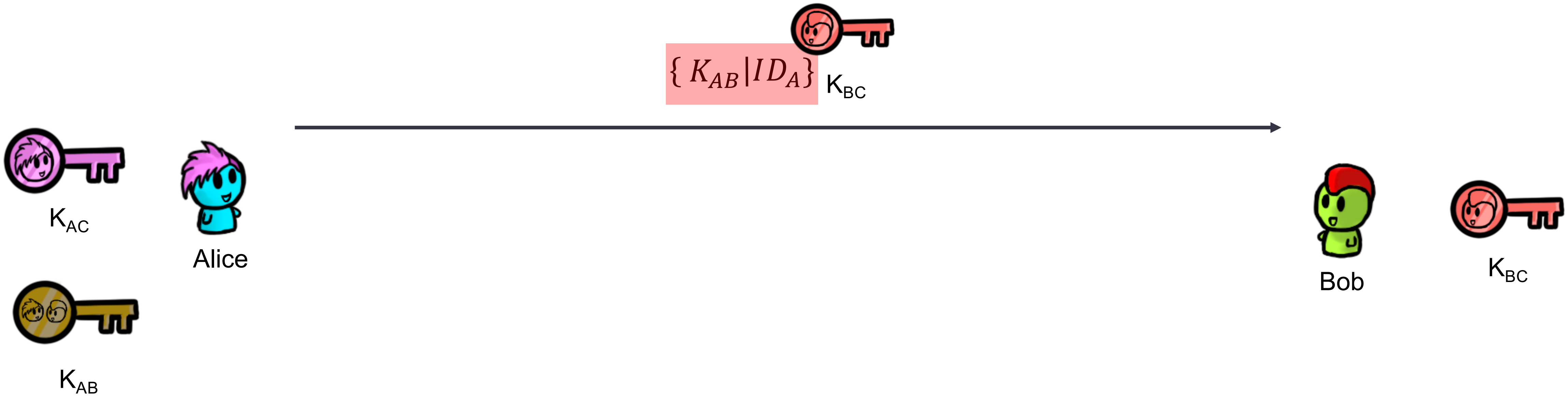


Breaking Down Needham-Schroeder Part 1



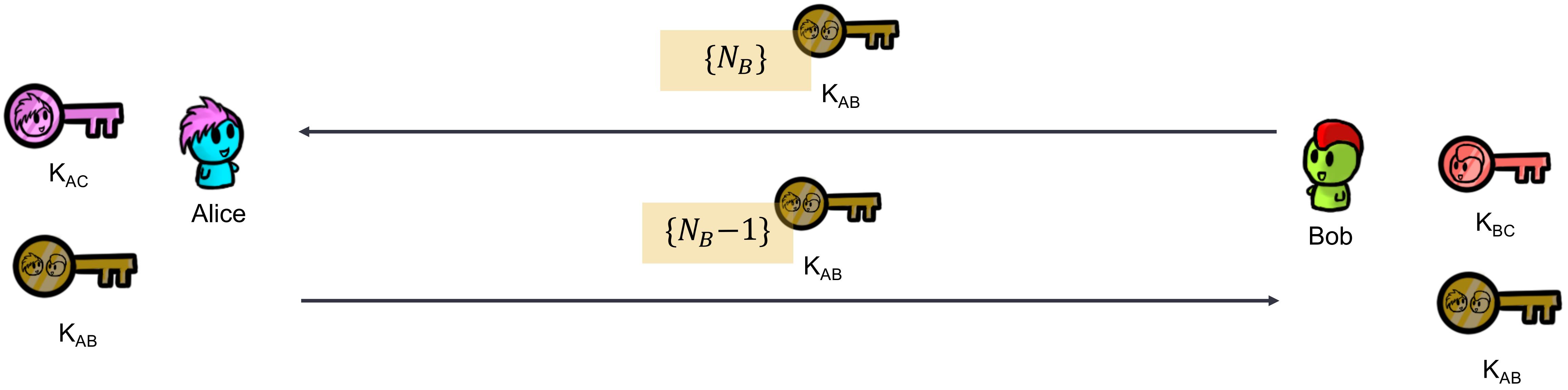
- First message in plain
- N_A is a nonce used to prevent reply attacks against Alice

Breaking Down Needham-Schroeder Part 2



- Simply forward the inner message

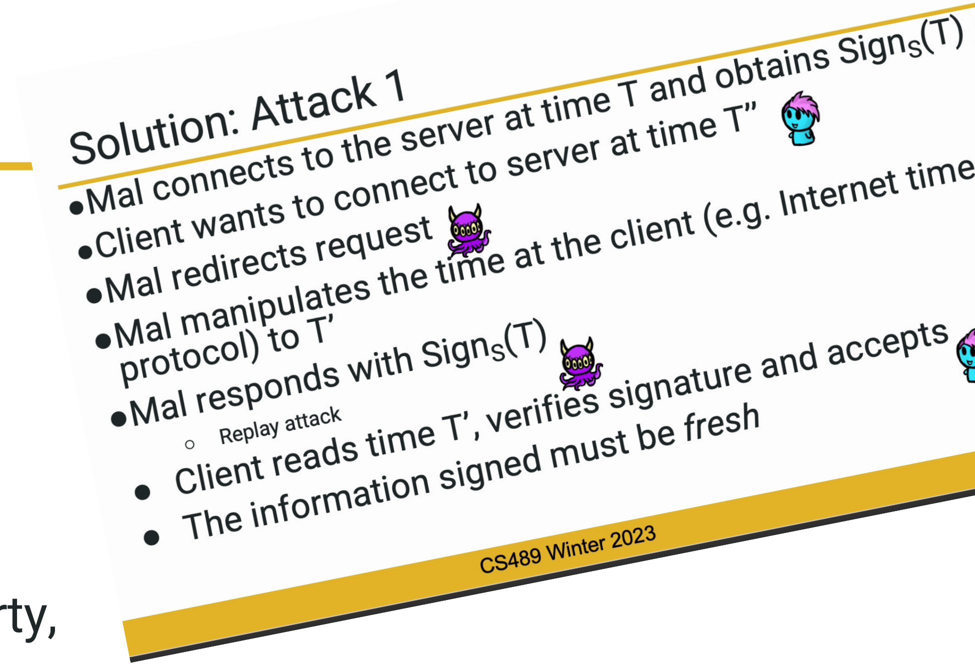
Breaking Down Needham-Schroeder Part 3

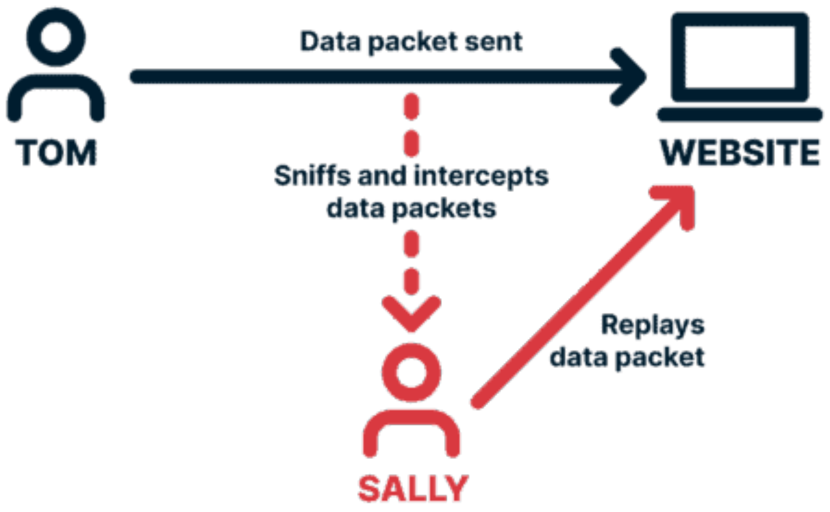


- Need to verify the keys

Replay Attacks

- We have seen examples in previous lectures.
- Definition:
  - Mallory intercepts a message meant for some other party,
  - They later send this message pretending to be some other party
    - Either delay or replay
- Example
  - Hashed password
  - Car Unlocking





Question

Act.

- Needham-Schroeder is vulnerable to replay attacks
- Write an explanation for why this is the case, using the algorithm as justification (one or two sentences)
- Sketch out a solution for how you could solve this issue (one or two sentences)
- Submit the write up to learn

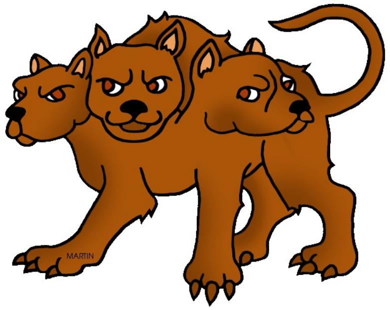
Typical Defenses against replays

- Need to ensure the data is “fresh”

E.g.

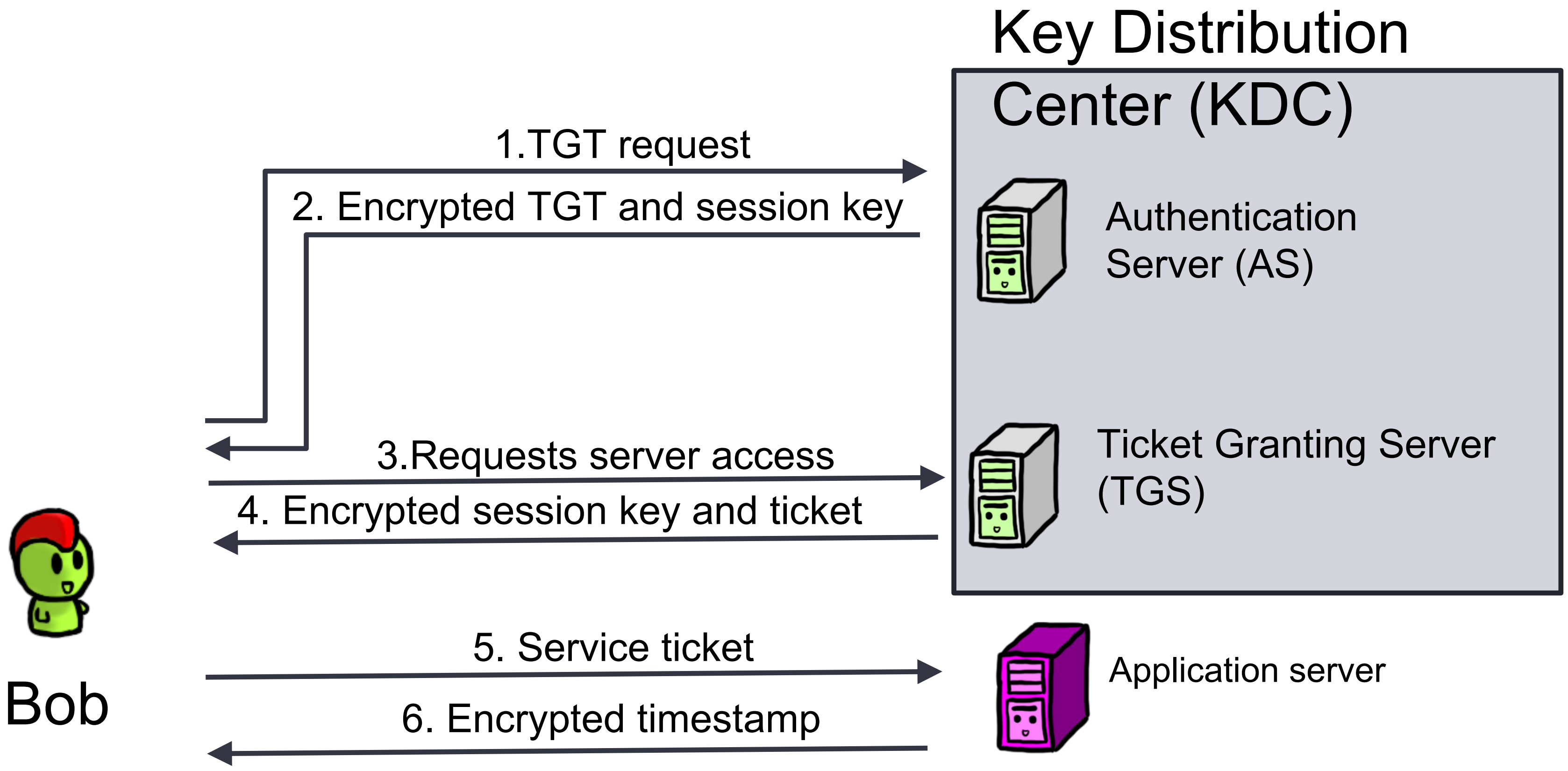
- Using a Nonce
- Timestamps
    - Ensure Synchronization
- Caching Responses

Kerberos

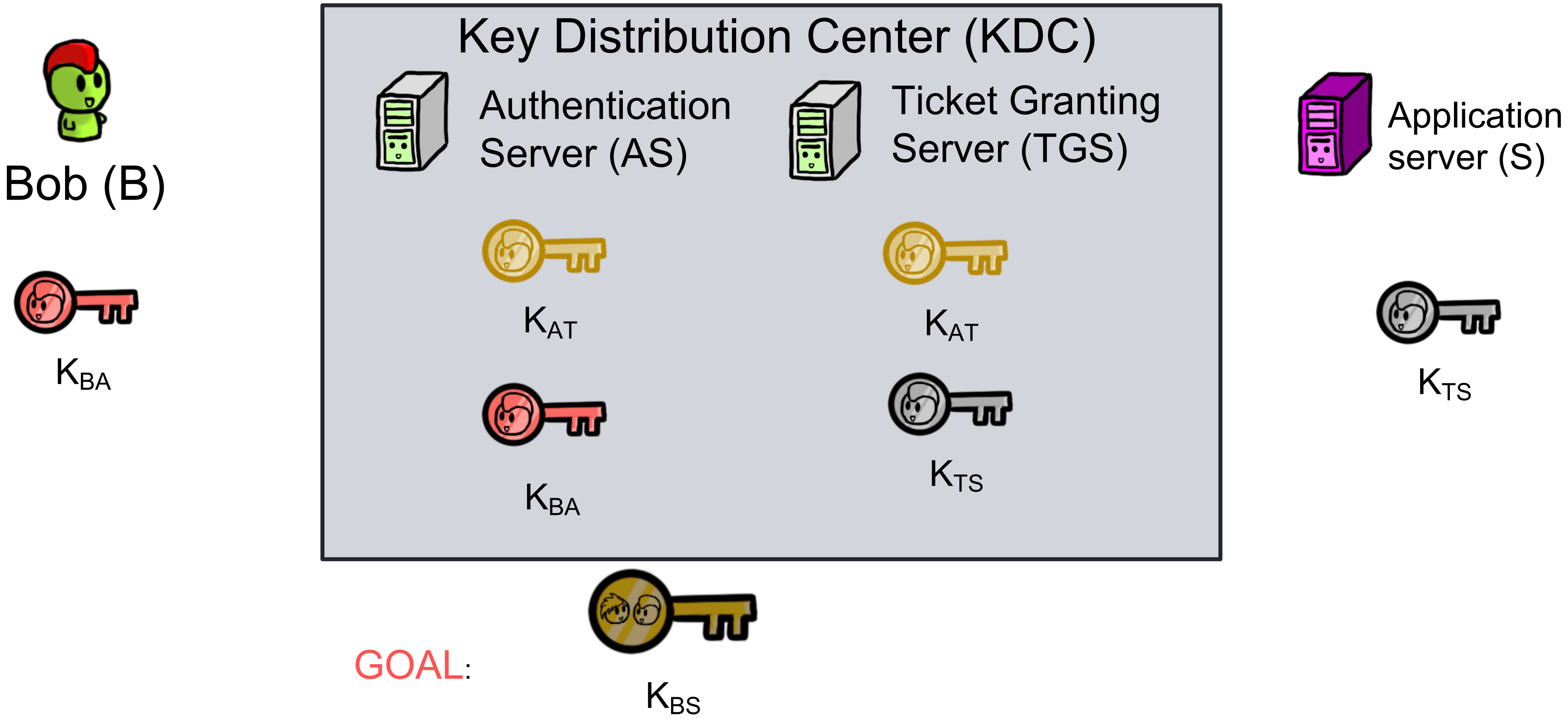


- Based on the Needham-Schroeder protocol
- Fixes the potential for a replay attack vulnerability by adding a timestamp
- Used in Windows Active Directory
- Effective Access Control
  - Each client only needs single key.
  - Each server also only needs a single key.
  - Mutual Authentication.

Kerberos Overview

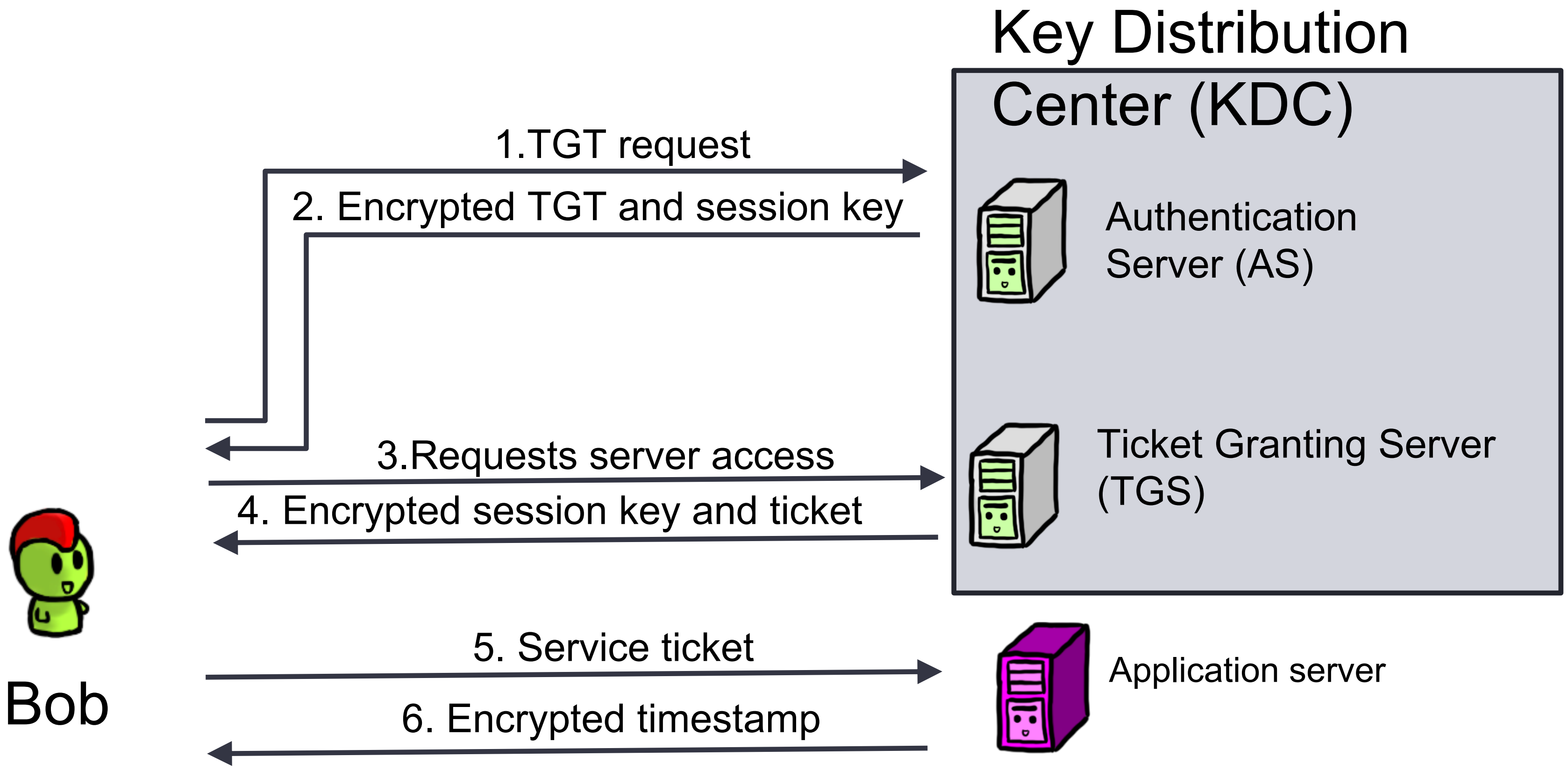


The Keys

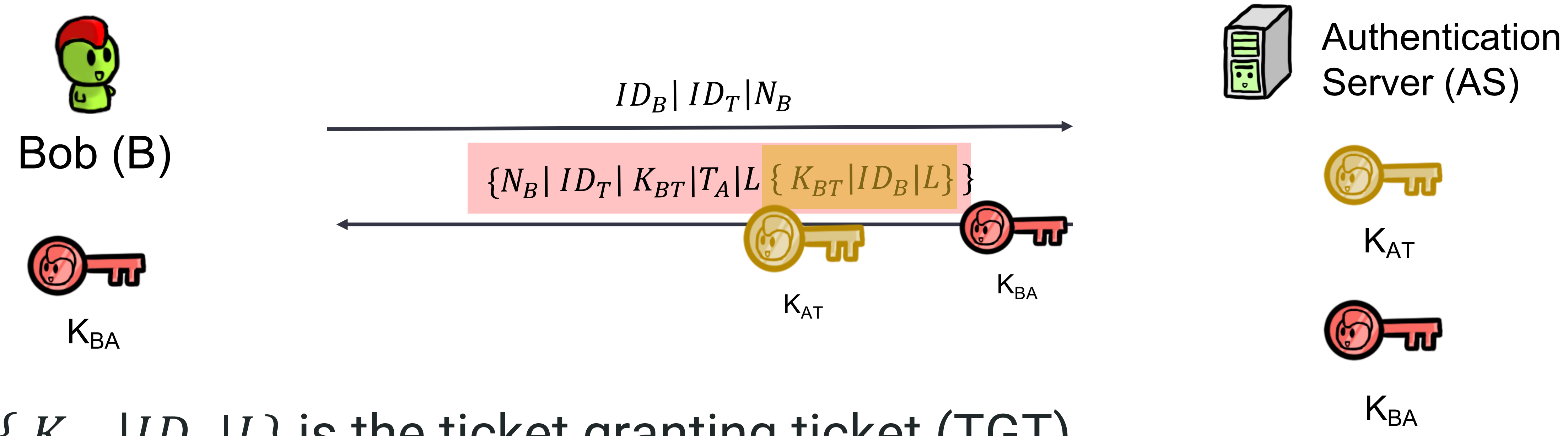


Bob (B)
K_{BA}
Key Distribution Center (KDC)
Authentication Server (AS)
Ticket Granting Server (TGS)
K_{AT}
K_{AT}
K_{BA}
K_{TS}
Application server (S)
K_{TS}
GOAL:
K_{BS}

Kerberos Overview

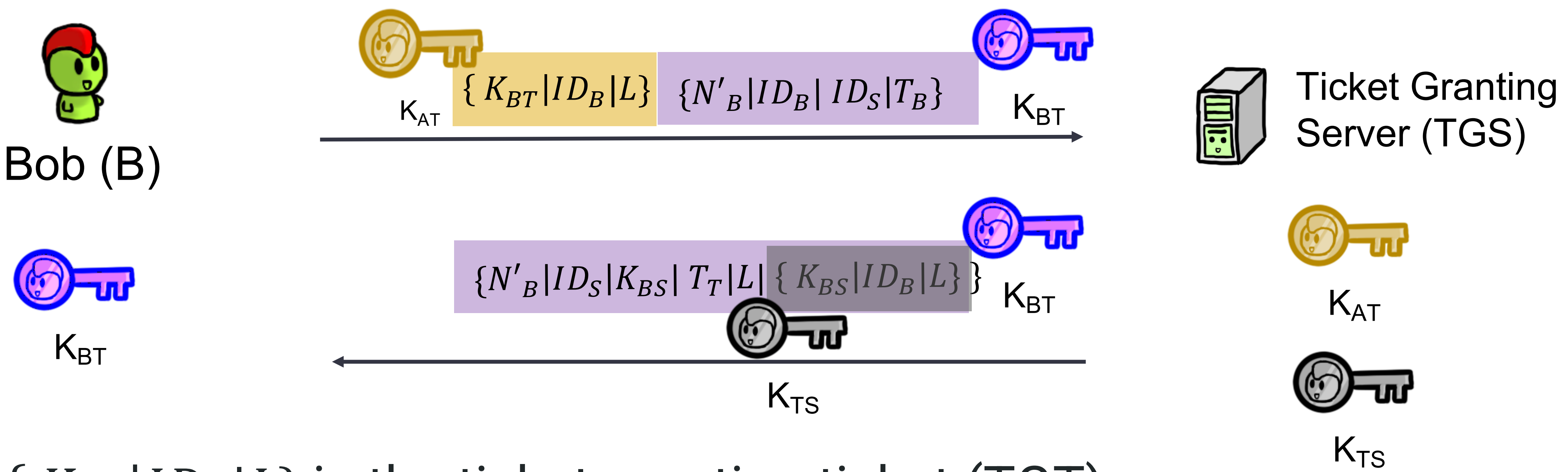


Breaking Down Kerberos – Part 1



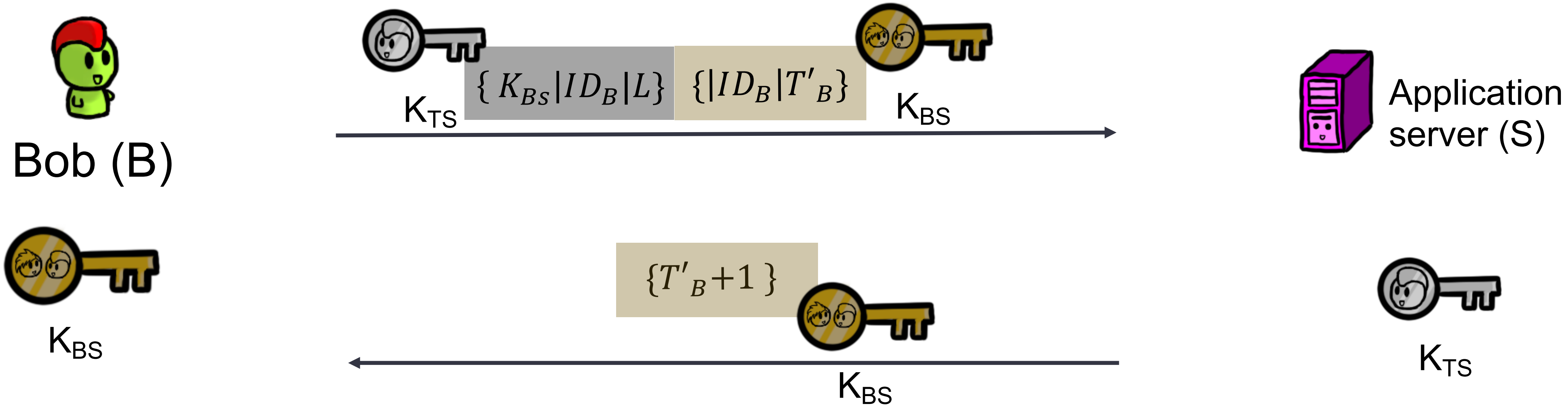
- $\{\ K_{BT}|ID_B|L\}$ is the ticket granting ticket (TGT)
- L is lifetime, T_A is the timestamp at A, N_B is a nonce

Breaking Down Kerberos – Part 2



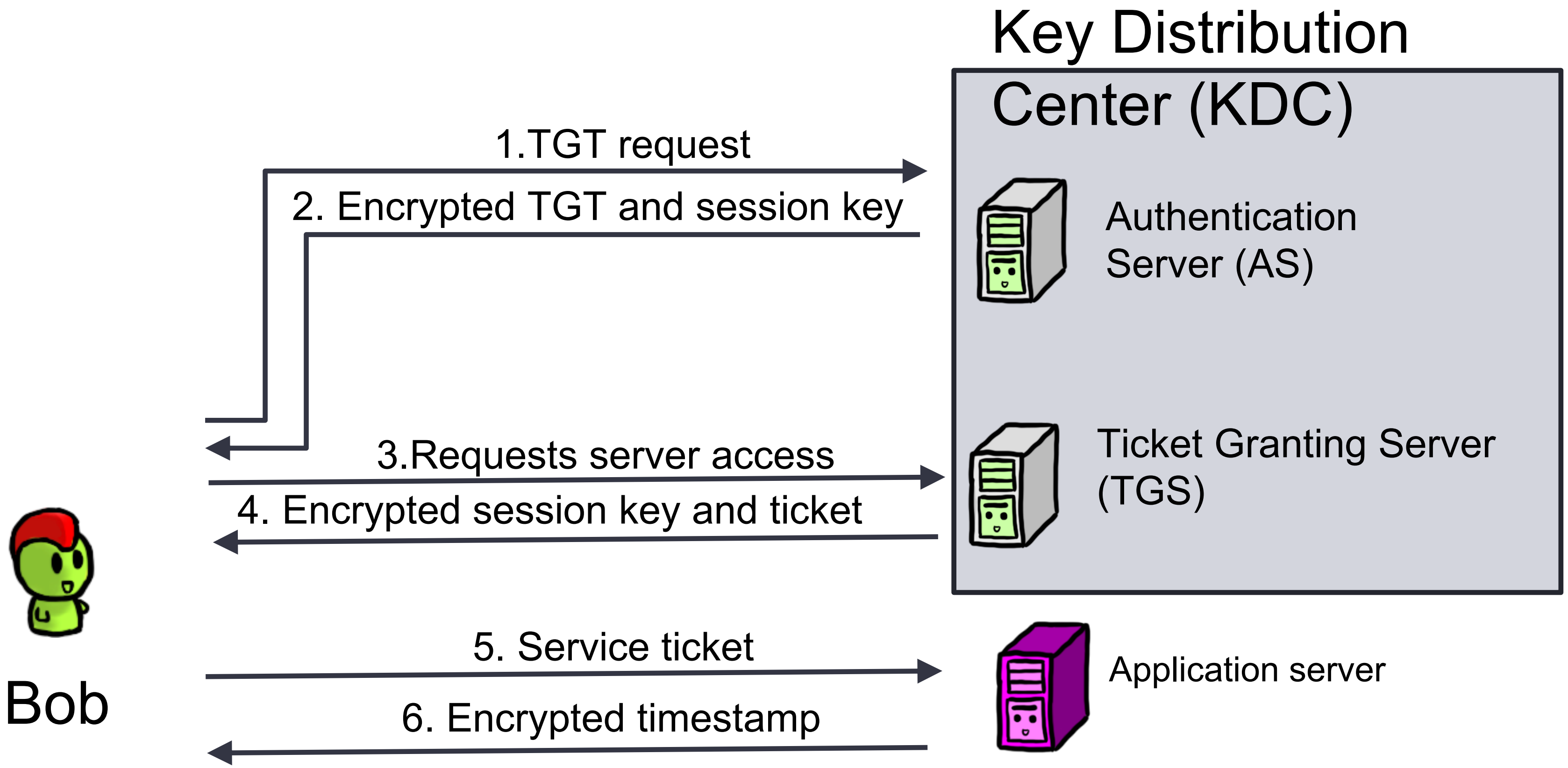
- $\{K_{BT}|ID_B|L\}$ is the ticket granting ticket (TGT)
- $\{K_{BS}|ID_B|L\}$ is the service ticket (ST)
- K_{BT} is a session key between Bob and the TGS

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



Reflect, why does Kerberos fix it

- Timestamps in previously insecure messages
- All tickets include a Lifetime (time they expire)
- Newer versions also include caches of previous messages
  - Bob: service ticket
  - TGS: User ID and timestamp
  - Service: User ID and timestamp

Kerberos vulnerabilities

- Kerberoasting: try to crack the first message sent.
  - Fix -> pre - authenticate the client first
- Forged service authentication ticket (silver ticket):
  - Requires compromising various accounts. Fix -> CVE and better passwords
- Stolen KDC key (golden ticket)
  - Compromise the KDC

Asymmetric Crypto Authentication

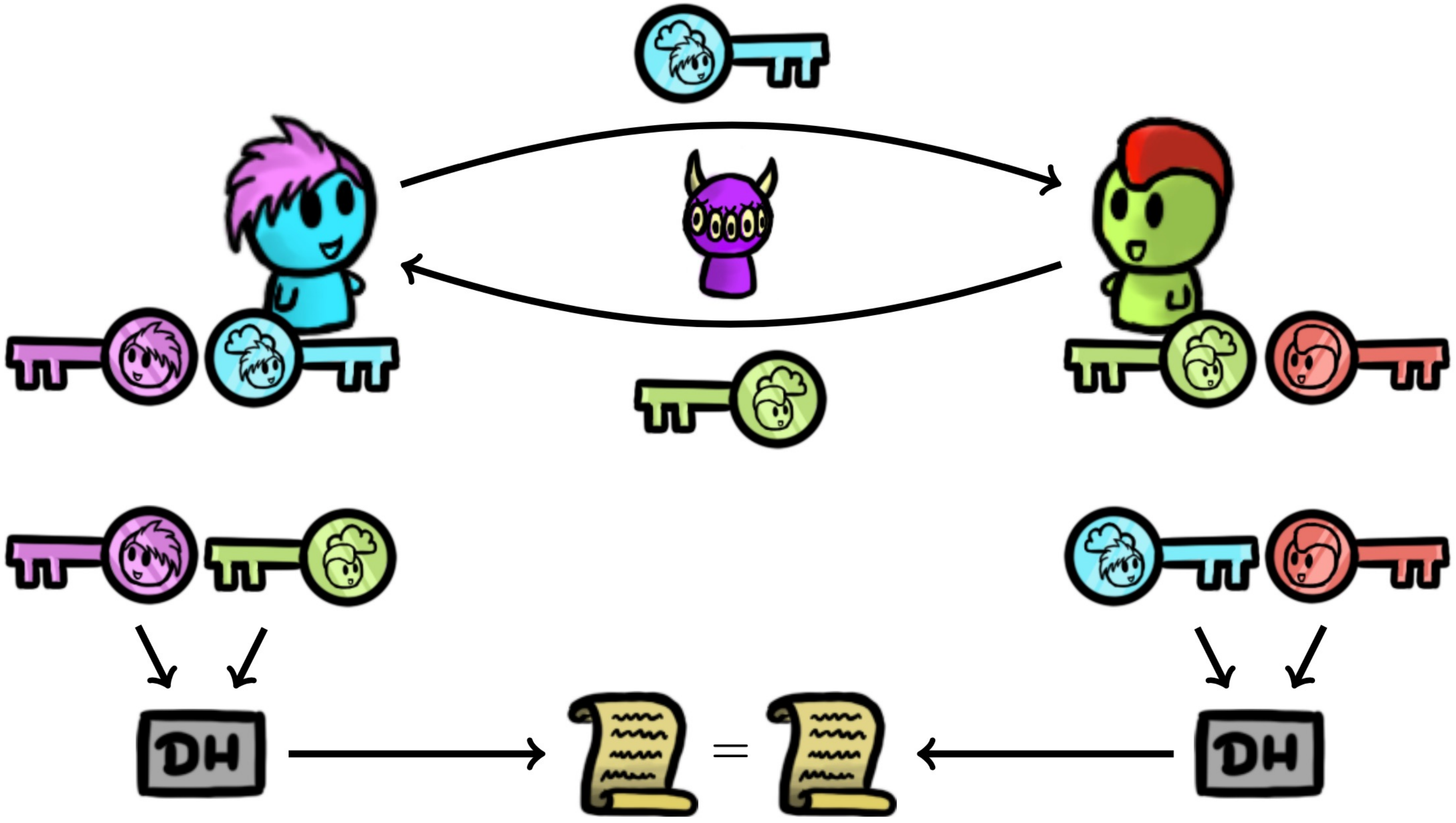
Diffie-Hellman key exchange

A public-key protocol published in 1976 by Whitfield Diffie and Martin Hellman

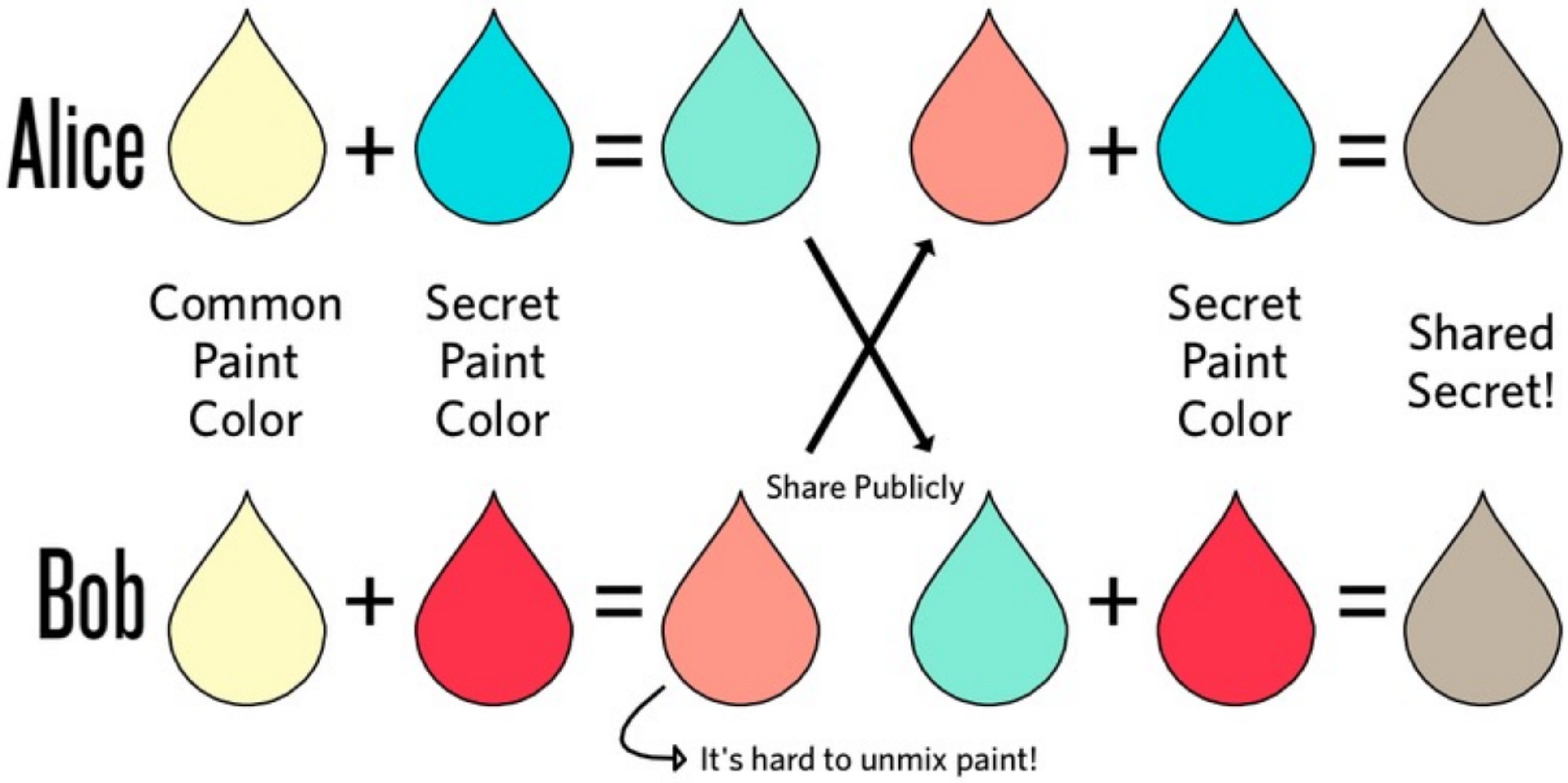
Allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure channel

Key used to encrypt subsequent communications using a symmetric key cipher

Diffie-Hellman key exchange – Visualization



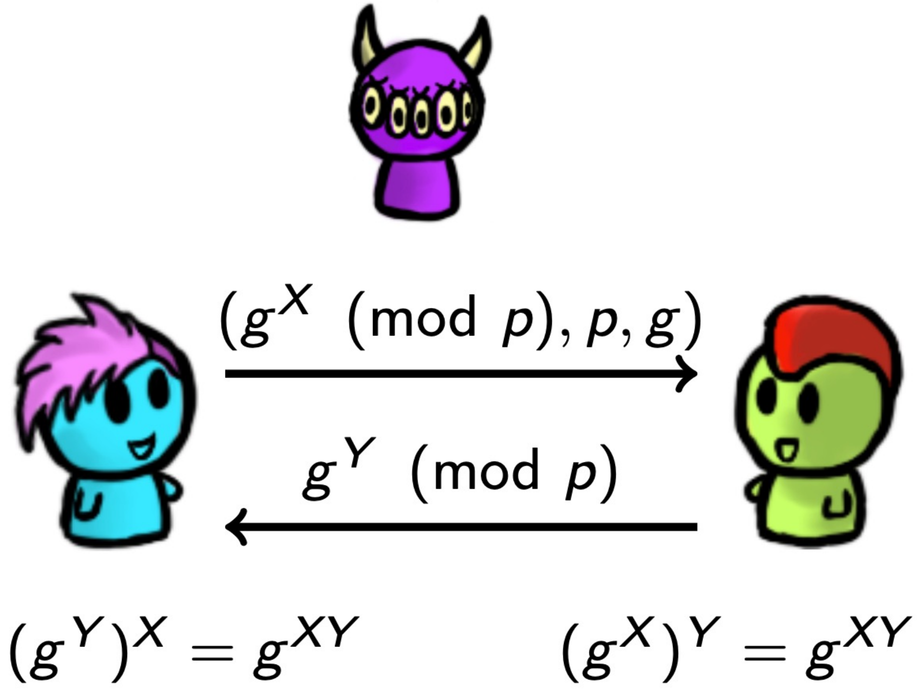
DH as paint!



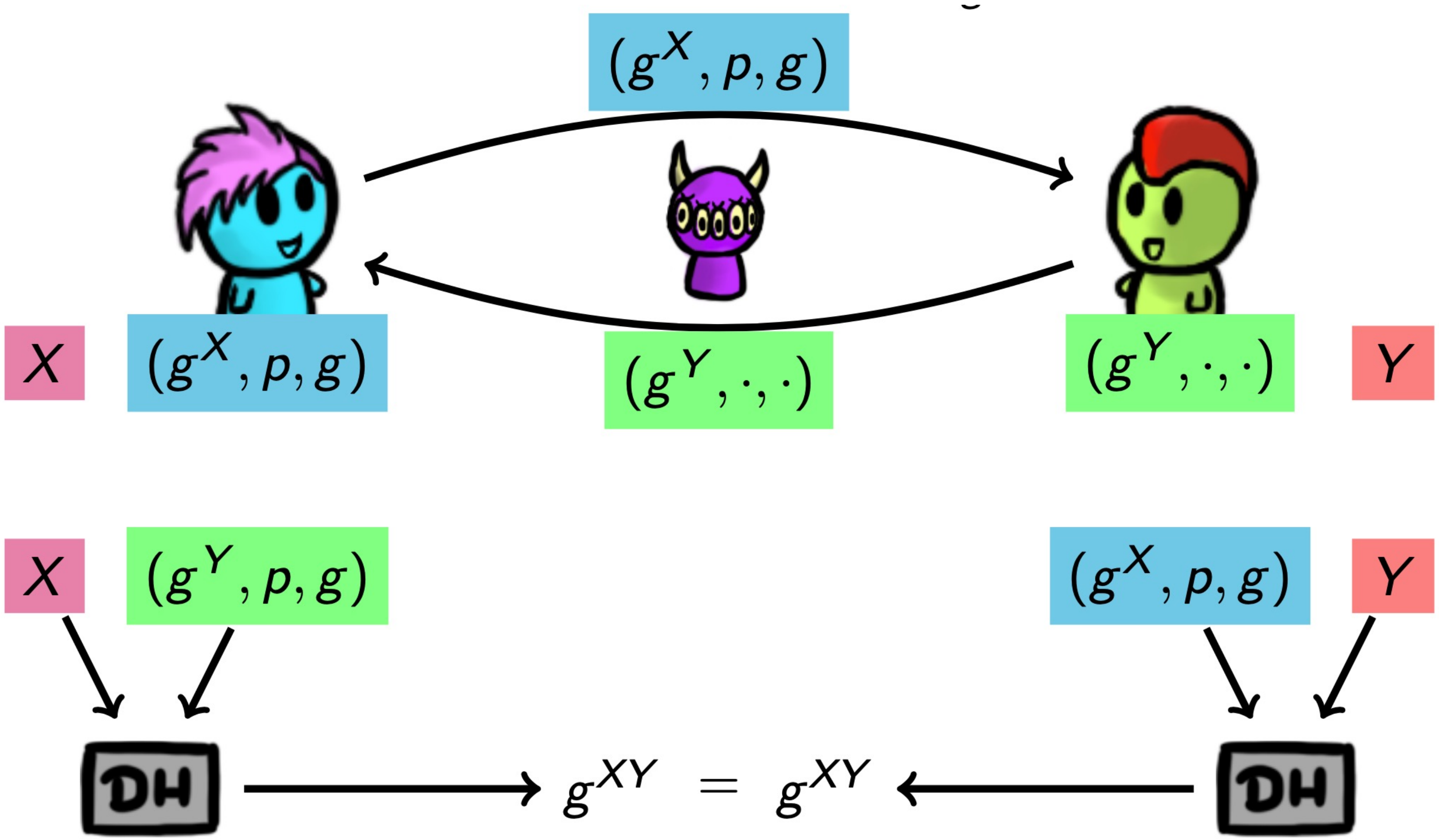
Source: Kelly Robinson

Diffie-Hellman key exchange – The Math

- Alice chooses prime p at random and finds a generator g
- Alice chooses $X \leftarrow_R \{2, 3, \ldots, p-2\}$ and sends $A = g^X \pmod{p}$ to Bob, together with p and g
- Bob chooses $Y \leftarrow_R \{2, 3, \ldots, p-2\}$ and sends $B = g^Y \pmod{p}$ to Alice
- Alice and Bob both compute $s = g^{XY} \pmod{p}$
  - Alice does that by computing $B^X \pmod{p}$
  - Bob does that by computing $A^Y \pmod{p}$
- Now they share a common secret s which can be used to derive a symmetric key



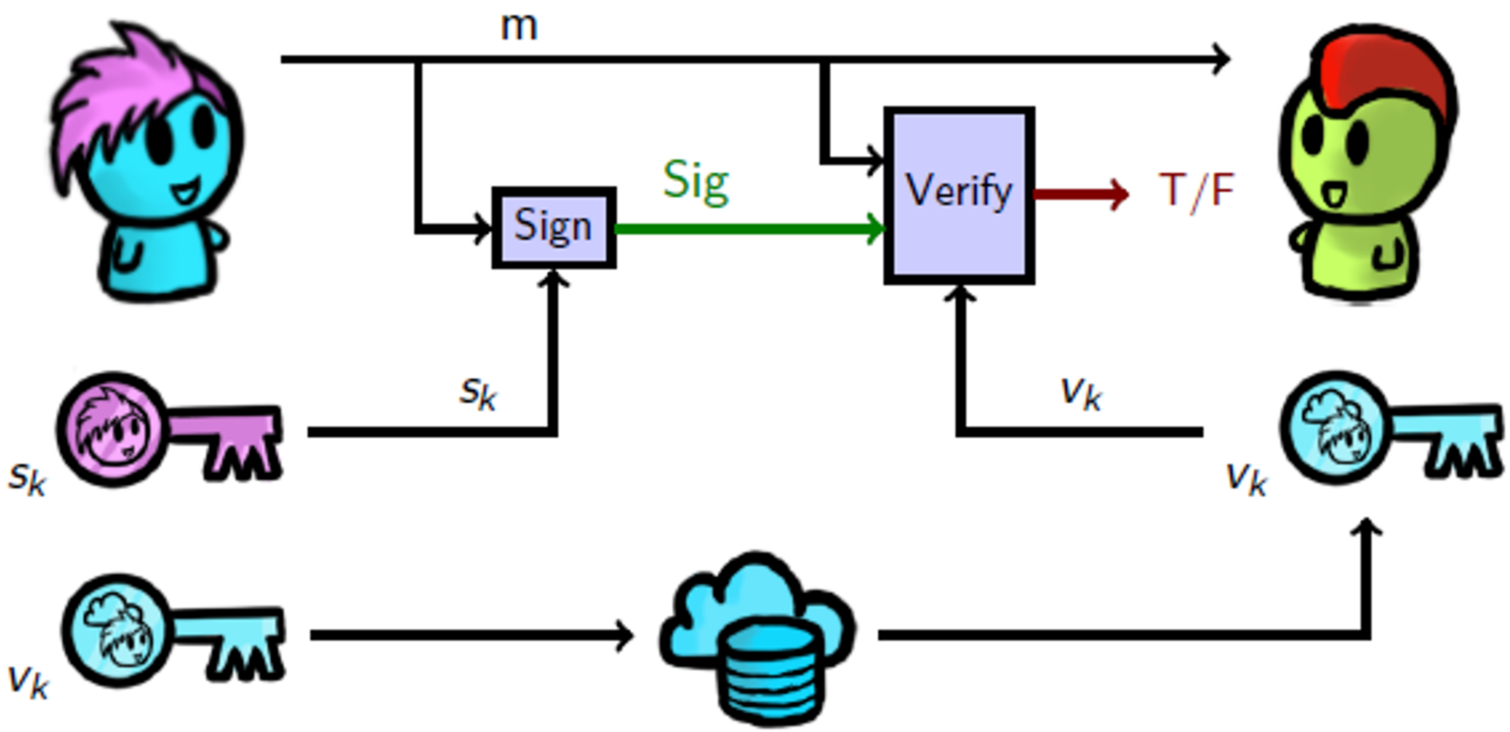
Diffie-Hellman key exchange – Altogether



What's the Problem!

- Authentication!
- Need to verify the public keys!

Recall, Digital Signatures

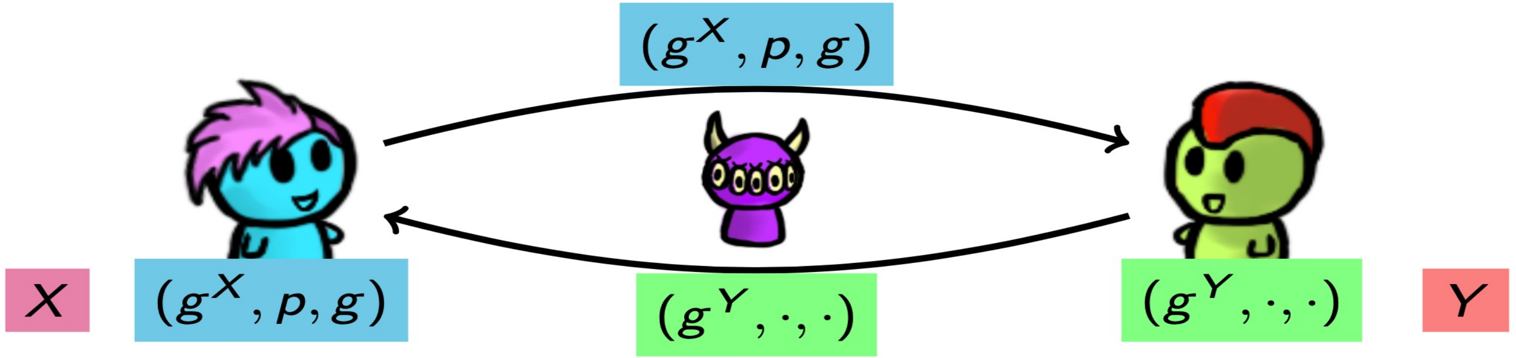


The Key Management Problem

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

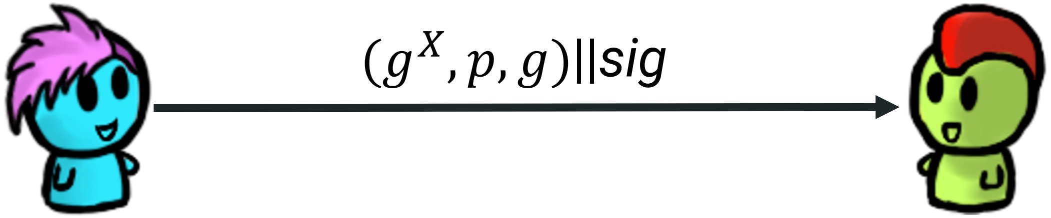
A: By having each other's verification key!

Before



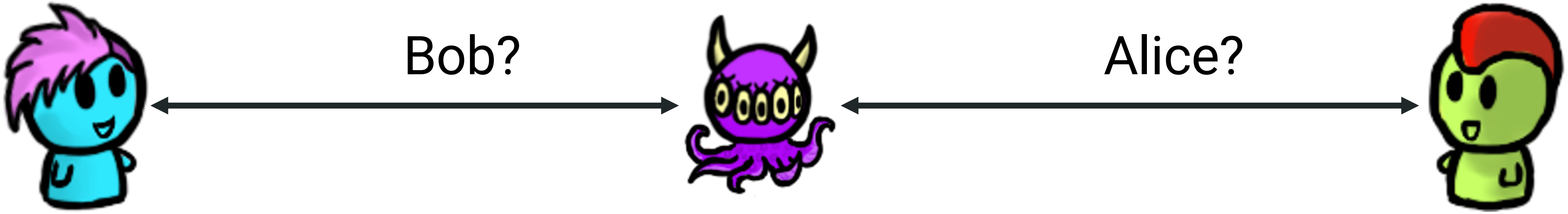
After

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



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

The Key Management Problem

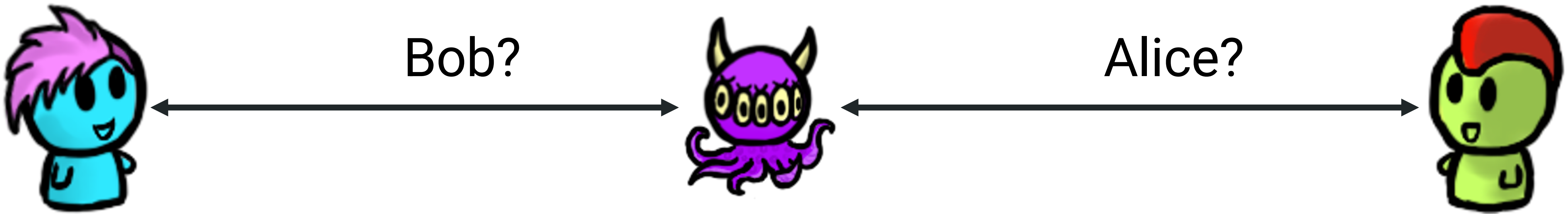


Q: How can Alice and Bob be sure they're talking to each other?

A: By having each other's verification key!

Q: But how do they get the keys…

The Key Management Problem…Solutions?



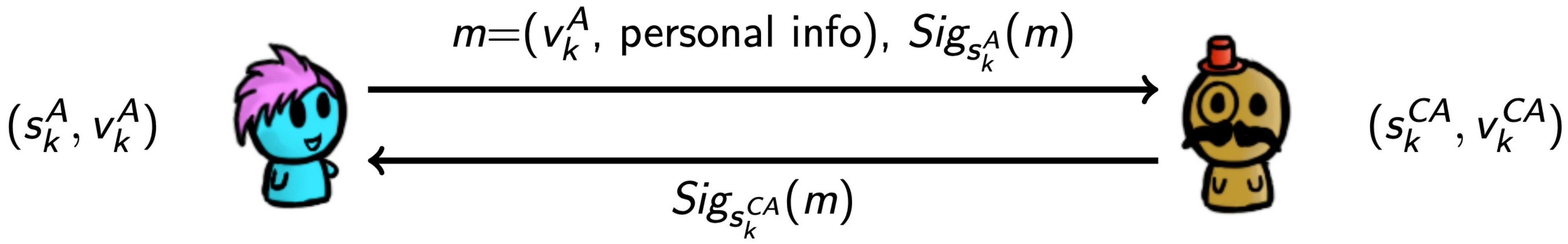
Q: But how do they get the keys…

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)

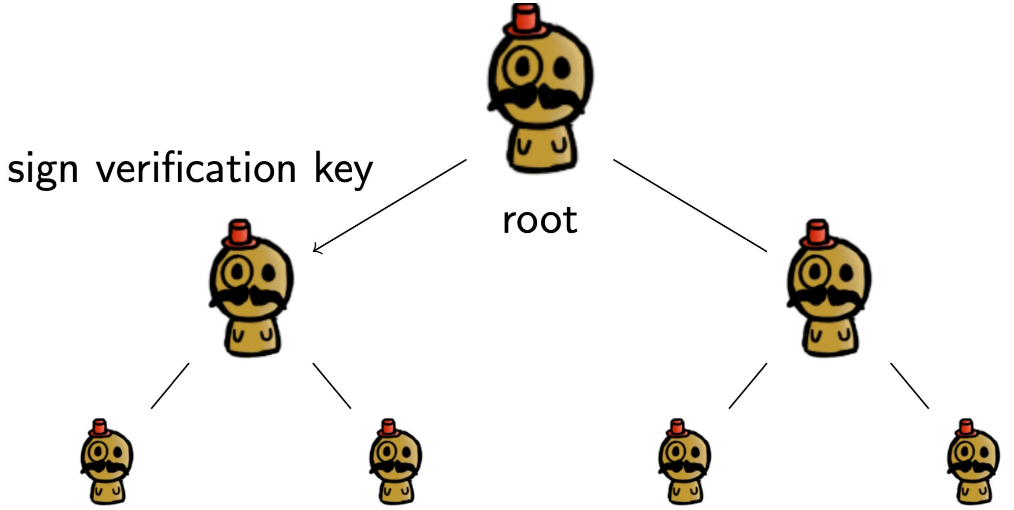
Certificate Authorities (CAs)



- 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

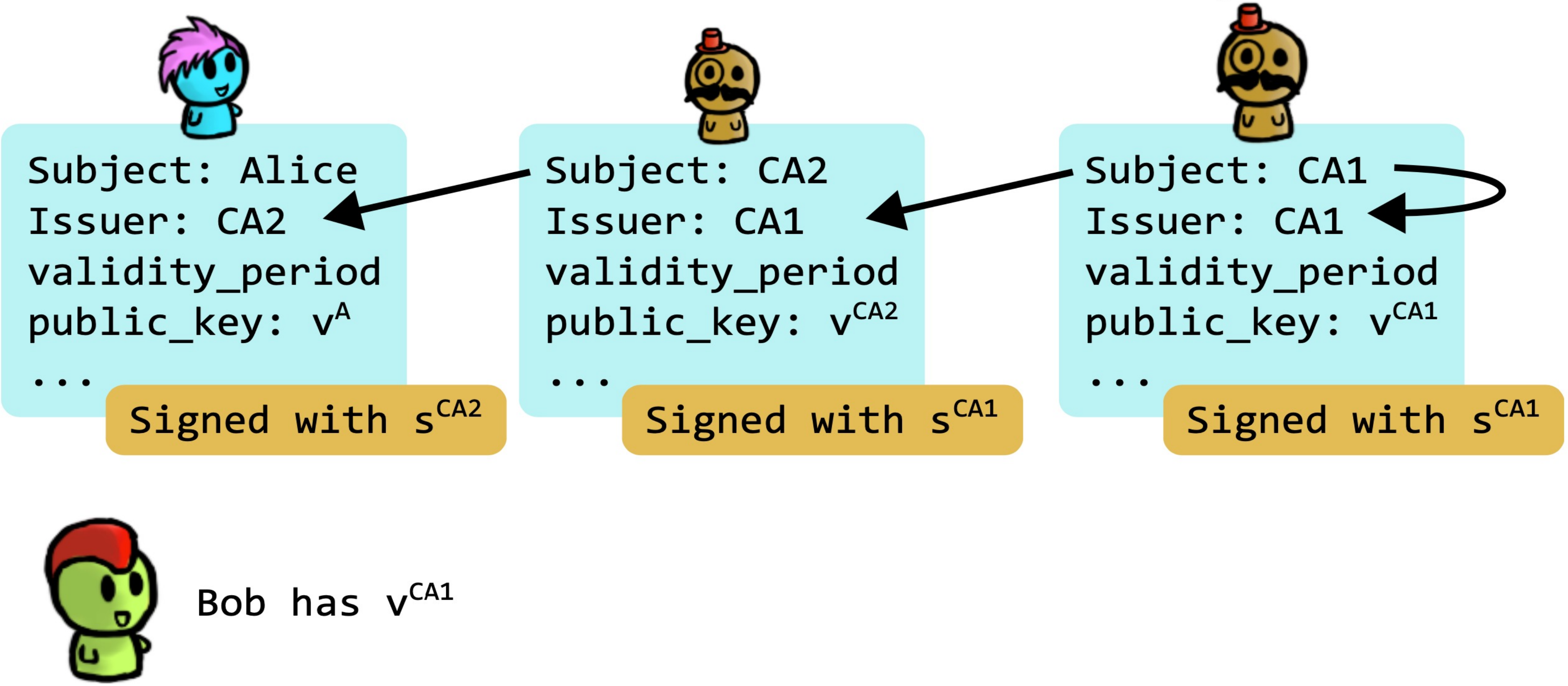
Certificate Authorities

- 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.

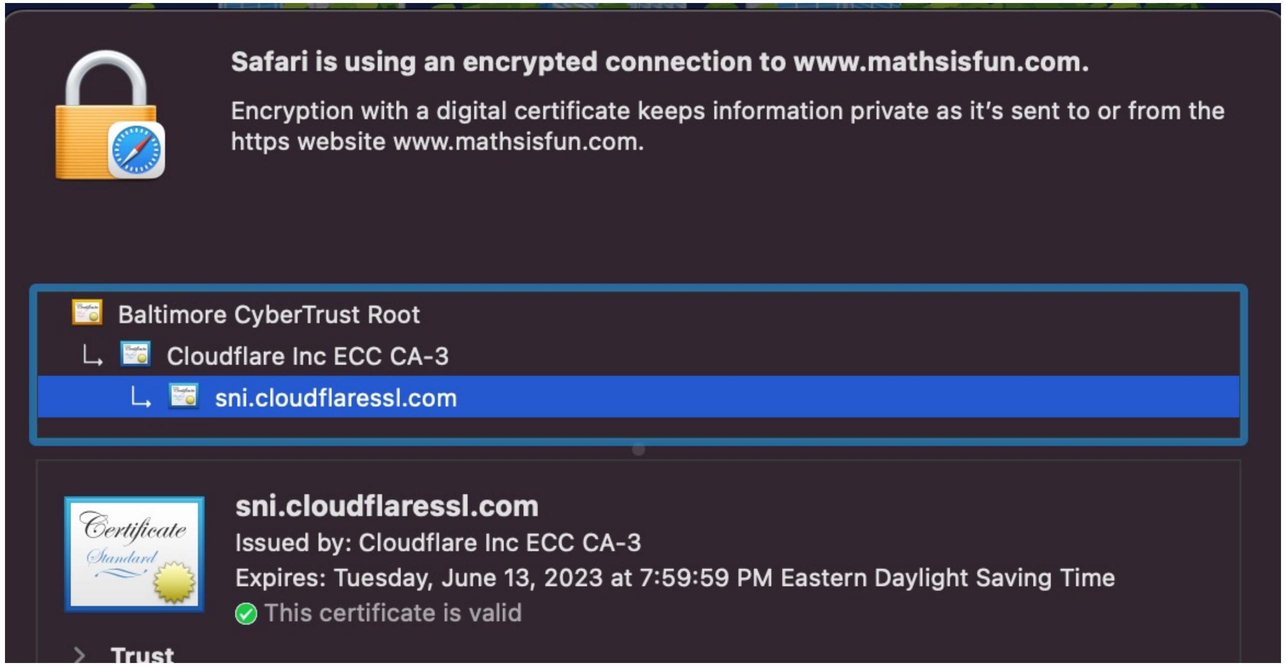


CAs on the web

- 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	43.4%	48.9%
2	DigiCert	16.6%	18.7%
3	Sectigo (Comodo Cybersecurity)	13.8%	15.5%
4	Let's Encrypt	7.2%	8.2%
5	GoDaddy	5.4%	6.1%
6	GlobalSign	2.4%	2.7%

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
Certificate Standard
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 ...

A Note on PAKEs

How do we authenticate passwords?

- Typically send password in plain over a secure channel (TLS)
- Server’s store only hash’s (with salt)
    - Will see the password at least briefly
- We are good at crypto, can it help?

Password-authenticated key exchange (PAKE)

- A special form of cryptographic key exchange protocol introduced by Bellovin and Merritt
- Designed to help two parties (Bob and Alice) agree on a shared encryption key using a password
  - Balanced: Both parties have password
  - Augmented: Only client (server does not)
- Problem: Hard to get it right!

Goals of PAKEs

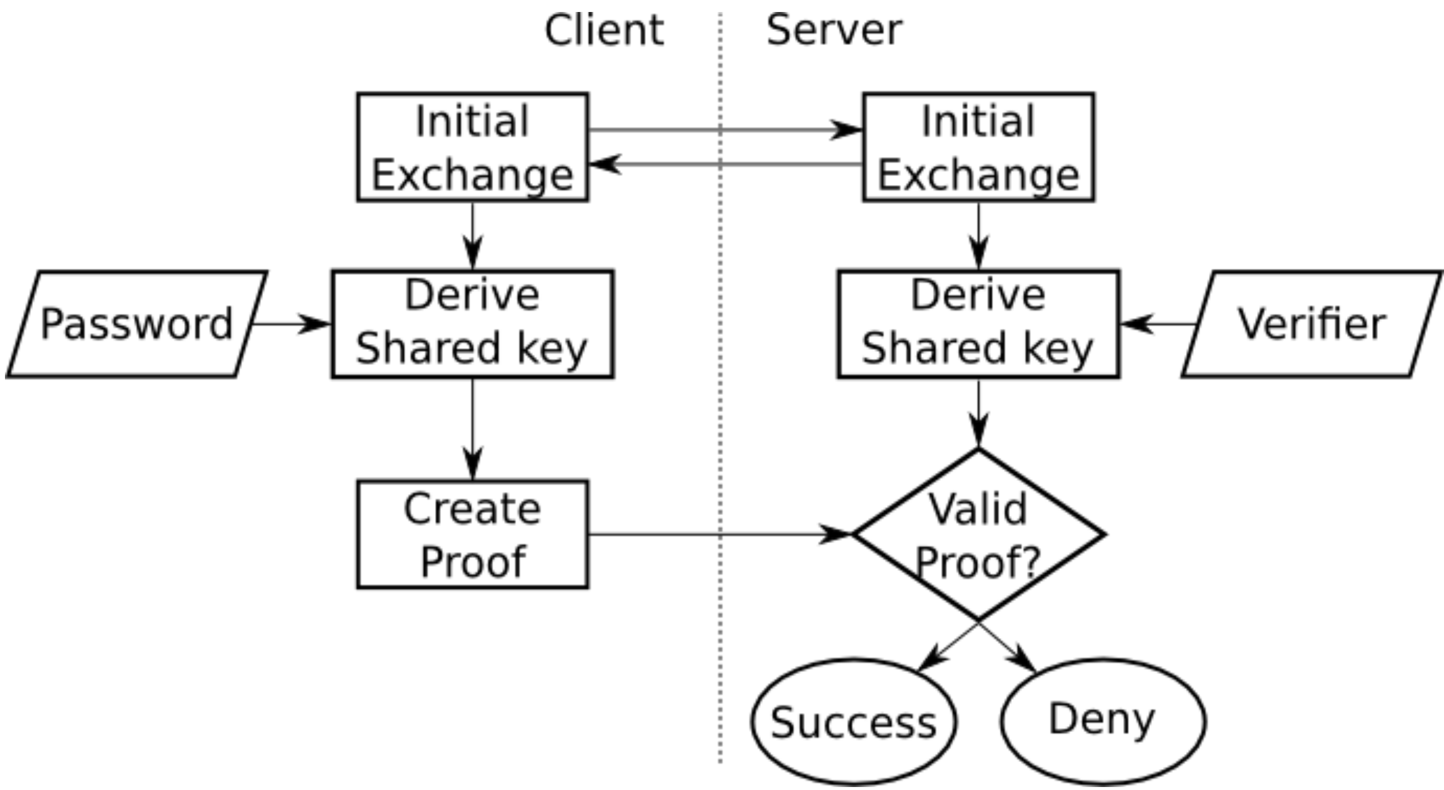
- The secret keys will match if the passwords match, and appear random otherwise.
- Participants do not need to trust third parties (in particular, no Public Key Infrastructure)
- The resulting secret key is not learned by anyone not participating in the protocol - including those who know the password.
- The protocol does not reveal either parties' password to each other (unless the passwords match), or to eavesdroppers.

Attacks on PAKEs

- Off-line dictionary attack
- On-line dictionary attacks
- Replay attacks
- Implementation Issues
- Entropy!?

Example: SRP

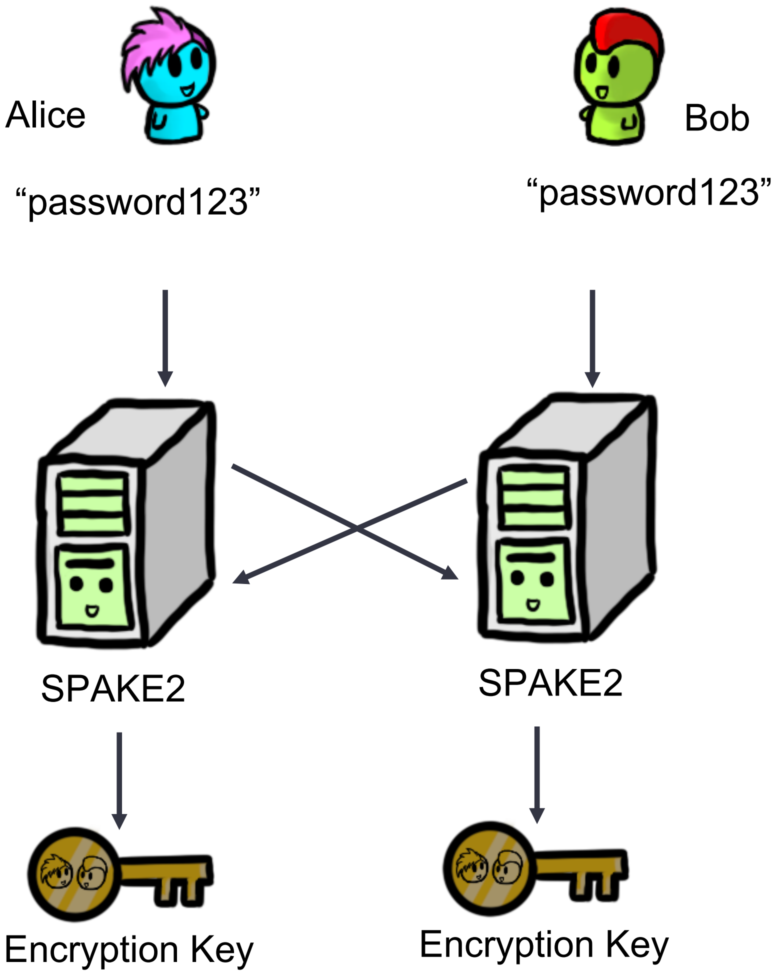
- Early widely deployed PAKEs
  - Apple iCloud!
- Poor security proof
  - On V6a (keeps getting broken)
- Vulnerable to offline dictionary attacks



Example: OPAQUE

- Proposed in 2018
- Has much stronger security proof
- Uses OPRFs to avoid leaking the salt to attacker
- Efficient, works for any hash of passwords on the server
- https://eprint.iacr.org/2018/163.pdf

Example: SPAKE2

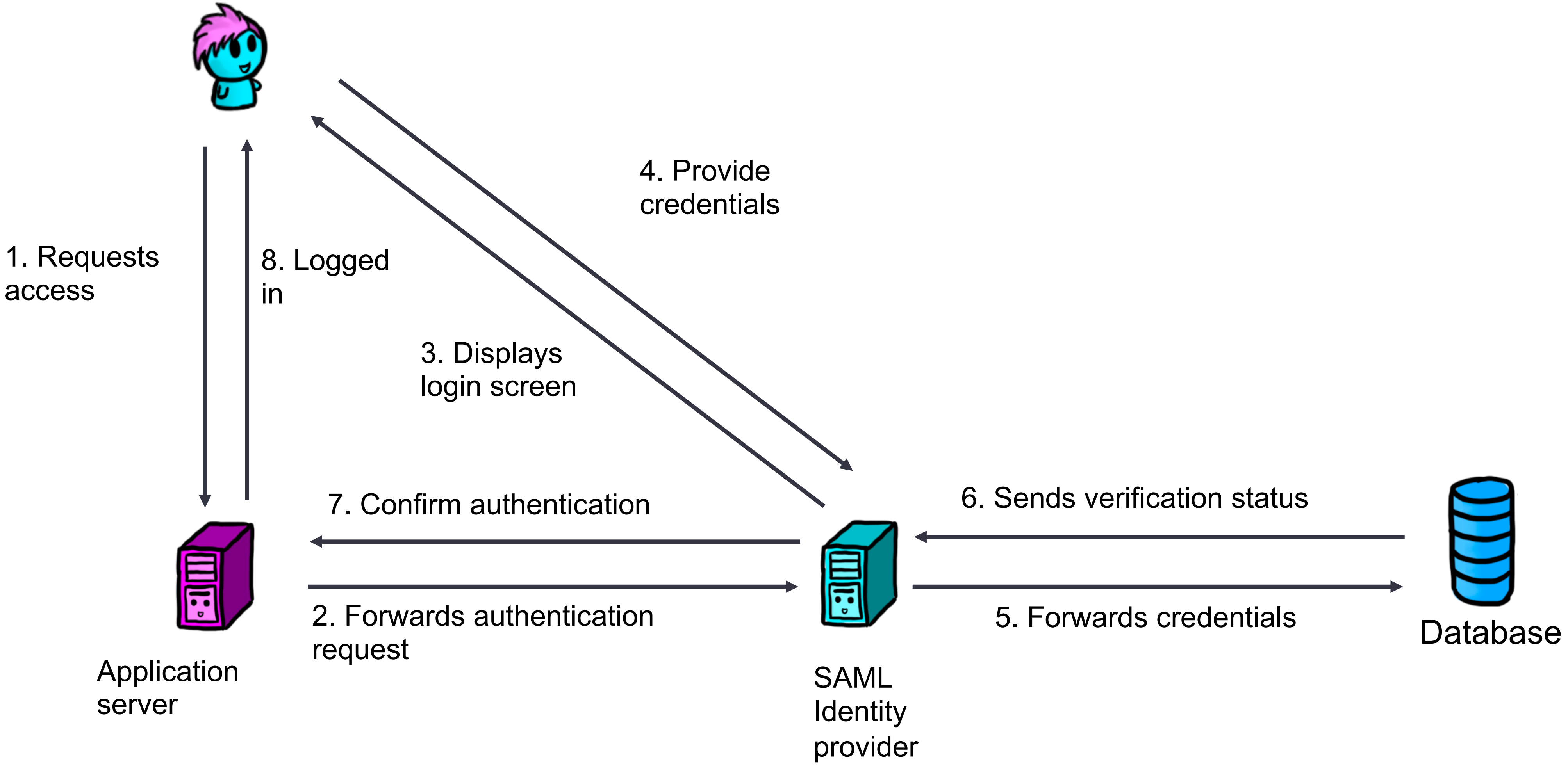


Key Management - SSO

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)

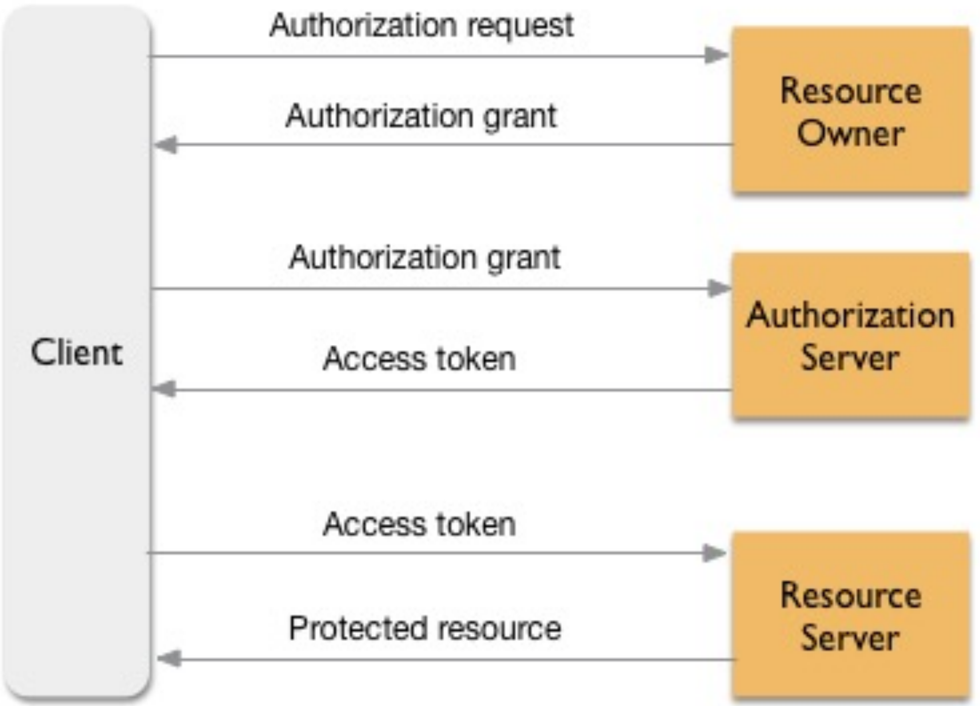


Security Assertion Markup Language (SAML)

- 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

OAuth

- Like SAML it provides a framework and formatting for granting tokens
- Key difference: Authorization not authentication
    - i.e., a set of capabilities not attestation that you are who you say you are
    - Tokens are not tied to you



DNSSEC

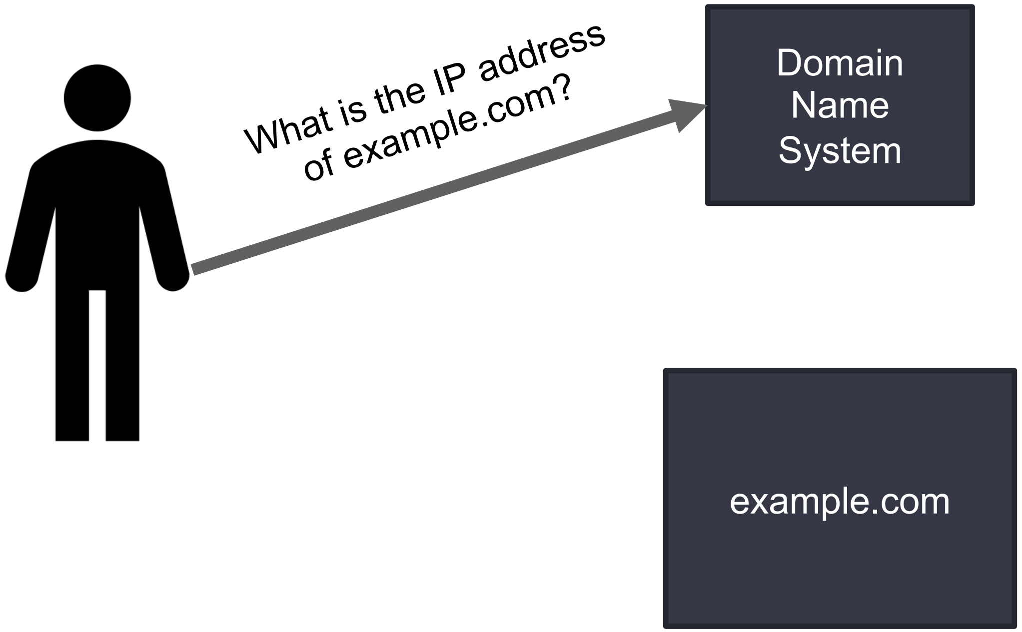
Source: Jason Goertzen and Miti Mazmudar

RECALL, WHAT IS DNS?

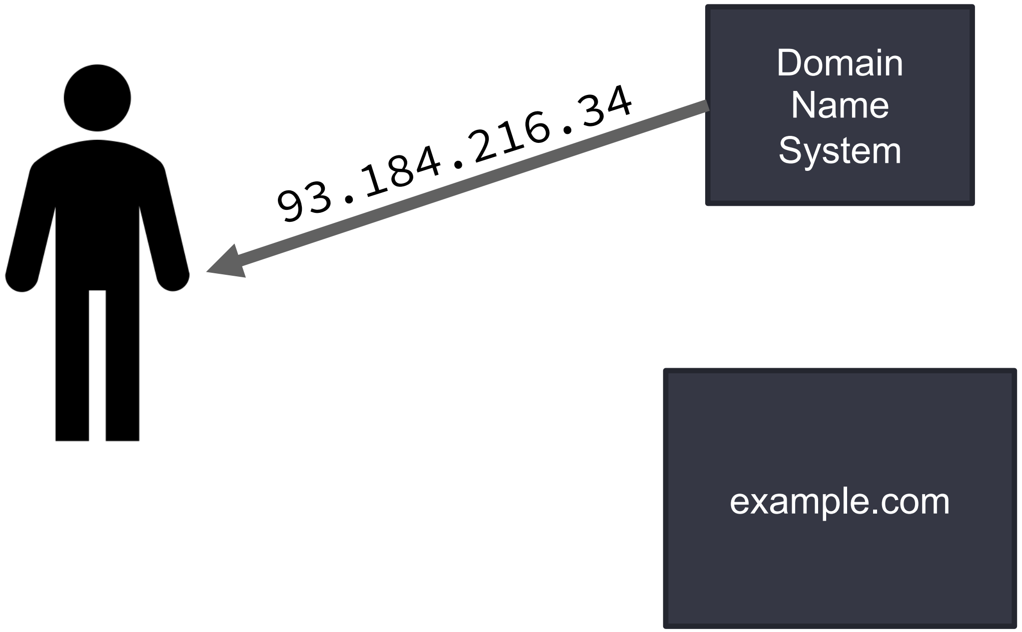
- The internet uses IP addresses to determine where to send messages
- IP addresses are difficult for people to remember!
- The Domain Name System is responsible to translating something easy for a human to remember into IP addresses

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

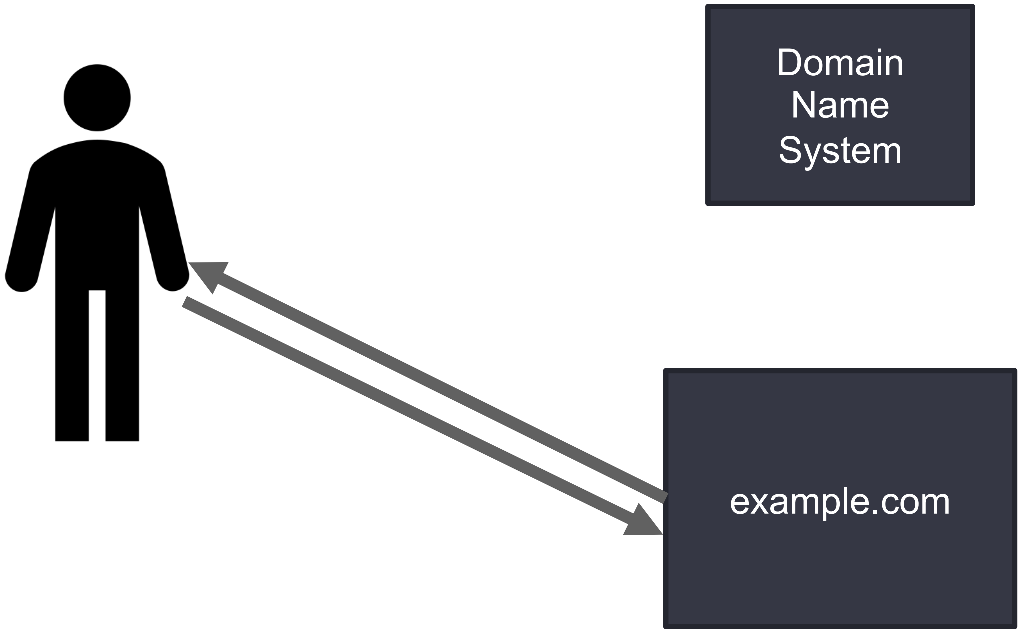
WHAT IS DNS?


What is the IP address of example.com?
Domain Name System
example.com

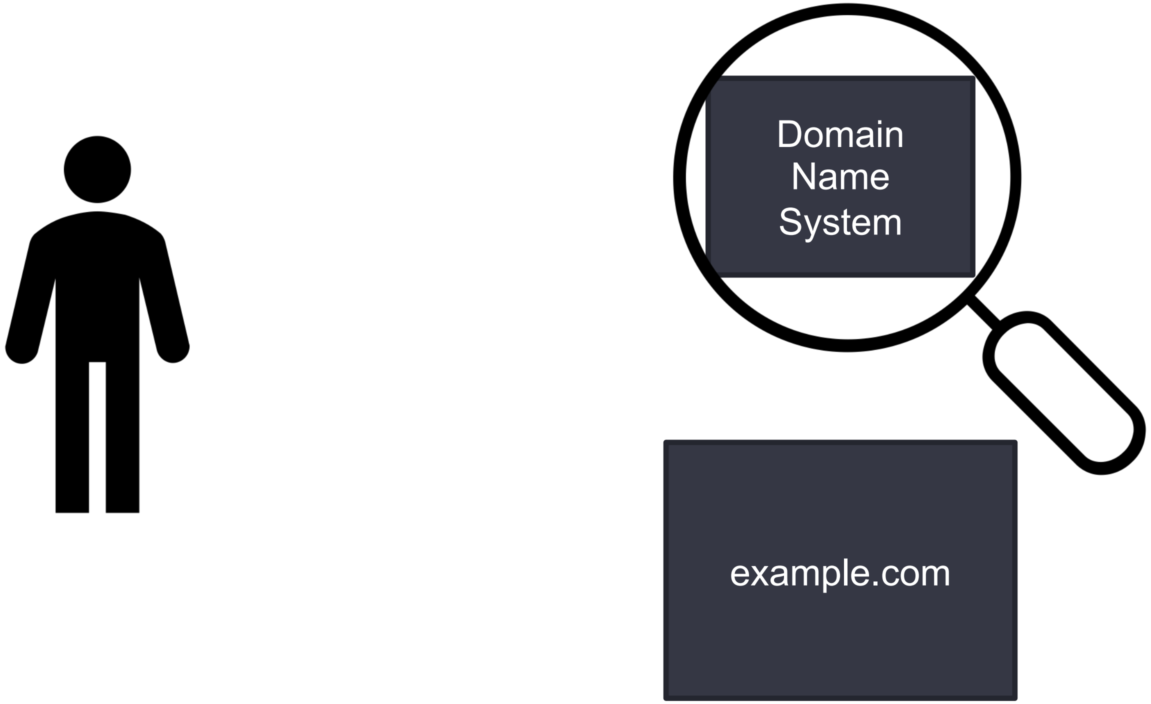
WHAT IS DNS?



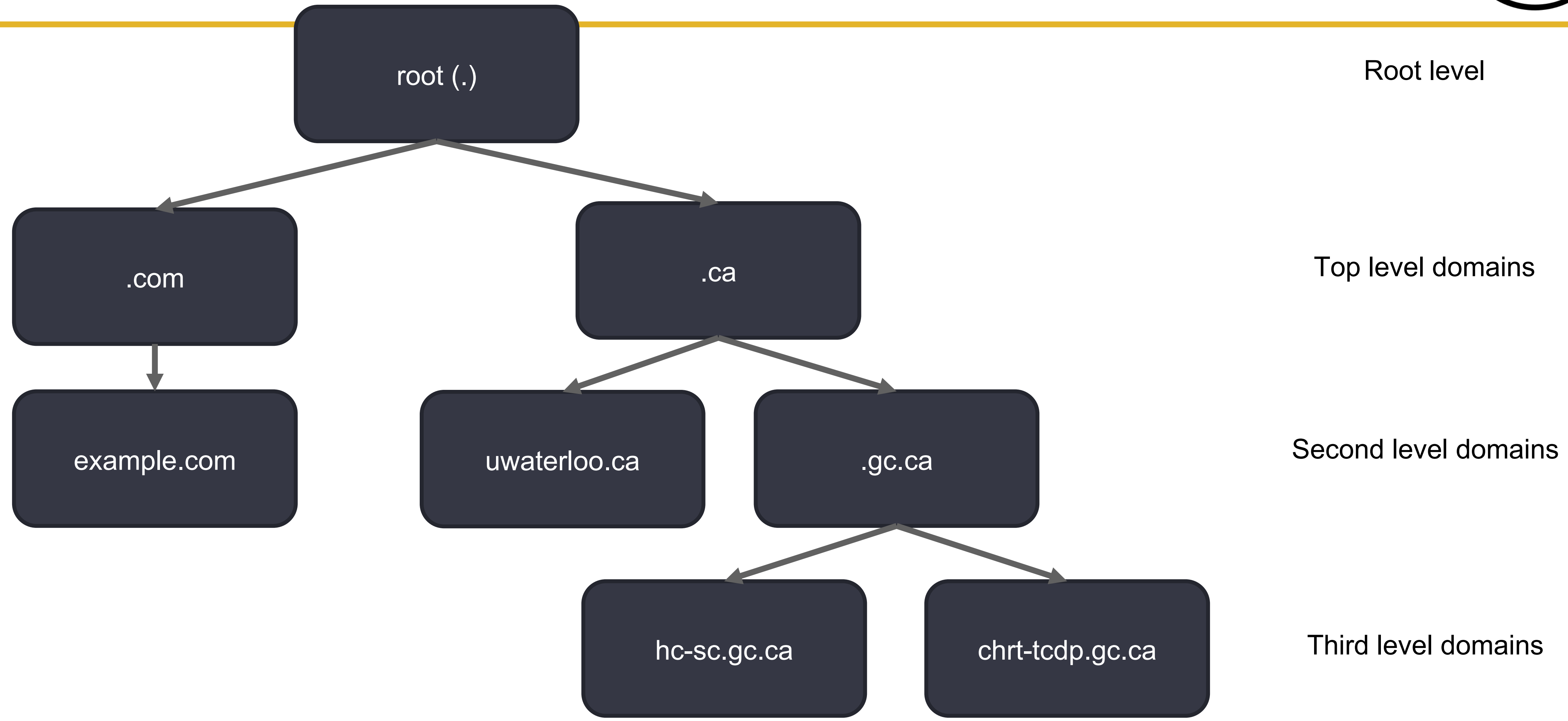
WHAT IS DNS?



WHAT IS DNS?



DNS IS BROKEN UP INTO ZONES



Domain Name System (DNS) - *dig* command

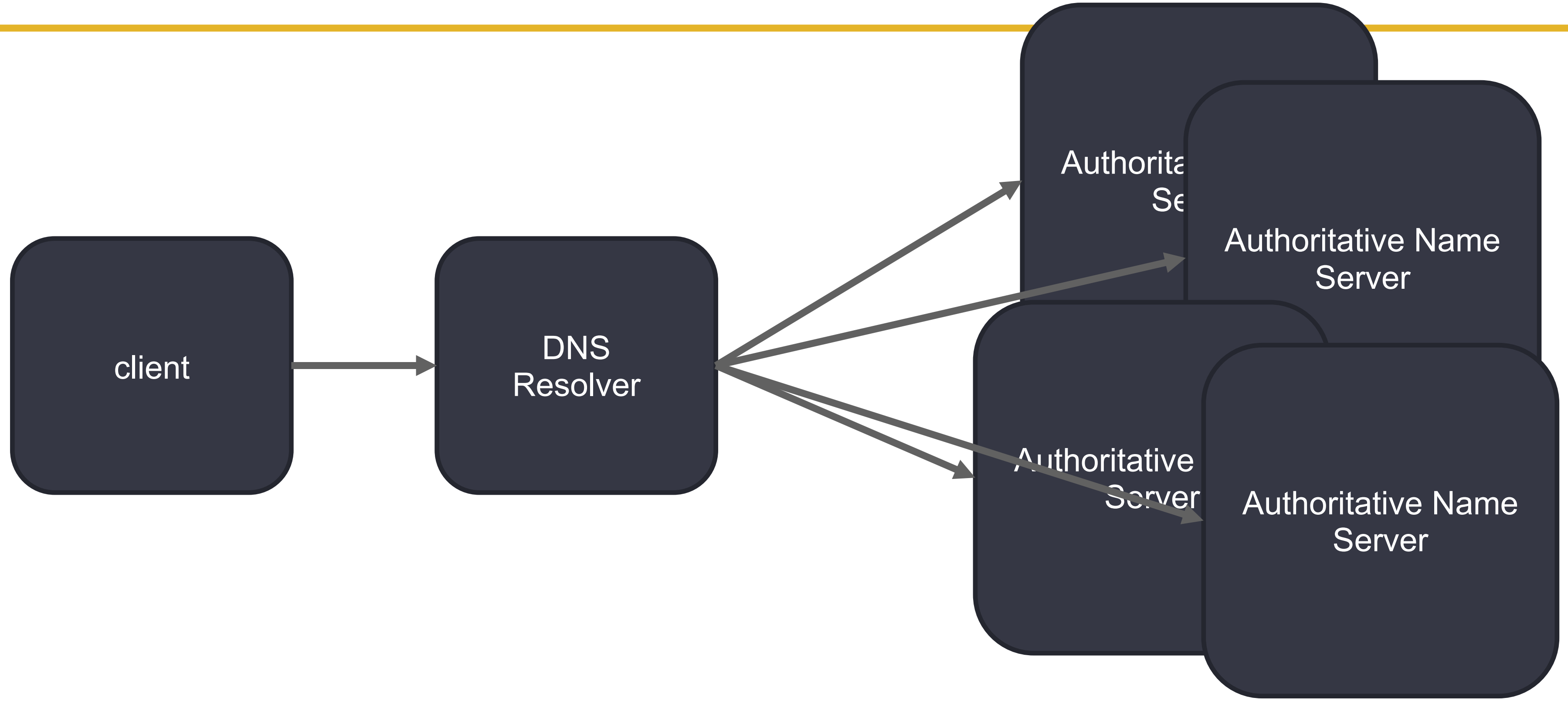
```
; <<>> DiG 9.16.15 <<>> crysp.uwaterloo.ca
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34154
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; 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`

ZONES CONTAIN RESOURCE RECORDS

example.com. 57094 IN AAAA	2606:2800:220:1:248:1893:25c8:1946
example.com. 57047 IN A	93.184.216.34
example.com. 57094 IN NS	b.iana-servers.net.
example.com. 57094 IN NS	a.iana-servers.net.

CLIENTS RARELY QUERY DIRECTLY



DNS protocol

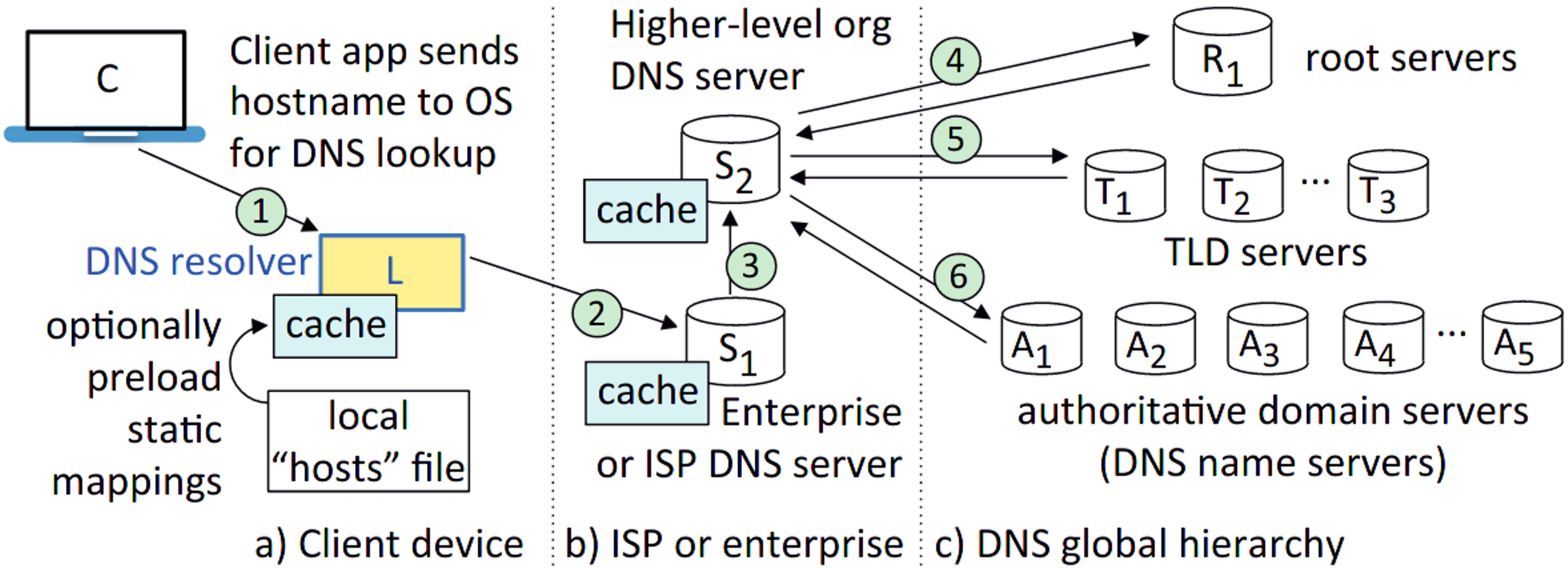
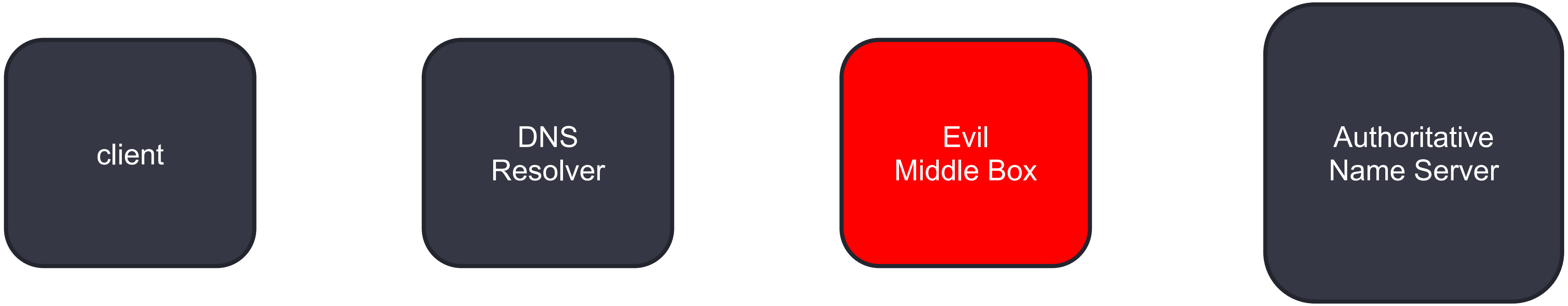


Figure 11.7: DNS name resolution and query hierarchy (simplified).

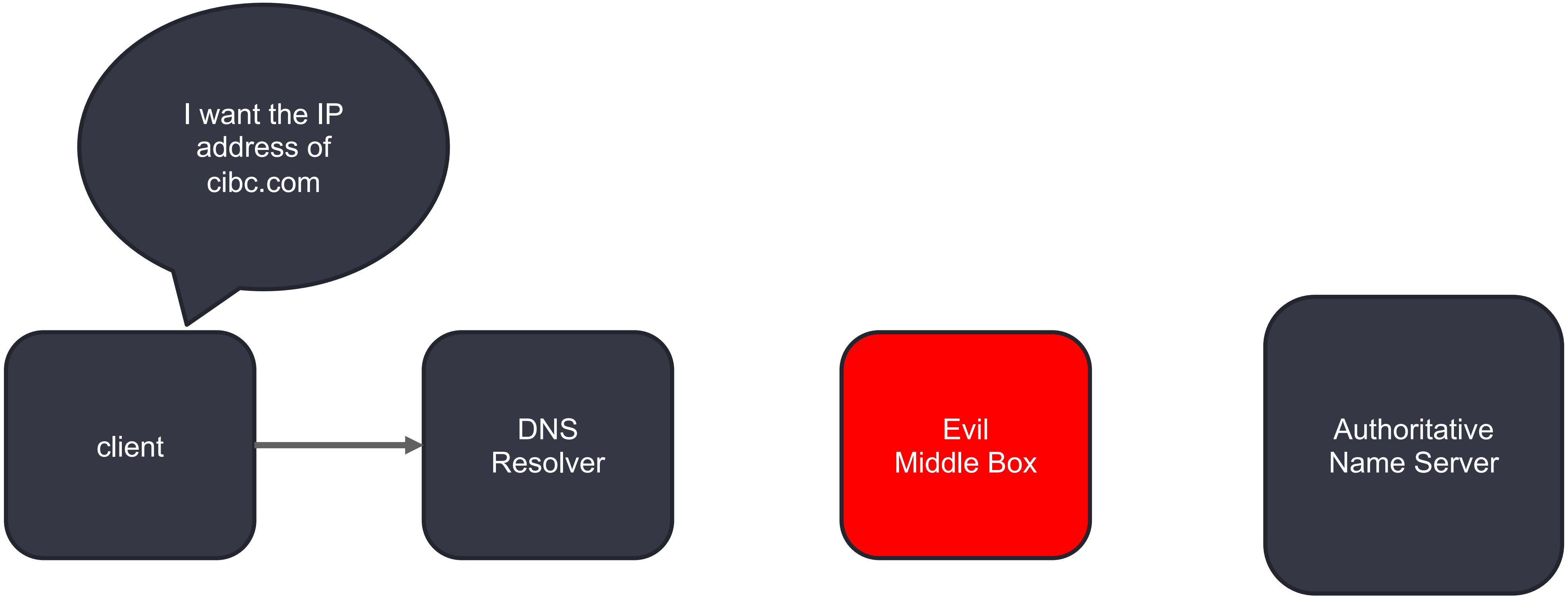
PROBLEM WITH DNS

- Designed with no integrity projection



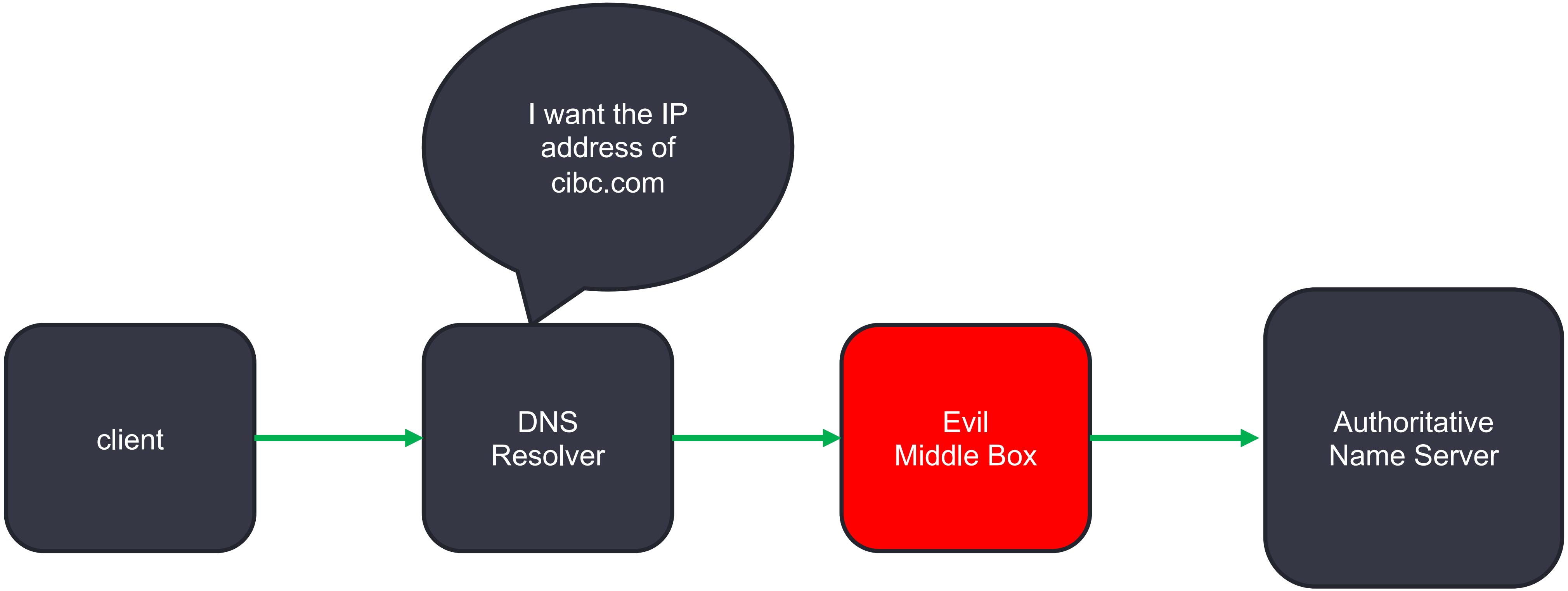
PROBLEM WITH DNS

- Designed with no integrity projection



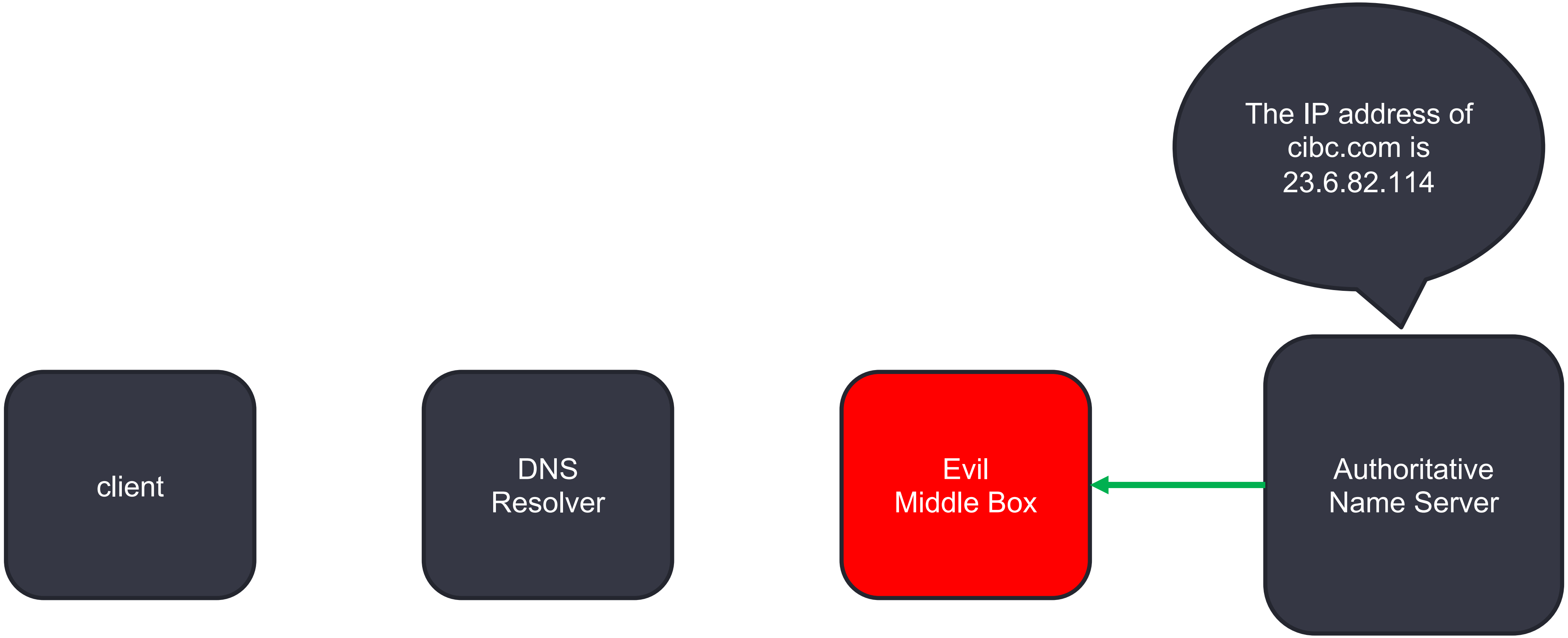
PROBLEM WITH DNS

- Designed with no integrity projection



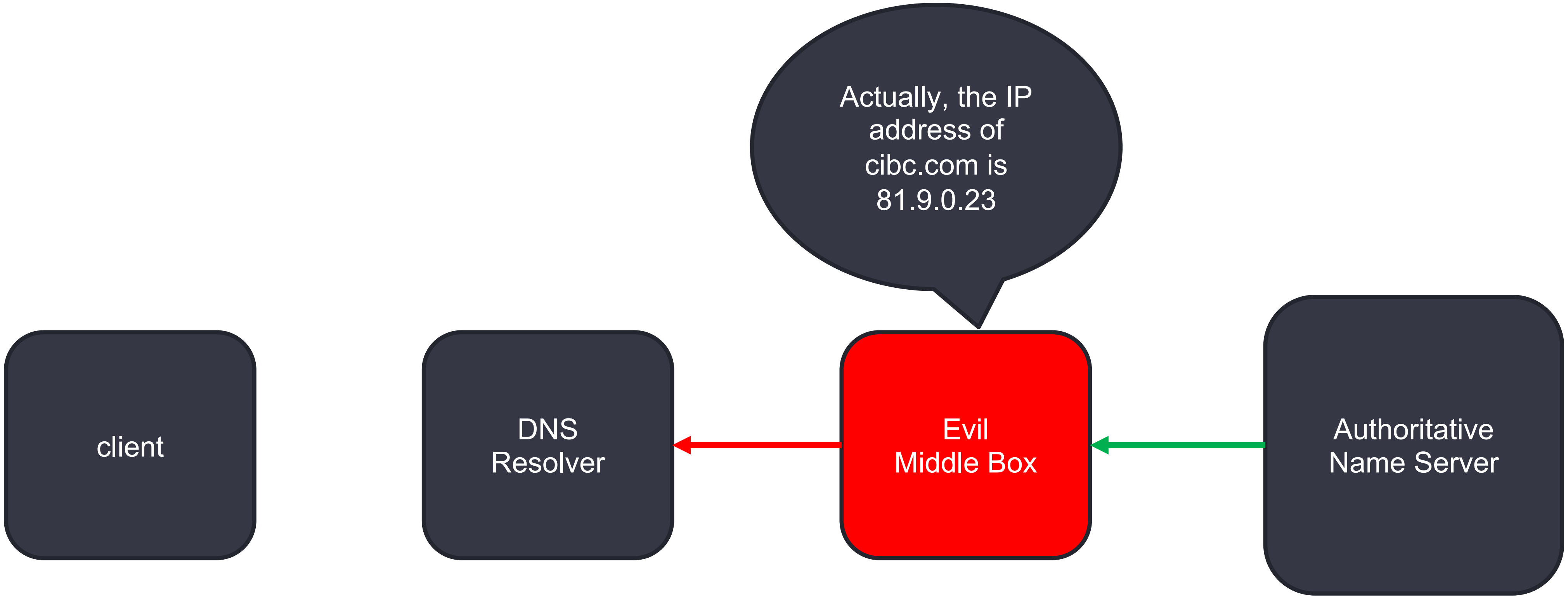
PROBLEM WITH DNS

- Designed with no integrity projection



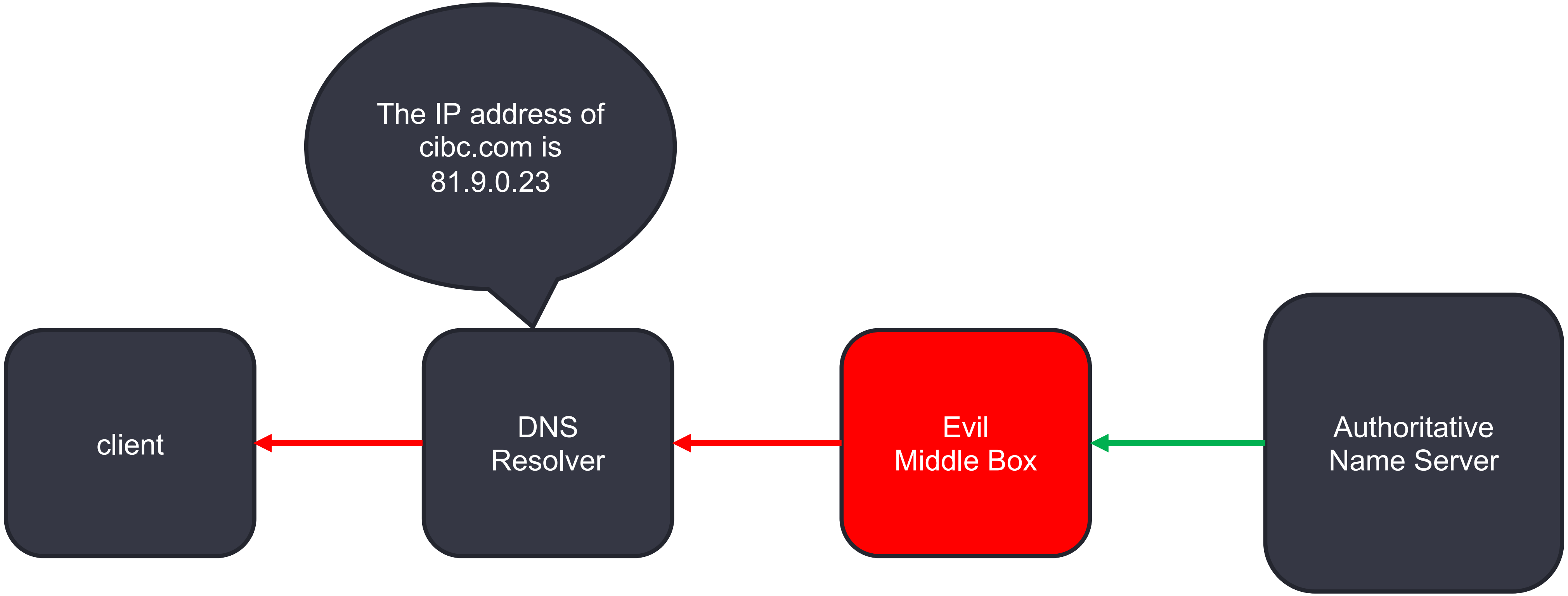
PROBLEM WITH DNS

- Designed with no integrity projection



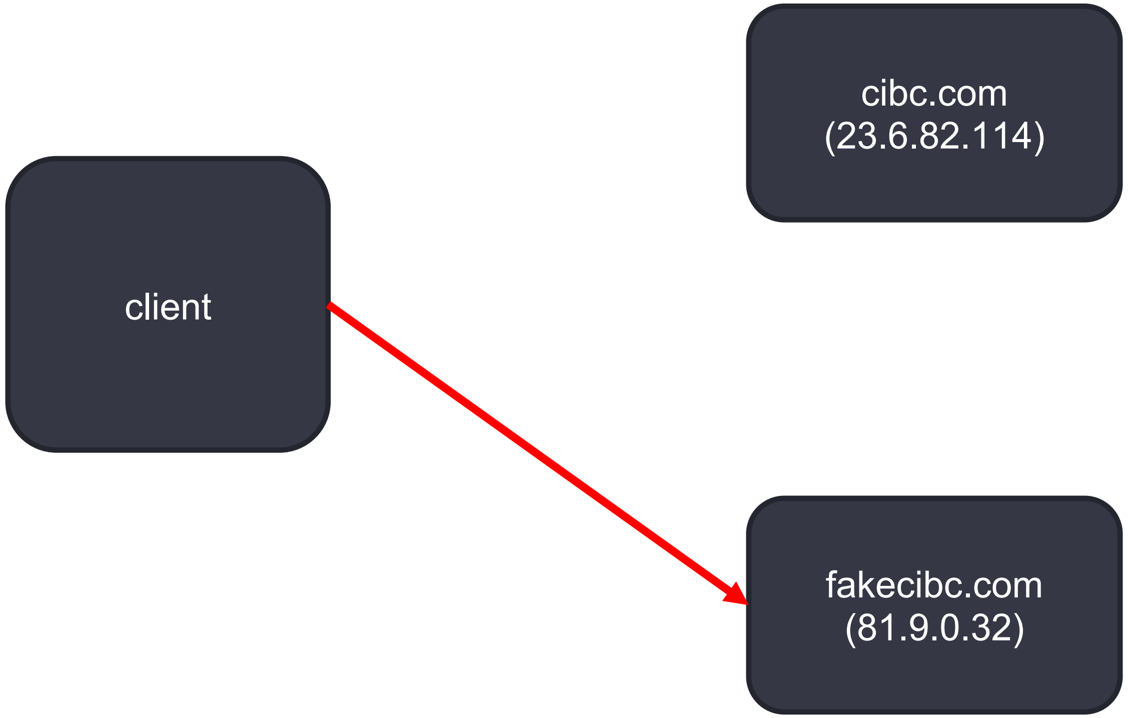
PROBLEM WITH DNS

- Designed with no integrity projection

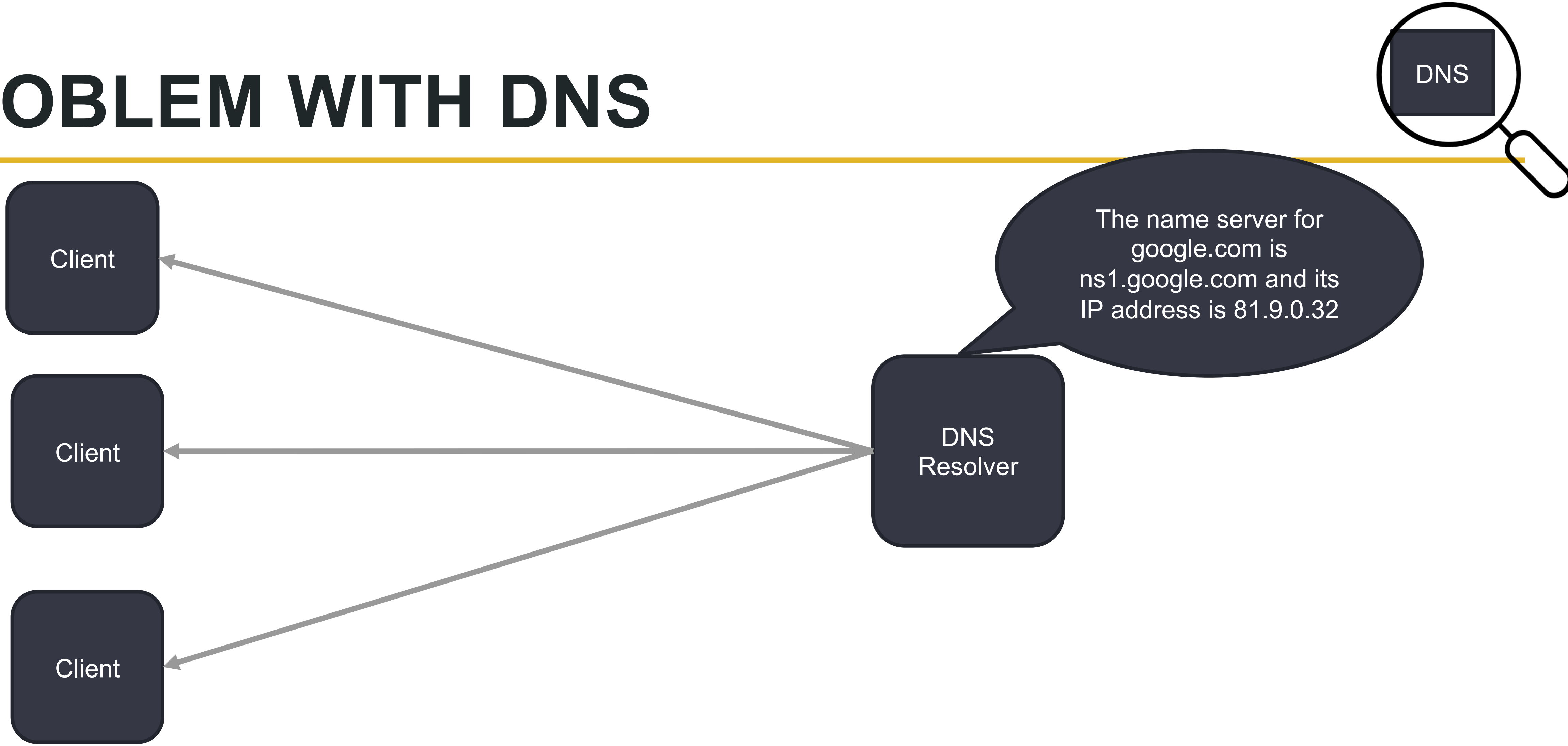


PROBLEM WITH DNS

- Designed with no integrity projection



PROBLEM WITH DNS



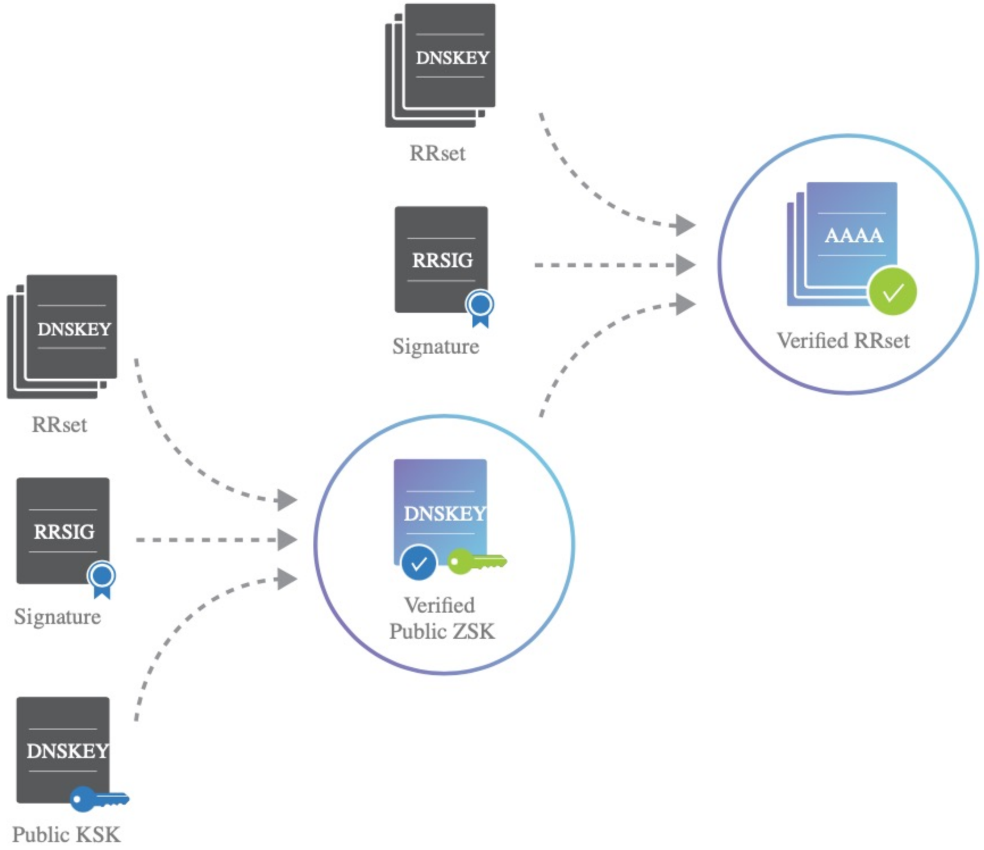
SOLVING DNS INTEGRITY

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

- New records added to DNSSEC signed zone
- Sets of records (RRSets) are signed, rather than individual records
- Have two keys:
  - Key Signing Key : kept in trusted hardware, hard to change
  - Zone Signing Key : changed more often, smaller, used for records

Verification Procedure

- Assuming you trust the public KSK
- Use it to verify the RRset containing the ZSK
- Then use ZSK to verify the records



Source: cloudflare blog

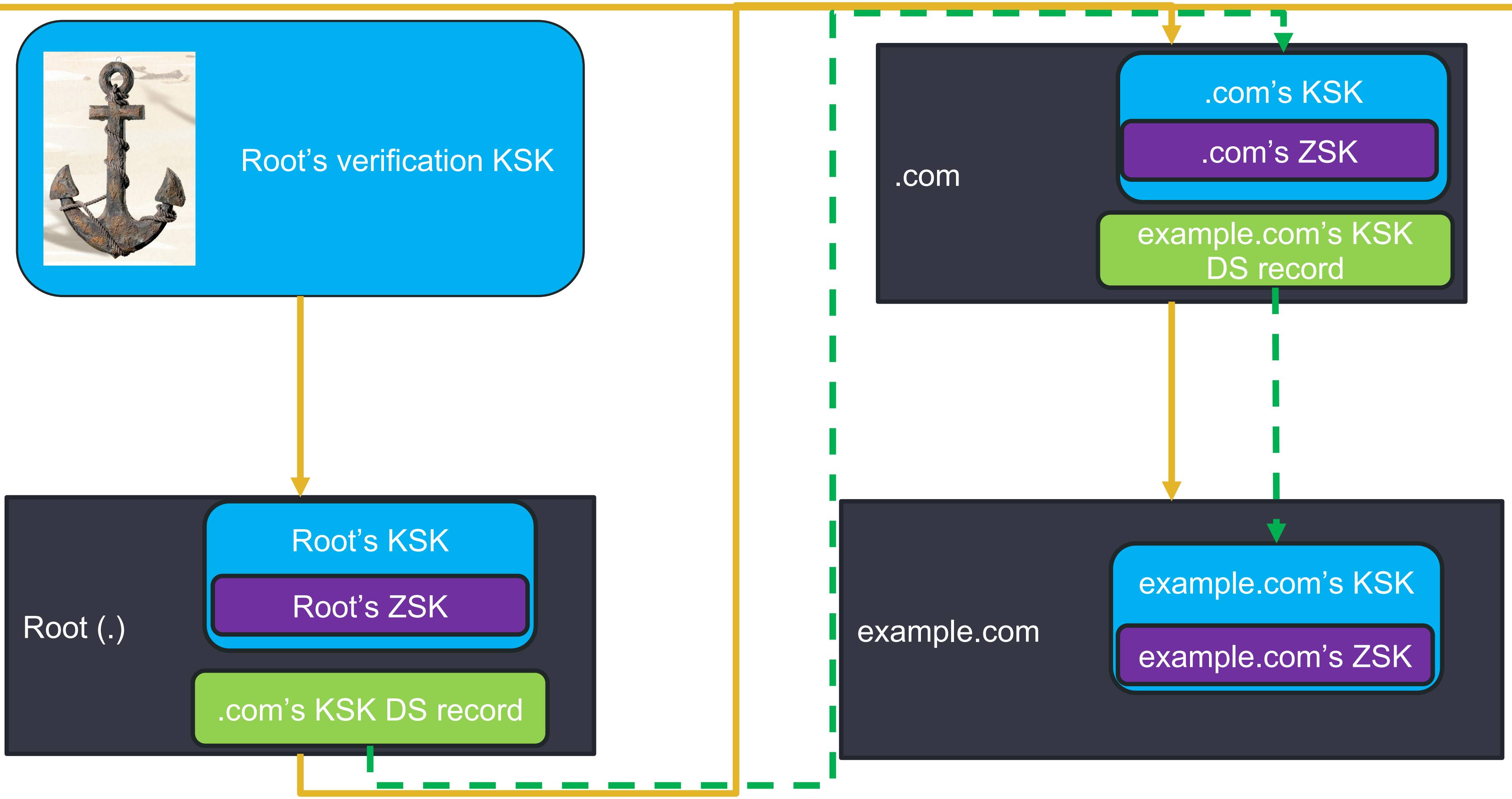
HOW DO WE MAINTAIN KEY INTEGRITY?

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
- Each zone's parent zone contains a "Delegate signer" (DS) record which is used to verify zone's KSK
    - Hash of KSK

HOW DO WE MAINTAIN KEY INTEGRITY?



SOLVING DNS INTEGRITY

example.com.	86379	IN	A	93.184.216.34
example.com.	86328	IN	RRSIG	A 8 2 86400 20221120061546 20221030132217 59208 example.com. rZvjehQxdT5pJ4cw+o1y/BYmLkBLuqzjFaEOn9773Bhywt4qhKmME8DK oKD4yLjYJYFaqhUNCYb+iimCTdK+9+3UjJ35gRIDC3kuZ9hogtCoLBnt ltfgFwLQ0mdye8iH/FDDVKTm+CAz3UMfcwNzNahvg4BOnZ04HqnpZcWW pu4=
example.com.	73820	IN	NS	a.iana-servers.net.
example.com.	73820	IN	NS	b.iana-servers.net.
example.com.	86237	IN	RRSIG	NS 8 2 86400 20221122065049 20221101032317 59208 example.com. Uit3UXCeClM+iwVkc2lX8n5A1O0CD9mH8rsTSfcsjQaZD9y54q7bT2mM cGMaiyjj/sODGKLNvbFKLEgHgP0LNF4i+YzHvpct5MZD1c8JqnzYisOf xq+JQ4tLcsDmrnhJEinBVbiq/epEXs04I4GES+zyEgnz5TPErjTNRDzP 7CE=
example.com.	3600	IN	DNSKEY	257 3 8 AwEAAZ0aqu1rJ6orJynrRfNpPmayJZoAx9Ic2/Rl9VQWLMHyjxxem3VU SoNUIFXERQbj0A9Ogp0zDM9YIccKLRd6LmWiDCt7UJQxVdD+heb5Ec4q lqGmyX9MDabkvX2NvMwsUecbYBq8oXeTT9LRmCUt9KUt/WOi6DKECxoG /bWTykrXyBR8elD+SQY43OAVjlWrVltHxgp4/rhBCvRbmdflunaPIgu2 7eE2U4myDSLT8a4A0rB5uHG4PkOa9dIRs9y00M2mWf4lyPee7vi5few2 dbayHXmieGcaAHrx76NGAABeY393xjlmDNcUkF1gpNWUla4fWZbbaYQz A93mLdrng+M=
example.com.	3600	IN	DNSKEY	256 3 8 AwEAAb1oJO+fCqdkxHtQYVB/tFPgJphc+VxjUYz+eVGf077zMxHKgce9 EwGBifFuKhjl2EA0VQPsWVX1vzuUmWri3OgsTBlITkdMz6VU4g94uO6T 9MIktokouOidIzvOqLR+O2LSXNhiYOIWA9s3Lxk5R2lrwd6vrRvT2CR1 GdZuUlKB
example.com.	2694	IN	RRSIG	DNSKEY 8 2 3600 20221129010414 20221107233521 31406 example.com. nMEQXWFatPZd/fkGgi9TI4ZO2vokX+6zNNmZPSOnweki1Vb25f+oISgH b1WEg84IzyUw+zzwmS2G4J08PvS8+rFfu9vprvPwKVMsgOzBSyt3CCLS qa1DtY20BMWXCzqHD1n16220AUMNGuvrta6ikmuGfXT/gXyK5isenUPn kSbGsbrgEQKPZZQU6H/9nLK2qttyBscCQmJ4ziIbsMyannBWgXtJgXhu 4AhiVAZIxCqII/ISNei3vOcl+h6C+RgjYsnoPD59HkpnC2H7TsaiLNf7 uYtbCjzRKLhRzIwIS3ASbWccGJ3LXruZwUNd0E/XqrxaCZXuwFrq+vtP RYAaPA==

SOLVING DNS INTEGRITY

RRSet #1

```
example.com.    86379   IN   A        93.184.216.34
example.com.    86328   IN   RRSIG    A 8 2 86400 20221120061546 20221030132217 59208 example.com.
                                      rZvjehQxdT5pJ4cw+o1y/BYmLkBLuqzjFaEOn9773Bhywt4qhKmME8DK
                                      oKD4yLjYJYFaqhUNCYb+iimCTdK+9+3UjJ35gRIDC3kuZ9hogtCoLBnt
                                      ltfgFwLO0mdye8iH/FDDVKTm+CAz3UMfcwNzNahvg4BOnZ04HqnpZcWW pu4=
example.com.    73820   IN   NS       a.iana-servers.net.
example.com.    73820   IN   NS       b.iana-servers.net.
example.com.    86237   IN   RRSIG    NS 8 2 86400 20221122065049 20221101032317 59208 example.com.
                                      Uit3UXCeClM+iwVkc2lX8n5A1O0CD9mH8rsTSfcsjQaZD9y54q7bT2mM
                                      cGMaiyjj/sODGKLNvbFKLEgHgP0LNF4i+YzHvpct5MZD1c8JqnzYisOf
                                      xq+JQ4tLcsDmrnhJEinBVbiq/epEXs04I4GES+zyEgnz5TPErjTNRDzP 7CE=
example.com.    3600    IN   DNSKEY   257 3 8 AwEAAZ0aqu1rJ6orJynrRfNpPmayJZoAx9Ic2/RI9VQWLMHyjxxem3VU
                                      SoNUIFXERQbj0A9Ogp0zDM9YIccKLRd6LmWiDCt7UJQxVdD+heb5Ec4q
                                      lqGmyX9MDabkvX2NvMwsUecbYBq8oXeTT9LRmCUt9KUt/WOi6DKECxoG
                                      /bWTykrXyBR8elD+SQY43OAVjlWrVltHxgp4/rhBCvRbmdflunaPIgu2
                                      7eE2U4myDSLT8a4A0rB5uHG4PkOa9dIRs9y00M2mWf4lyPee7vi5few2
                                      dbayHXmieGcaAHrx76NGAABeY393xjlmDNcUkF1gpNWUla4fWZbbaYQz A93mLdrng+M=
example.com.    3600    IN   DNSKEY   256 3 8 AwEAAb1oJO+fCqdkxHtQYVB/tFPgJphc+VxjUYz+eVGf077zMxHKgce9
                                      EwGBifFuKhjl2EA0VQPsWVX1vzuUmWri3OgsTBlITkdMz6VU4g94uO6T
                                      9MIktokouOidIzvOqLR+O2LSXNhiYOIWA9s3Lxk5R2lrwd6vrRvT2CR1 GdZuUlKB
example.com.    2694    IN   RRSIG    DNSKEY 8 2 3600 20221129010414 20221107233521 31406 example.com.
                                      nMEQXWFatPZd/fkGgi9TI4ZO2vokX+6zNNmZPSOnweki1Vb25f+oISgH
                                      b1WEg84IzyUw+zzwmS2G4J08PvS8+rFfu9vprvPwKVMsgOzBSyt3CCLS
                                      qa1DtY20BMWXCzqHD1n16220AUMNGuvrta6ikmuGfXT/gXyK5isenUPn
                                      kSbGsbrgEQKPZZQU6H/9nLK2qttyBscCQmJ4ziIbsMyannBWgXtJgXhu
                                      4AhiVAZIxCqII/ISNei3vOcl+h6C+RgjYsnoPD59HkpnC2H7TsaiLNf7
                                      uYtbCjzRKLhRzIwIS3ASbWccGJ3LXruZwUNd0E/XqrxaCZXuwFrq+vtP RYAaPA==
```

SOLVING DNS INTEGRITY

RRSet #1

```
example.com.   86379   IN   A        93.184.216.34
example.com.   86328   IN   RRSIG    A 8 2 86400 20221120061546 20221030132217 59208 example.com.
                                     rZvjehQxdT5pJ4cw+o1y/BYmLkBLuqzjFaEOn9773Bhywt4qhKmME8DK
                                     oKD4yLjYJYFaqhUNCYb+iimCTdK+9+3UjJ35gRIDC3kuZ9hogtCoLBnt
                                     ltfgFwLO0mdye8iH/FDDVKTm+CAz3UMfcwNzNahvg4BOnZ04HqnpZcWW pu4=
```

RRSet #2

```
example.com.   73820   IN   NS       a.iana-servers.net.
example.com.   73820   IN   NS       b.iana-servers.net.
example.com.   86237   IN   RRSIG    NS 8 2 86400 20221122065049 20221101032317 59208 example.com.
                                     Uit3UXCeClM+iwVkc2lX8n5A1O0CD9mH8rsTSfcsjQaZD9y54q7bT2mM
                                     cGMaiyjj/sODGKLNvbFKLEgHgP0LNF4i+YzHvpct5MZD1c8JqnzYisOf
                                     xq+JQ4tLcsDmmhJEinBVbiq/epEXs04I4GES+zyEgnz5TPEjTNRDzP 7CE=
```

```
example.com.   3600    IN   DNSKEY   257 3 8 AwEAAZ0aqu1rJ6orJynrRfNpPmayJZoAx9Ic2/RI9VQWLMHyjxxem3VU
                                     SoNUIFXERQbj0A9Ogp0zDM9YIccKLRd6LmWiDCt7UJQxVdD+heb5Ec4q
                                     lqGmyX9MDabkvX2NvMwsUecbYBq8oXeTT9LRmCUt9KUt/WOi6DKECxoG
                                     /bWTykrXyBR8elD+SQY43OAVjlWrVltHxgp4/rhBCvRbmdflunaPIgu2
                                     7eE2U4myDSLT8a4A0rB5uHG4PkOa9dIRs9y00M2mWf4lyPee7vi5few2
                                     dbayHXmieGcaAHrx76NGAABeY393xjlmDNcUkF1gpNWUla4fWZbbaYQz A93mLdrng+M=
example.com.   3600    IN   DNSKEY   256 3 8 AwEAAb1oJO+fCqdkxHtQYVB/tFPgJphc+VxjUYz+eVGf077zMxHKgce9
                                     EwGBifFuKhjl2EA0VQPsWVX1vzuUmWri3OgsTBlITkdMz6VU4g94uO6T
                                     9MIktokouOidIzvOqLR+O2LSXNhiYOIWA9s3Lxk5R2lrwd6vrRvT2CR1 GdZuUlKB
example.com.   2694    IN   RRSIG    DNSKEY 8 2 3600 20221129010414 20221107233521 31406 example.com.
                                     nMEQXWFatPZd/fkGgi9TI4ZO2vokX+6zNNmZPSOnweki1Vb25f+oISgH
                                     b1WEg84IzyUw+zzwmS2G4J08PvS8+rFfu9vprvPwKVMsgOzBSyt3CCLS
                                     qa1DtY20BMWXCzqHD1n16220AUMNGuvrta6ikmuGfXT/gXyK5isenUPn
                                     kSbGsbrgEQKPZZQU6H/9nLK2qttyBscCQmJ4ziIbsMyannBWgXtJgXhu
                                     4AhiVAZIxCqII/ISNei3vOcl+h6C+RgjYsnoPD59HkpnC2H7TsaiLNf7
                                     uYtbCjzRKLhRzIwIS3ASbWccGJ3LXruZwUNd0E/XqrxaCZXuwFrq+vtP RYAaPA==
```

SOLVING DNS INTEGRITY

example.com.	86379	IN	A	93.184.216.34	RRSet #1
example.com.	86328	IN	RRSIG	A 8 2 86400 20221120061546 20221030132217 59208 example.com. rZvjehQxdT5pJ4cw+o1y/BYmLkBLuqzjFaEOn9773Bhywt4qhKmME8DK oKD4yLjYJYFaqhUNCYb+iimCTdK+9+3UjJ35gRIDC3kuZ9hogtCoLBnt ltfgFwLO0mdye8iH/FDDVKTm+CAz3UMfcwNzNahvg4BOnZ04HqnpZcWW pu4=	
example.com.	73820	IN	NS	a.iana-servers.net.	RRSet #2
example.com.	73820	IN	NS	b.iana-servers.net.	
example.com.	86237	IN	RRSIG	NS 8 2 86400 20221122065049 20221101032317 59208 example.com. Uit3UXCeClM+iwVkc2lX8n5A1O0CD9mH8rsTSfcsjQaZD9y54q7bT2mM cGMaiyjj/sODGKLNvbFKLEgHgP0LNF4i+YzHvpct5MZD1c8JqnzYisOf xq+JQ4tLcsDmmhJEinBVbiq/epEXs04I4GES+zyEgnz5TPEjTNRDzP 7CE=	
example.com.	3600	IN	DNSKEY	257 3 8 AwEAAZ0aqu1rJ6orJynrRfNpPmayJZoAx9Ic2/Rl9VQWLMHyjxxem3VU SoNUIFXERQbj0A9Ogp0zDM9YIccKLRd6LmWiDCt7UJQxVdD+heb5Ec4q lqGmyX9MDabkvX2NvMwsUecbYBq8oXeTT9LRmCUt9KUt/WOi6DKECxoG /bWTykrXyBR8elD+SQY43OAVjlWrVltHxgp4/rhBCvRbmdflunaPIgu2 7eE2U4myDSLT8a4A0rB5uHG4PkOa9dIRs9y00M2mWf4lyPee7vi5few2 dbayHXmieGcaAHrx76NGAABeY393xjImDNcUkFTgpNWUla4fWZbbaYQz A93mLdrng+M=	
example.com.	3600	IN	DNSKEY	256 3 8 AwEAAb1oJO+fCqdkxHtQYVB/tFPgJphc+VxjUYz+eVGf077zMxHKgce9 EwGBifFuKhjl2EA0VQPsWVX1vzuUmWri3OgsTBlITkdMz6VU4g94uO6T 9MIktokouOidIzvOqLR+O2LSXNhiYOIWA9s3Lxk5R2lrwd6vrRvT2CR1 GdZuUlKB	
example.com.	2694	IN	RRSIG	DNSKEY 8 2 3600 20221129010414 20221107233521 31406 example.com. nMEQXWFatPZd/fkGgi9TI4ZO2vokX+6zNNmZPSOnweki1Vb25f+oISgH b1WEg84IzyUw+zzwmS2G4J08PvS8+rFfu9vprvPwKVMsgOzBSyt3CCLS qa1DtY20BMWXCzqHD1n16220AUMNGuvrta6ikmuGfXT/gXyK5isenUPn kSbGsbrgEQKPZZQU6H/9nLK2qttyBscCQmJ4ziIbsMyannBWgXtJgXhu 4AhiVAZIxCqII/ISNei3vOcl+h6C+RgjYsnoPD59HkpnC2H7TsaiLNf7 uYtbCjzRKLhRzIwIS3ASbWccGJ3LXruZwUNd0E/XqrxaCZXuwFrq+ytP RYAaPA==	

Next Class, Confidentiality!