Certificates

- A ubiquitous form of authentication
- Generally used with public key cryptography
- A signed electronic document proving you are who you claim to be
- Often used to help distribute other keys

Public Key Certificates

- The most common kind of certificate
- Addresses the biggest challenge in widespread use of public keys
  - How do I know whose key it is?
- Essentially, a copy of your public key signed by a trusted authority
- Presentation of the certificate alone serves as authentication of your public key

Implementation of Public Key Certificates

- Set up a universally trusted authority
- Every user presents his public key to the authority
- The authority returns a certificate
  - Containing the user's public key signed by the authority's private key
- In essence, a special type of key server

Checking a Certificate

- Every user keeps a copy of the authority's public key
- When a new user wants to talk to you, he gives you his certificate
- Decrypt the certificate using the authority's public key
- You now have an authenticated public key for the new user
- Authority need not be checked on-line

Scaling Issues of Certificates

- If there are 1-2 billion Internet users needing certificates, can one authority serve them all?
- Probably not
- So you need multiple authorities
- Does that mean everyone needs to store the public keys of all authorities?

Certification Hierarchies

- Arrange certification authorities hierarchically
- Single authority at the top produces certificates for the next layer down
- And so on, recursively

Using Certificates From Hierarchies

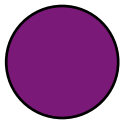
- I get a new certificate
- I don't know the signing authority
- But the certificate also contains that authority's certificate
- Perhaps I know the authority who signed this authority's certificate

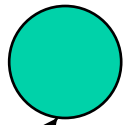
Extracting the Authentication

- Using the public key of the higher level authority,
  - Extract the public key of the signing authority from the certificate
- Now I know his public key, and it's authenticated
- I can now extract the user's key and authenticate it

A Example

Alice gets a message with a certificate

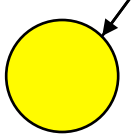




Then she uses
to check 

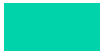
Should Alice believe that he's really  ?

So she uses 
to check



Give me a certificate saying that I'm 

Alice has never heard of
But she has heard of 

How can  prove who he is?

Certification Hierarchies Reality

- Not really what's used
  - For the most part
- Instead, we rely on large numbers of independent certifying authorities
  - Exception is that each of them may have internal hierarchy
- Essentially, a big list
- Is this really better?

Certificates and Trust

- Ultimately, the point of a certificate is to determine if something is trusted
  - Do I trust the request enough to perform some financial transaction?
- So, Trustysign.com signed this certificate
- How much confidence should I have in the certificate?

Potential Problems in the Certification Process

- What measures did Trustysign.com use before issuing the certificate?
- Is the certificate itself still valid?
- Is Trustysign.com's signature/ certificate still valid?
- Who is trustworthy enough to be at the top of the hierarchy?

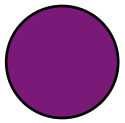
Trustworthiness of Certificate Authority

- How did Trustysign.com issue the certificate?
- Did it get an in-person sworn affidavit from the certificate's owner?
- Did it phone up the owner to verify it was him?
- Did it just accept the word of the requestor that he was who he claimed to be?
- Has authority been compromised?

What Does a Certificate Really Tell Me?

- That the certificate authority (CA) tied a public/private key pair to identification information
- Generally doesn't tell me why the CA thought the binding was proper
- I may have different standards than that CA

Showing a Problem Using the Example

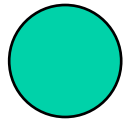


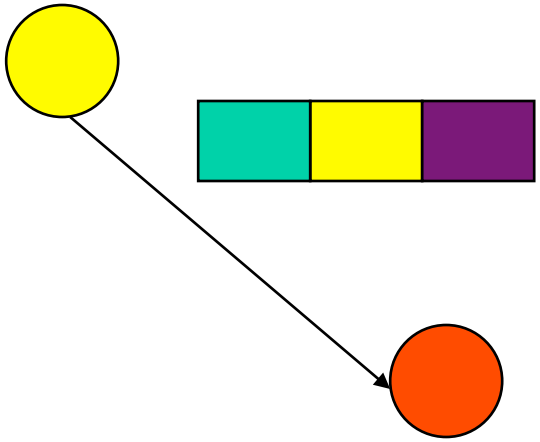
Alice likes how verifies identity

But is she equally happy with how verifies identity?

Does she even know how verifies identity?

What if uses 's lax policies to pretend to be ?





Another Big Problem

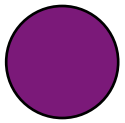
- Things change
  - E.g., 2012 compromise of Adobe private keys
- One result of change is that what used to be safe or trusted isn't any more
- If there is trust-related information out in the network, what will happen when things change?

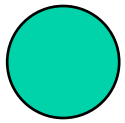
Revocation

- A general problem for keys, certificates, access control lists, etc.
- How does the system revoke something related to trust?
- In a network environment
- Safely, efficiently, etc.
- Related to revocation problem for capabilities

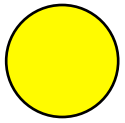
Revisiting Our Example

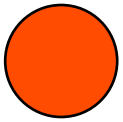
Someone discovers that  has obtained a false certificate for 





How does Alice make sure that she's not accepting  's false certificate?





Realities of Certificates

- Most OSes come with set of "pre-trusted" certificate authorities
- System automatically processes (i.e., trusts) certificates they sign
- Usually no hierarchy
- If not signed by one of these, present it to the user
  - Who always accepts it . . .

An Example

- Firefox web browser
- Makes extensive use of certificates to validate entities
  - As do all web browsers
- Comes preconfigured with several certificate authorities
  - Over 100 of them

Firefox Preconfigured Certificate Authorities

- Some you'd expect:
  - Microsoft, RSA Security, Verisign, etc.
- Some you've probably never heard of:
    - Unizeto Sp. z.o.o., Netlock Kft.,Chungwa Telecom Co. Ltd.

The Upshot

- If Netlock Kft. says someone's OK, I trust them
  - I've never heard of Netlock Kft.
  - I have no reason to trust Netlock Kft.
  - But my system's security depends on them

The Problem in the Real World

- In 2011, a Dutch authority (DigiNotar) was compromised
- Attackers generated lots of bogus certificates signed by DigiNotar
  - "Properly" signed by that authority
  - For popular web sites
- Until compromise discovered, everyone trusted them

Effects of DigiNotar Compromise

- Attackers could transparently redirect users to fake sites
  - What looked like Twitter was actually attacker's copycat site
- Allowed attackers to eavesdrop without any hint to users
- Apparently used by authorities in Iran to eavesdrop on dissidents

How Did the Compromise Occur?

- DigiNotar had crappy security
  - Out-of date antivirus software
  - Poor software patching
  - Weak passwords
  - No auditing of logs
  - Poorly designed local network
- A company providing security services paid little attention to security

But how were you supposed to know that?

Another Practicality

- Certificates have expiration dates
  - Important for security
  - Otherwise, long-gone entities would still be trusted
- But perfectly good certificates also expire
  - Then what?

The Reality of Expired Certificates

- When I hear my server's certificate has expired, what do I do?
  - I trust it anyway
  - After all, it's my server
- But pretty much everyone does that
  - For pretty much every certificate
- Not so secure

The Core Problem With Certificates

- Anyone can create some certificate
- Typical users have no good basis for determining whose certificates to trust
  - They don't even really understand what they mean
- Therefore, they trust almost any certificate

Should We Worry About Certificate Validity?

- Starting to be a problem
  - Stuxnet is one example
  - Compromise of DigiNotar and Adobe also
  - Increasing incidence of improper issuance, like Verisign handing out Microsoft certificates
- Not the way most attackers break in today
- With all their problems, still not the weakest link
  - But now being exploited, mostly by the most sophisticated adversaries

Certificate Pinning

- A new(ish) technique for improving certificate security
- A browser technology
- Accept certificates for particular sites only from specified authorities
- In Firefox and Chrome since 2014
- Only a few sites pinned
  - Firefox itself, Google, Twitter, Dropbox
- Doesn't scale well