March 1, 2021
DNS

- Today
  - Paper 2 Info
  - DNS, the good, the bad and the ugly
  - Paper Presentation
- Assignments
  - Project
    - Outline: Due Monday, Mar 29

Paper 2

- Need selection of dates (1st and 2nd choices) by next week
- Choice of paper must be approved by Dr C at least two weeks prior to the presentation
  - Preference is for current (no more than 2-3 years) academic journal or conference article
  - Commercial whitepapers are acceptable unless they're pure marketing with no technical content
  - If possible, paper should support your project choice.
- Choice of paper will be announced to the class at least one week prior to the presentation

Domain Name System

- Transaction Structure
- Message Flow
- Proxies and Caches

DNS

- DNS is distributed
  - Organized as a tree, with the **root nameservers** at the top
  - Each **top-level domain (TLD)** (e.g., .com, .edu, .gov, .uk) served by a separate root nameserver
- Authoritative Name Servers responsible for their domains
- Domain information stored as a **zone record**

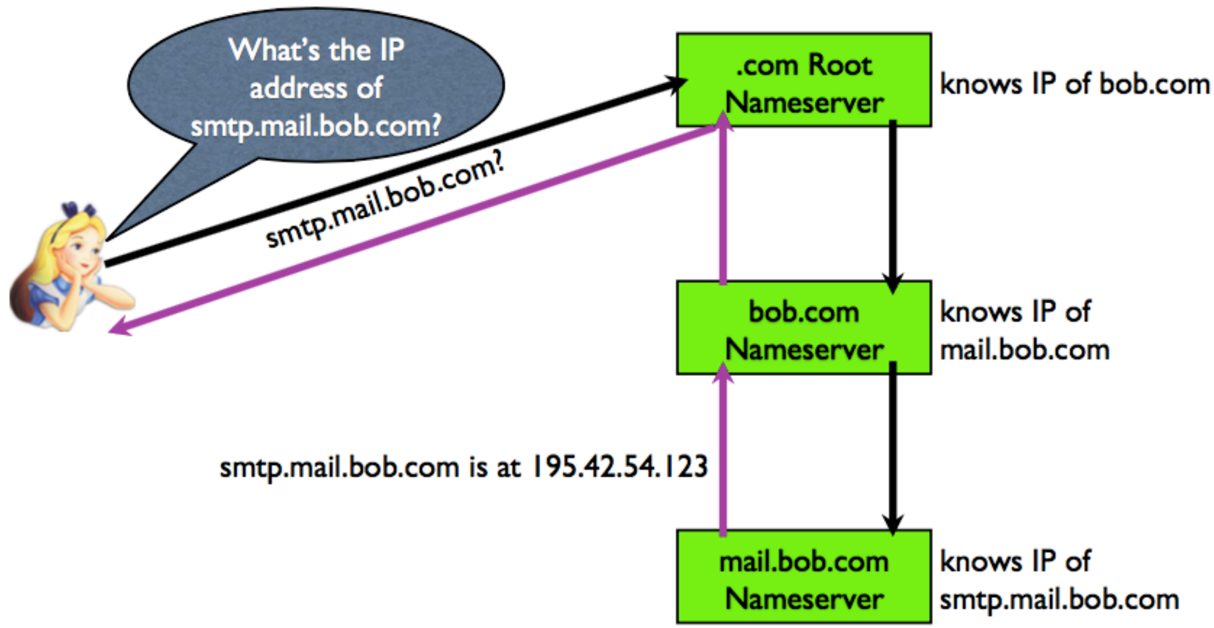
Name Servers

- **AuthoritativeNameServer**: gives authoritative results for hostnames that have been configured
- Domains are registered with a **domain name registrar** (e.g., GoDaddy)
- Each domain must have one primary and at least one secondary name servers
- For reliability in case of failure

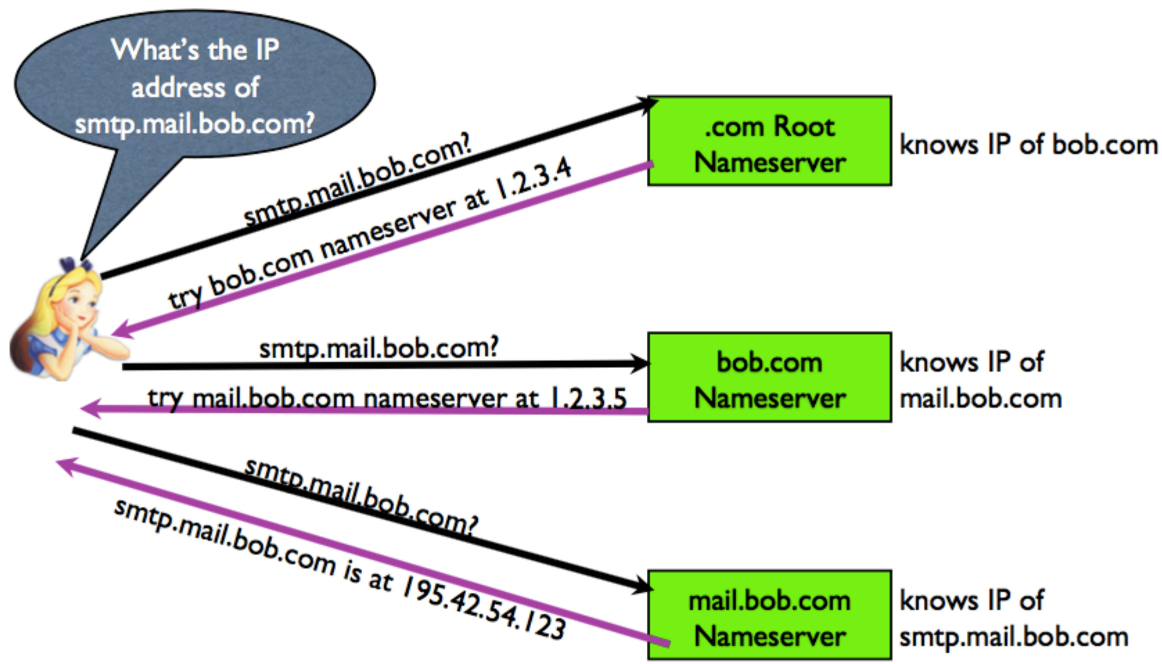
TLDs

- Name servers pre-loaded with IP addresses of TLD name servers

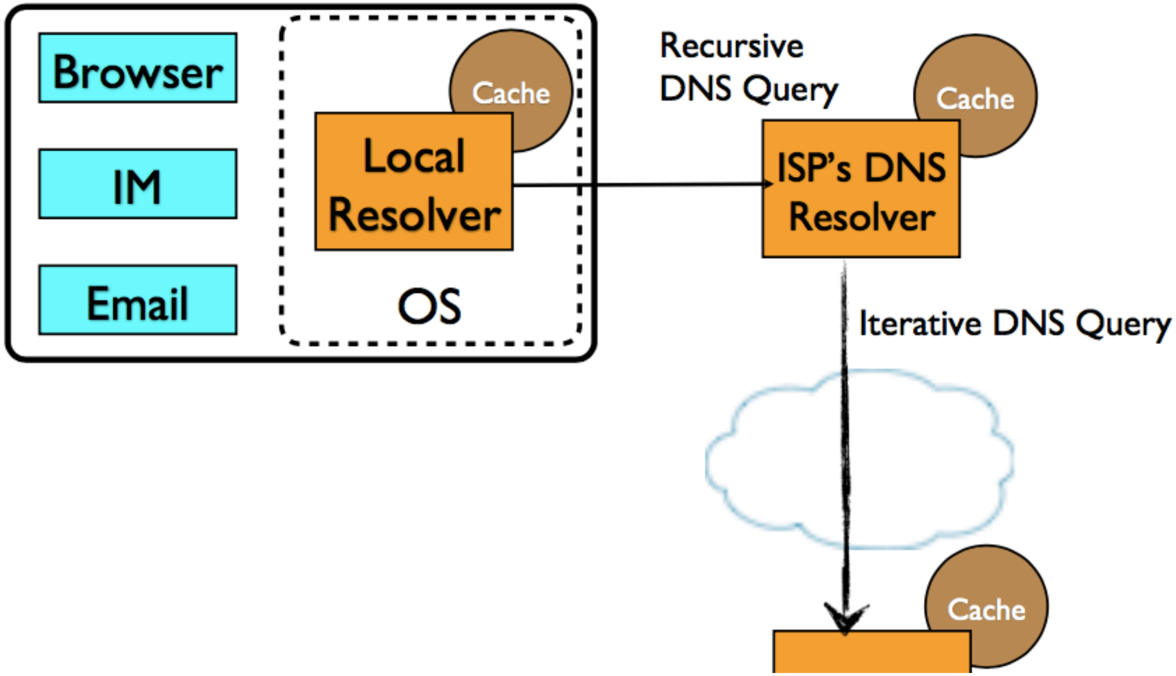
Naïve Recursive Query



Naïve Iterative Query



DNS in the real world



DNS Vulnerabilities

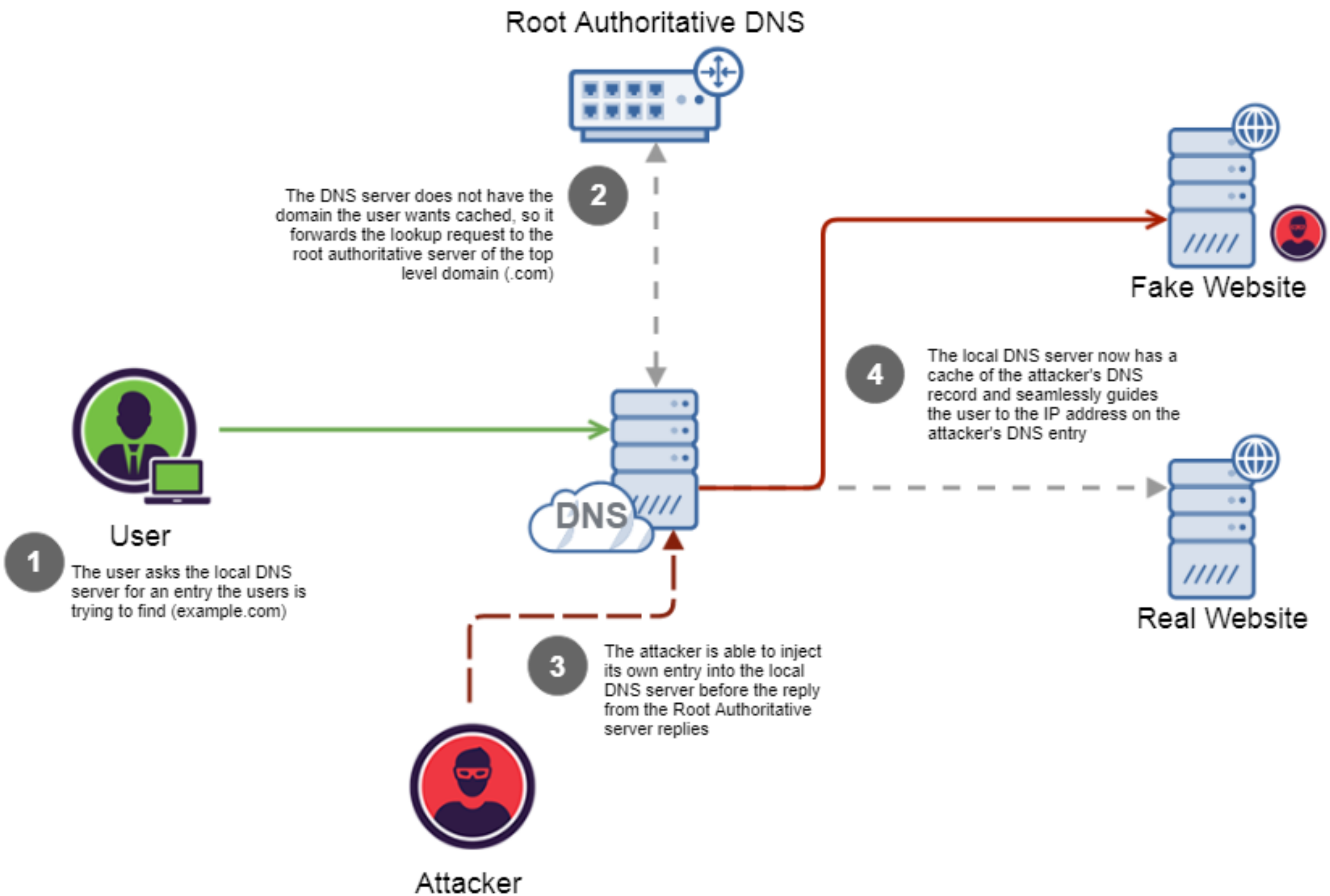
- DNS requests and responses are not authenticated
  - Yet many applications trust DNS resolutions
  - ... or, more accurately, they don't consider the threat at all
  - Spoofing of DNS is very dangerous **-- WHY?**
- Caching doesn't help:
  - DNS relies heavily on caching for efficiency, enabling **cache pollution** attacks
  - Once something is wrong, it can remain that way in caches for a long time
  - Data may be corrupted before it gets to authoritative server

A Cache Poisoning Attack

- All DNS requests have a unique query ID
- The nameserver/resolver uses this information to match up requests and responses -- this is useful since DNS uses UDP
- If an adversary can guess the query ID, then it can forge the responses and pollute the DNS cache
  - 16-bit query IDs (only 2^{16}=65536 possible query IDs)
  - Some servers increment IDs (or use some other predictable algo)
  - gethostbyname returns as soon as it gets a response, so first one in wins!!!
- Note: If you can observe the traffic going to a name server, you can pretty much arbitrarily 0wn the Internet for the clients it serves

A Cache Poisoning Attack

- A simple (and extremely effective) attack:
    1. Wait for Alice to send DNS request to nameserver
    2. Intercept request
    3. Quickly insert a fake response
- If attacker is faster and/or closer to Alice than the DNS server, then the attack is successful
- Advantage attacker: unlike the name server, the attacker doesn't have to do any actual resolving


Root Authoritative DNS
2
The DNS server does not have the domain the user wants cached, so it forwards the lookup request to the root authoritative server of the top level domain (.com)
Fake Website
4
The local DNS server now has a cache of the attacker's DNS record and seamlessly guides the user to the IP address on the attacker's DNS entry
DNS
User
1
The user asks the local DNS server for an entry the users is trying to find (example.com)
Real Website
3
The attacker is able to inject its own entry into the local DNS server before the reply from the Root Authoritative server replies
Attacker

Attack Limitations

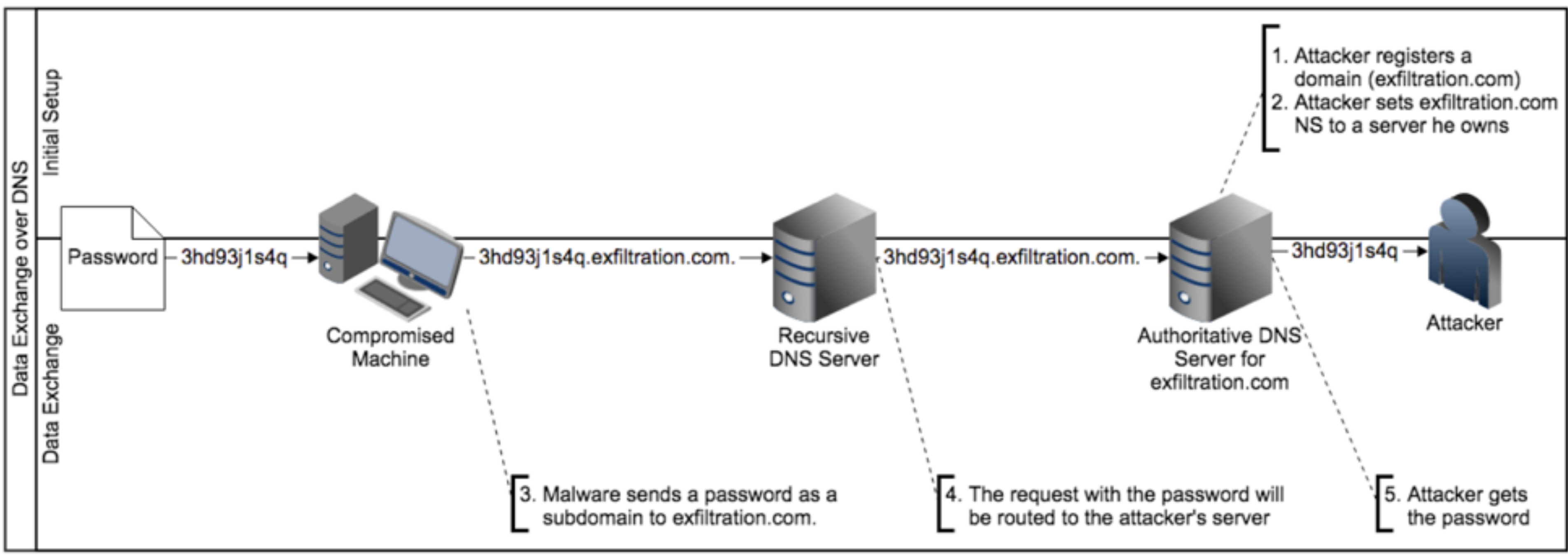
- Victim hostname cannot already be in the cache
- Randomizing the QueryID makes the race condition much harder to exploit
  (2^{16} possible QueryIDs)

Kaminsky Attack

- Hijacks the entire name server of victim host • Basic idea
- Choose a random hostname in the domain (guaranteed not to be cached)
- Try to beat real name server response (guessing the QueryID)
- Forged response specifies an update for the name server IP address (to attacker)
- Repeat until successful
- All future DNS queries for the victim domain now directed to the attacker’s DNS server (until TTL expires)

DNS for Exfiltration

- Assuming the Kaminsky attack has succeeded
  - Or (think about the structure of DNS)
- What can we do with it?



Mitigations?

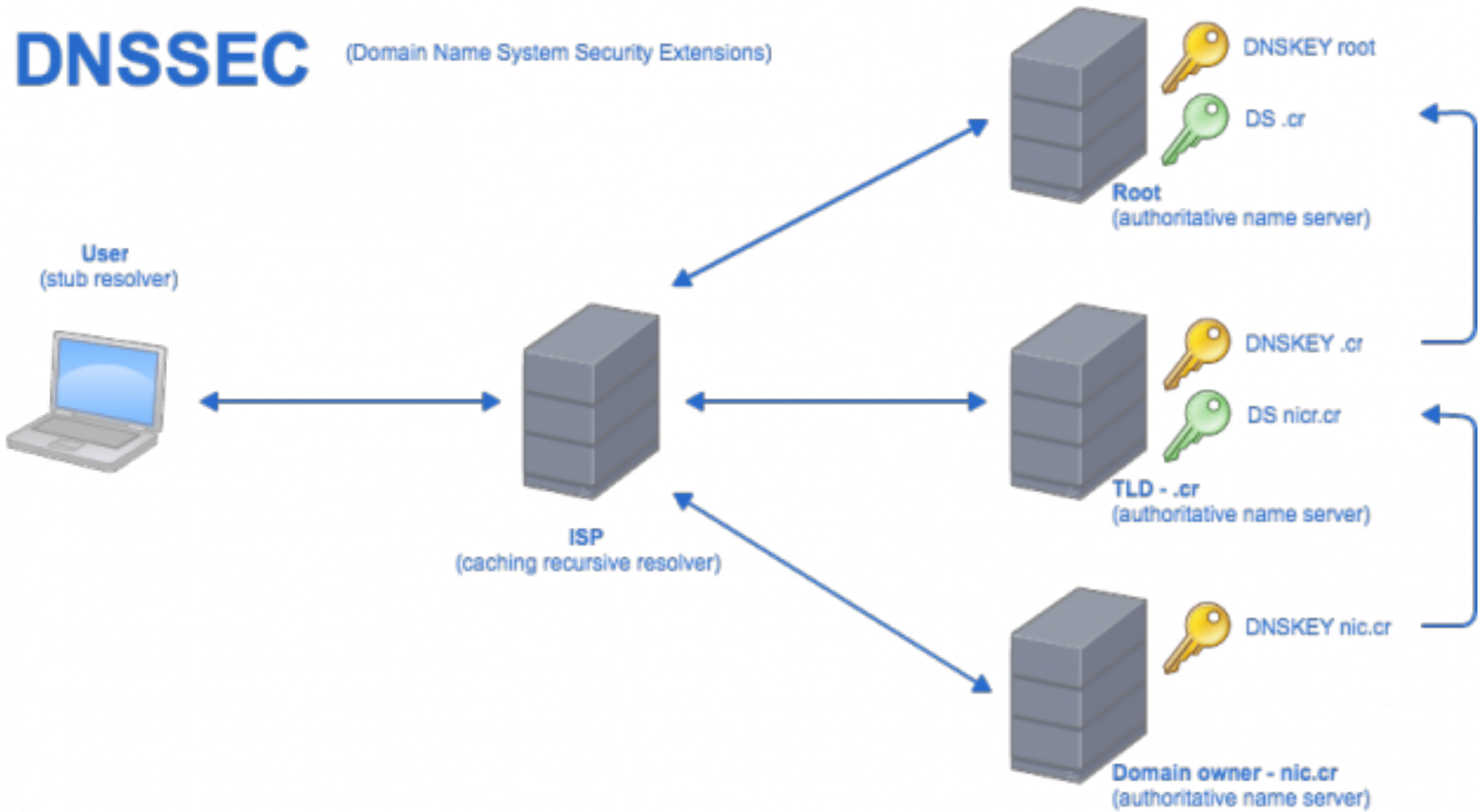
- The QueryID is 16 bits.
- Increasing the size would break the Internet
- What else can we randomize?
  - Source port address

DNS Going Forward

DNSSec

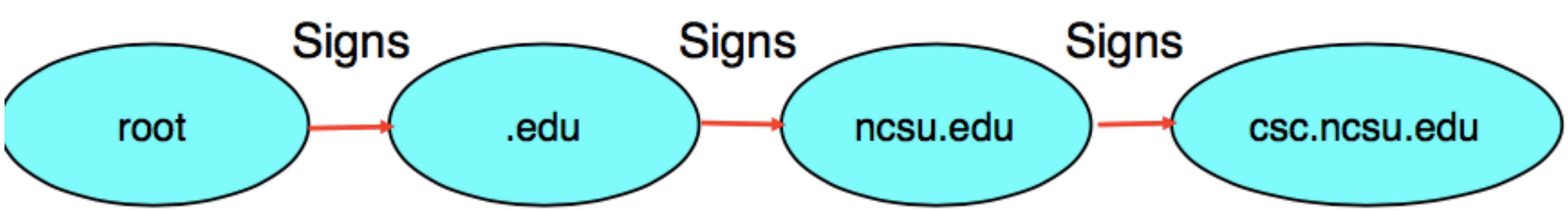
A standards-based (IETF) solution to security in DNS

- Prevents data spoofing and corruption
- Authentication (verifiable DNS) using public key infrastructure
- Authenticates:
  - Communication between servers
  - DNS data
    - Content
    - Existence
    - non-existence
  - Public keys


DNSSEC
(Domain Name System Security Extensions)
DNSKEY root
DS .cr
Root
(authoritative name server)
User
(stub resolver)
DNSKEY .cr
DS nicr.cr
TLD - .cr
(authoritative name server)
ISP
(caching recursive resolver)
DNSKEY nic.cr
Domain owner - nic.cr
(authoritative name server)

DNSSEC Mechanisms

- Each domain signs their “zone” with a private key
- Public keys published via DNS
- Zones signed by parent zones
- Ideally, you only need a self-signed root, and follow keys down the hierarchy



DNSSec Challenges

- Incremental deployability
  - Everyone has DNS, can’t assume a flag day
- Resource imbalances
  - Some devices can’t afford real authentication
- Performance
  - Certificate process
- Cultural
  - Who gets to control the root keys? (US, China, CofC?)
  - Most people don’t have any strong reason to have secure DNS ($$$ not justified in most environments)
  - Lots of transitive trust assumptions
- Take away: DNSSEC will be deployed, but it is unclear whether it will be used appropriately/widely