50.020 Security
Lecture 16 - Digital Signatures

This lecture

- This lecture:
  - Digital Signature schemes
  - Certificates

RSA for Digital Signature Signing and Verification

Simple RSA Signatures

Public key: e, n
Private key: d
wants to send
authenticated m

e, n

$s = m^d \bmod n$

m, s

check if
$m = s^e \bmod n$

Possible attacks on RSA signature: Attack I

- (Textbook) RSA signature signing is malleable
- In particular, an attacker can generate a valid signature for m'=ab if he is given the the signature of a, and the signature of b:
  - for m'=ab, its signature s'=$(ab)^d$ mod n=$(a^d$ mod n) * (b^d mod n)=(signature of a) * (signature of b)

Attacks on RSA signatures: Attack II

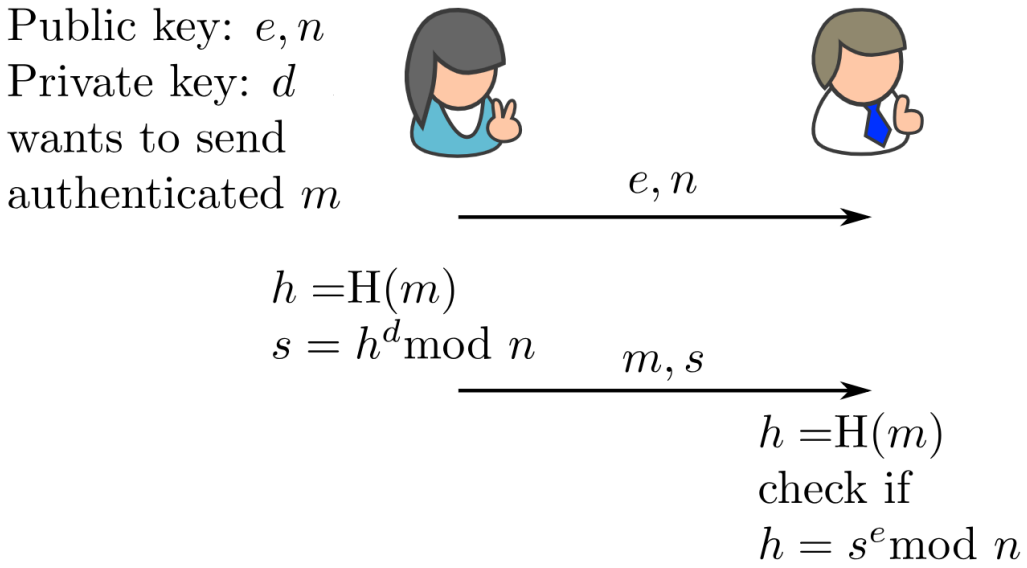
- There is another attack on the presented simple scheme
- Generate a valid signature s if m can be chosen by attacker
  - m will most likely look random to receiver
- Do you have ideas to achieve this?

Attacks on RSA signatures: Attack II

- There is another attack on the presented simple scheme
- Generate a valid signature s if m can be chosen by attacker
  - m will most likely look random to receiver
- Do you have ideas to achieve this?

- Choose random s. Then $m = s^e \bmod n$
- This attack can be prevented with signing a hash of the message, instead of the message directly

RSA signature protocol (against the two attacks)



- Signing a hash H(m) instead of m defeats both attacks
- Alice computes $s=H(m)^d$ mod n with her private key
- Bob can verify the signature using Alice's public key
- If s^e mod n=H(m), message integrity is intact
- Because H(ab)≠ H(a)*H(b), Attack I is prevented
- Because it is hard to find a m for a given H(m), Attack II is prevented

Mixing RSA Operation Modes (Use RSA for Both Encryption and Digital Signature Signing): Confidentiality and Integrity

Confidentiality and integrity

- Lets assume the following setting:
  - Alice and Bob want to exchange a few short messages
  - Alice and Bob both have public/private key pairs: $k_{e,A}, k_{d,A}, n_A, k_{e,B}, k_{d,B}, n_B$
- How can they establish bi-directional authenticated and confidential channel?

Confidentiality and integrity

- Lets assume the following setting:
  - Alice and Bob want to exchange a few short messages
  - Alice and Bob both have public/private key pairs: $k_{e,A}, k_{d,A}, n_A, k_{e,B}, k_{d,B}, n_B$
- How can they establish bi-directional authenticated and confidential channel?

- Alice encrypts messages with public key of Bob
- Alice then signs the encrypted message with her private key
- Bob does the same with changed keys
- Now they communicate confidential and authenticated

Mixing RSA operation modes

- We discussed attacks on RSA encryption and signing
- If same key pair is used for both, another attack is possible
- Setting: Bob is a server with e,d,n
- Bob offers two services:
  - Submission of encrypted exams by user
  - RSA signatures of certificates by the user
- Alice sends Bob a ciphertext $c = m^e \bmod n$
- How can the attacker learn m?

Mixing RSA operation modes

- We discussed attacks on RSA encryption and signing
- If same key pair is used for both, another attack is possible
- Setting: Bob is a server with e,d,n
- Bob offers two services:
  - Submission of encrypted exams by user
  - RSA signatures of certificates by the user
- Alice sends Bob a ciphertext $c=m^e \bmod n$
- How can the attacker learn m?

- Attacker sends c to Bob, and asks him to sign it
  - Result will be $m^{ed} \bmod n=m$
- You should never use same key pair for both enc and sign

Other Digital Signature Algorithms

Other Digital Signature Algorithms: DSA/ECDSA

- Digital Signature Algorithm (DSA) is the US Federal Standard for digital signatures
- The "message" we saw earlier in Elgamal is now the hash of the message to be signed
- There is also an Elliptic-Curve variant (ECDSA) (faster)

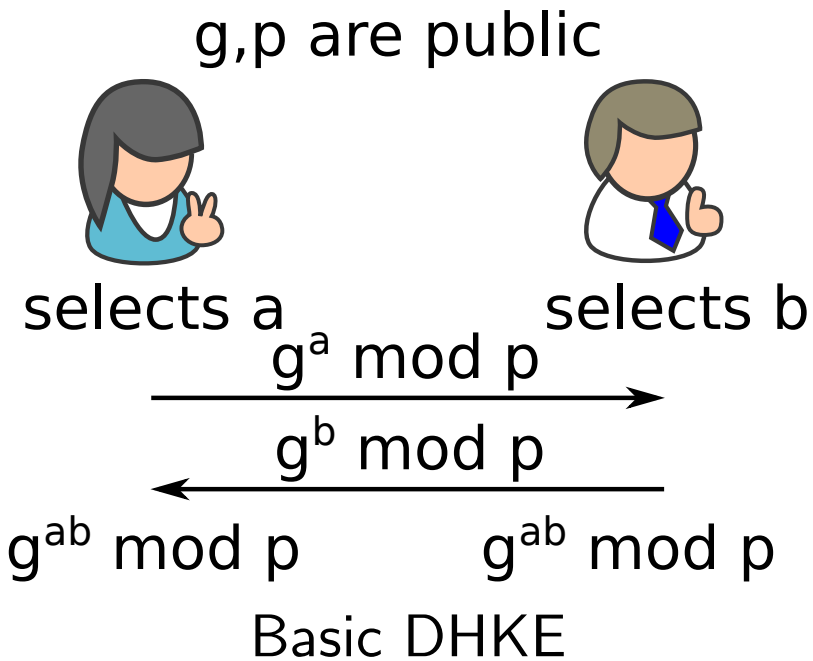
Certificates

Problem: Man-In-the-Middle

- So far, passive attacker for key establishment
  - In DHKE: attacker cannot obtain key
  - In RSA: attacker cannot decrypt
- What if the attacker can stop messages, and replace with own?
  - Such attacks are called *Man-In-the-Middle* (MitM)

MitM Example

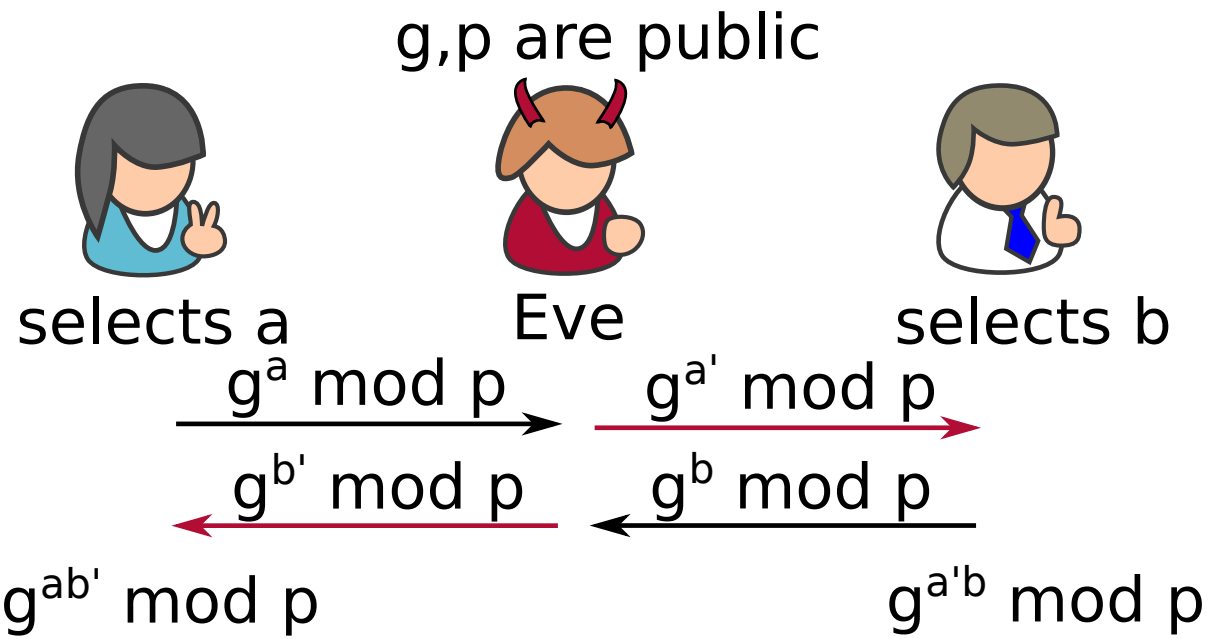
- Consider DHKE with active MitM attacker



Basic DHKE

MitM Example

- Consider DHKE with active MitM attacker



A MitM attacker can actively compromise the session keys

Discussion of MitM

- Requires attacker to intercept and drop messages
- Can be possible on local network (ARP spoofing)
- Or: strong attacker (ISP)
- How to protect against this?
  - Chicken-and-Egg problem!
- Certificates solve this problem

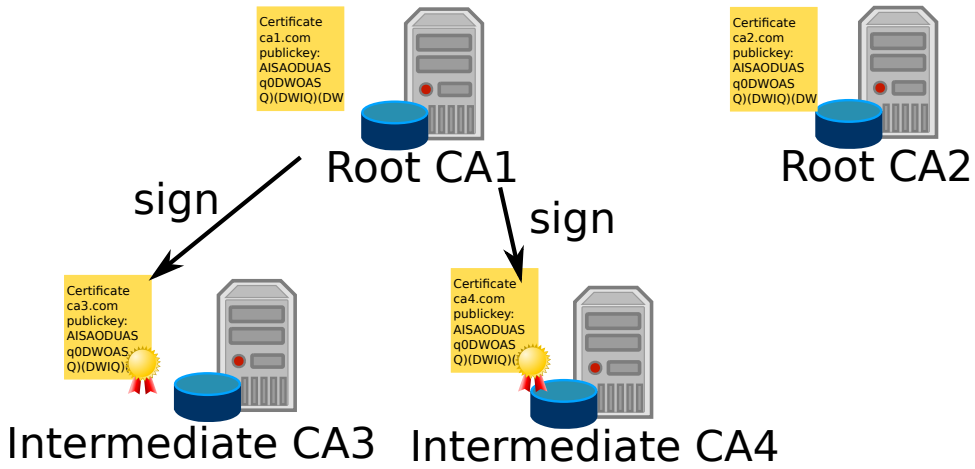
Motivation Certificates

- So far, we just discussed e,n as public keys
- But they need to be tied to *identities*
- Alice wants to start communicating with Bob
  - She gets some e,n that supposedly are from Bob
  - How do I know that there is no Man-in-the-Middle?
- We still did not solve our key establishment problem
- Certificates solve this problem

Certificates

- A basic certificate consists of three things:
  - The public key e of a user
  - The unique ID of the users
  - A signature s over (e,ID) by a *certificate authority* (CA/Charly)
- The user has the public key of Charly
  - We have to bootstrap somewhere (to generate public keys)(to discuss later, PKI or Web of Trust)
- Now, the user can verify whether Charly thinks e belongs to Bob
  - MitM attack is prevented!
- But Charly has to sign all signatures...

Public Key Infrastructure (PKI)

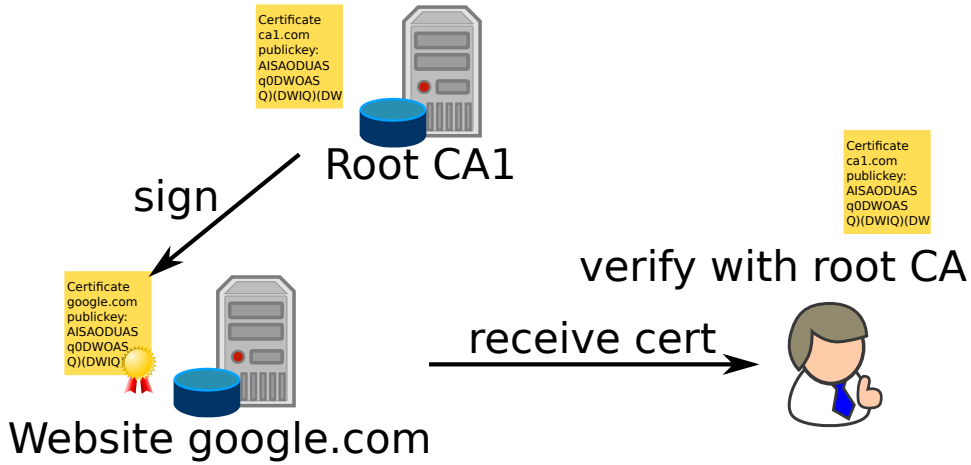


- A public key infrastructure contains a set of CAs, certificates, and mechanisms to verify the certificates
- *Root CAs* will have their certificate distributed to all users
  - E.g. as part of OS and/or browser installation
- These root CAs then sign certificates for *intermediate CAs*
- The intermediate CAs sign certificates for end users
- This established a *chain of trust* between root CA and end user
- PKIs also commonly support *key revocation*

Key revocation and expiring

- Problem to solve: how to mitigate compromised keys?
- Two approaches: key revocation lists/checks and expiring
- Expiring: Certificates usually have limited time validity
  - E.g., only valid between Jan 1 2014 and Jan 1 2015
  - Expired certificates cannot be used any more
  - This ensures some freshness and periodic renewal
- Key revocation lists:
  - A (signed) list of certificates that should not be trusted
  - Has to be updated frequently
  - Has to be checked every time a certificate is validated
- Online revocation check: OCSP is implemented in most browsers
  - Similar to revocation list, remote lookup on demand

Where certificates are used



- Nowadays, certificates are used to secure http traffic
- Any website you visit that uses https will authenticate using a certificate
  - Protocol that is used is SSL/TLS (see next lecture)
- This enables authenticated and confidential communication with server
- TLS also allows the clients to authenticate, so bi-directional authentication is possible

Letsencrypt

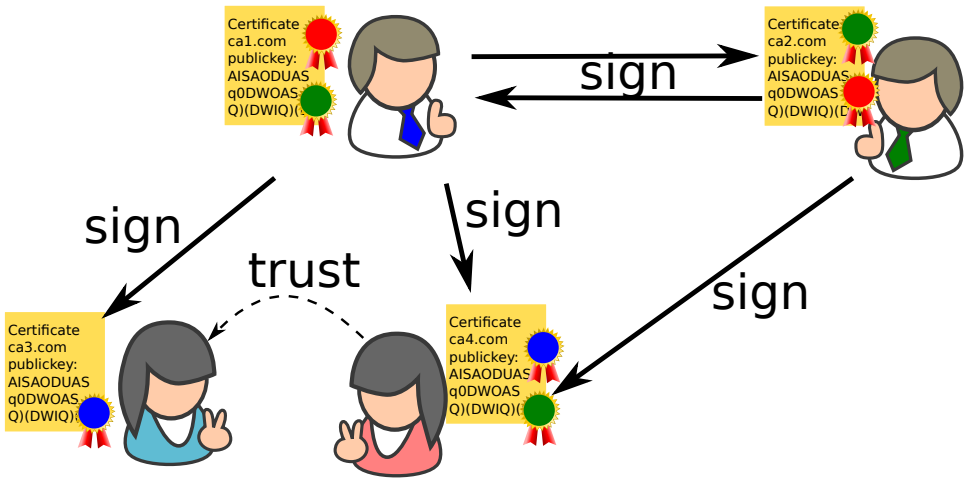
- If you run your own webserver, you might want a certificate
- But buying these is expensive!
  - Godaddy.com: certificate up to 400 USD/year
- Letsencrypt is an initiative to provide free certificates
  - Provide an automated tool to obtain and maintain certificates
  - `https://letsencrypt.org/`

PKI vs. Web of Trust

Problems with PKI

- Nowadays, PKI are huge, too many entities have to be trusted
- 1482 CAs trusted in Mozilla on Windows (EFF in 2011)
- 651 organizations need to be trusted
- If one of these certificates is compromised, MitM is possible
- Remember the "MD5 collisions, inc" intermediate CA certificate?
- VeriSign was hacken in 2010 (repeatedly)
- In 2014, Indian CA gave out certificates on Google.com

Web of Trust



- Idea: decentralize the PKI, implemented by PGP (Pretty Good Privacy) and GPG (GNU privacy Guard)
- Certificates get signed by other users
- If I trust one or more of the signers, I trust the certificate
- To sign a certificate, both users meet physically
  - This prevents network-based attacks
- Each user will have a *trust score* (assuming transferable trust)
- Getting a signature by a trusted user increases your trust

Key signing parties

- The Web of Trust idea is hard to bootstrap
  - There are few people with PGP keys
  - Even less of them have sufficiently signed keys
- The original idea requires proof of ID and validation of checksums
  - This requires physical contact between key owner and signer
- *Key signing parties* are intended to help with the initial signing
  - More info on `https://wiki.debian.org/Keysigning`

Web of Trust in practise

- Some users feedback that the web of trust idea does not work well
- Most of the keys are directly accepted by the user him/herself
- In practise, the problem of trust in the keys is delegated to the user himself/herself
- Not widely adopted, due t some other issues in practice:
  - If you ever lose your private key, you will not be able to read old documents...
  - Your key cannot be deleted. When you submit a key to a key server the key is also forwarded to other key servers around the world, and they in turn forward the key to still other servers. Deleting the key from one server would not cause it to be deleted from any of the other servers in the world.

Web of Trust Application: PGP

- PGP (Pretty Good Privacy) was a tool first published in 1991
  - It allowed digital signature generation, signature validation, encryption, and decryption
  - At that time, export of crypto algorithms from US was regulated, the author had to *print the source as a book* to legally distribute the source code outside US
  - PGP quickly became popular among security enthusiasts
- Instead of using a PKI, it uses the *Web of Trust* idea
- Core ideas of PGP have been translated into the OpenPGP standard
- GPG (GNU privacy Guard) is essentially a compatible open source tool

Example use of PGP

- Let's assume you want to send a SUTD friend a confidential mail
  - Even SUTD IT should not be able to read it
- You need your friend's public key
- You need a mail client that supports PGP, e.g. Thunderbird with enigmail add-on
- Import your friend's key into your enigmail keychain
- Then make sure to select "encrypt" when sending the mail to your friend
  - If you have your own key, you can also sign your mail

Example use of PGP II

- When your friend receives your mail, his/her mailclient will decrypt it using his/her private key
- If your friend has your public key, he/she can also validate your signature
- The the confidentiality and authenticity of the content was validated

Summary of public key crypto so far

- We discussed a key establishment protocol
  - Diffie-Hellman, secure against passive attacker
  - Use asymmetric keys to exchange symmetric keys
- We discussed public key encryption
  - Encryption always uses the public key
  - RSA needs OAEP or similar against manipulations
  - Elgamal can be used as encryption function
- We discussed public key signatures
  - RSA signatures using the private key
  - Use a secure hash to ensure integrity
  - Other digital signature algorithms such as DSA/ ECDSA
- We discussed certificates for authenticating a user who is the real user holding the public key
  - PKI is centralized. Trust issue if the centralized CA is compromised.
  - Web of Trust is decentralized, but not working well in practice.