Cryptography

Esther’s added slides (the rest are in the lecture slide deck)

RSA- Rivest Shamir Adleman

Uses modular arithmetic as its secret sauce.

- Generate large primes **p** and **q**.
- Calculate **n = p*q**. n is the “modulus” (public)
- Calculate the totient **phi = (p-1)(q-1)**, or lcm(p-1,q-1) in the new standard
- Choose integer **e** between 1 and phi s.t. e and phi are coprime.
  - e is the “public key exponent”
- Compute **d** such that **d*e = 1 mod phi**
  - d = (1 + k*phi)/e
  - d is private
- **Publish: (n, e) on key servers somewhere**
- **Keep private: (n, d)**

(“Side channel attack” still possible where someone steals your private key on your comp)

RSA- Rivest Shamir Adleman

- You can encrypt a message m by raising to the e power and taking the mod n to get c.
  - **$c = m^e \mod n$**
- Decrypt it to get m back by raising c to the d power and taking the mod n.
  - **$m = c^d \mod n$**
- Chinese remainder theorem: **$m^{ed} \equiv m \mod n$**. Since c is m^e mod n, c^d mod n is the desired m.

Why does this work?

- You can't compute d, p, or q from knowing n and e.
  - Prime factorization of large integers is hard, and if you pick one with a large number of digits (>=2048 bits) it's very secure.
  - The "RSA problem": to take e^{th} root of c, mod n. The RSA algo defines a **one way function**.

Digital Signatures

In summary:

- Allow you to verify that a file has not been tampered with (integrity) and it’s the right person who sent it (authenticity)
- Compute a hash of the file, encrypt it, and attach it to the end of the file as a signature.
- When the person receives the file, they hash it, decrypt the signature, and compare the hash with the decrypted signature.

Checking a hash

Checking debian checksum and signatures
https://www.debian.org/CD/verify
https://cdimage.debian.org/debian-cd/9.6.0-live/amd64/iso-hybrid/
https://linuxconfig.org/how-to-verify-an-authenticity-of-downloaded-debian-iso-images

Checking ubuntu-mate distro checksum http://ubuntu-mate.org/download/