ECE 435 – Network Engineering
Lecture 9

Vince Weaver

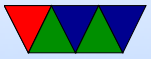
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

11 February 2026

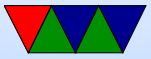
Announcements

- HW#3 was posted. Encryption. No coding.
- HW#2 was due



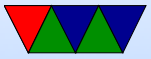
SHA-1

- Developed by NSA 1993
- 160-bits (40 hex digits)
- Deprecated by NIST since 2011
- SHAppening (2015)
- SHAttered (2017) first collision (pdf file)
- chosen-prefix attack 2019
- Used by git (oops)



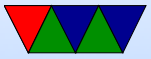
SHA-2, SHA-3

- SHA-2 (Secure-Hash Algorithm 2)
  - Designed by NSA, 2001
  - Family of 6 possible bit sizes: SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256
  - Merkle-Damgård construction
- SHA-3
  - Keccak, Sponge Construction
  - Different than others. Not meant to replace SHA-2 as SHA-2 not broken yet



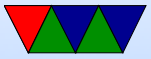
Cryptographic Hash Uses

- passwords (/etc/shadow) (explain this a bit)
- (mostly) uniquely identifying a file (git),
- verifying file contents (download, error checking),
- bitcoin?



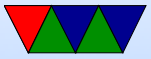
Proof of Concept || GTFO

- One issue of hacker magazine had fun generating collisions
- Distributed as PDF that included its own md5sum (should that be hard?)
- Same PDF file was also a zip file and an NES ROM you could run in an emulator, also showing the sum



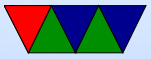
Encrypted Messages Not Always Enough

- Redundancy, some way to validate plaintext is valid. Example: if encrypting a binary blob where each byte indicates something (12 34 means order 34 cows or something), random garbage might decode to valid message
- Freshness – replay attacks. What if you record old message (Bank deposits $100 to account) and replay. Will have valid encryption.



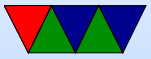
Encryption Problems

- Secret Keys leaked
  - Issue on hardware that has to decrypt but can't let user find the keys (DVD players, game consoles)
  - How to revoke keys that have been leaked? Firmware update? Over internet, on DVD?
- Poor random number generation
  - Famous Debian mistake in patching library, leading to most systems getting one of a few possible ssh keys
  - Move mouse when generating ssh keys?



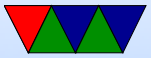
Breaking Encryption / Cryptanalysis

- A lot of work goes into breaking encryption, either to spy, or to ensure algorithms are strong
- differential cryptanalysis (start with similar plaintexts and see what patterns occur in output)
- Power/Timing analysis – note power usage or timing/cache/cycles when encryption going on, can leak info on key or algorithm
  Bane of perf. Cycle invariant code?
- Quantum computers?



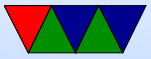
Trusting Trust

- When setting up an encrypted connection, how do you verify who is on the other side?
- How can you protect from man-in-the-middle attacks (MitM) where someone intercepts them downloading your public key, replaces with their own, then sits in the middle decrypting/re-encrypting in a transparent way?
- Some companies/countries will actually do this quite openly



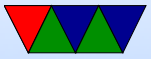
Key Signing Parties

- One way is to have get-togethers where friends sign each others keys
- If enough people do this, you can create a "chain of trust" where you can track someone's identity to someone you trust
- Linux kernel sorta tries this for git development
- Trouble for new people, or remote people, or people who don't travel much, or don't have many friends



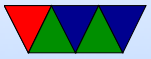
Self-signing Keys

- Can you just sign your own keys?
- You can, but how would someone know it's really you?
- (Though you could argue, does that always matter?)
- Most web browsers get very upset and will show lots of warnings if you use a self-signed key



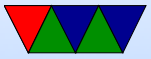
Certificate Authorities

- Certificate authority – an official organization that verifies identities
- Will sign a "certificate" saying who you say you are
- Operating Systems/Web-browsers will ship with a list of officially trusted Certificate Authorities
- Can hover over / click the lock symbol in URL bar to verify who signed for a website



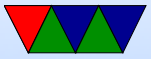
Certificate Authorities Trust

- Can you stop trusting a certificate authority?
- What if they have a habit of signing bad keys, accidentally or on purpose (for example: signing a key saying someone is google.com that isn't)
- CAs can be removed from trusted list
- Often done gradually as not to break companies / programs depending on existing certs



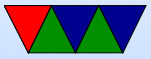
SSL/TLS

- Secure Socket Layer / Transport Layer Security (why the name change? Microsoft/Netscape rivalry back in the day)
- Handshake protocol followed by key exchange
- Browser says hello, which hashes/algorithms it supports
- Server picks one and sends back
- Server then sends a certificate (signed by authority) saying who it is, and what its public key is
- Client verifies certificate (via the CA public key it has



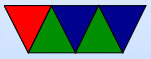
stored [comes with OS or browser]

- client generates a random number, encrypts with servers public key, sends to server, used as symmetric key
- What could go wrong, what if someone gets a hold of server private key? could decrypt logged data.
- Could try Diffie-Hellman key exchange – random number plus unique session key prevents problems if server private key leaked

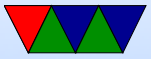


SSL/TLS other

- Long history of TLS, security issues, fixes, etc
- Timeline here: `https://www.feistyduck.com/ssl-tls-and-pki-history/`
- md5sum weakness actually used to trick certificate authorities at one point
- Public Key Pinning, used to find when fraudulent certificates signed by CAs
- Backwards compatibility fallback removed because could be forced to fall back to broken algorithm

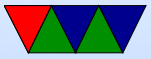


- quantum-resistant encryption methods released
- google removed the lock icon



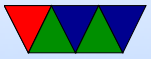
Diffie-Hellman (used by ssh)

- Both sides agree on large prime number
- Both sides agree on algorithm (AES?)
- Each side picks independently picks another secret prime number.
  This is not the authentication private key.
- The secret prime, AES, and shared prime are used to make a public key derived from the private key.
- The generated public key is shared
- The other side uses their own private key, the other side



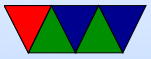
public key, and shared prime to figure out the shared secret key.

- This secret key is then used for symmetric encryption.
- Example on p812



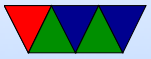
Encryption – Encrypting Hard Drives

- LUKS on Linux
- Bitlocker?
- This gets into the whole signed/trusted firmware on modern machines
- Ransomware



E-mail, Pretty Good Privacy (PGP)

- OpenPGP RFC 4880
- Encrypt message with symmetric key, send along the key encrypted via asymmetric
- GPG – free software replacement for PGP
- Can also PGP sign a message. Not encrypted, but signed with your key to verify it was in fact sent by you. Takes hash of the input, then encrypts the hash with key. Also, downloads from servers (like debian)
- Key stores and key servers?



Encryption – Ethics

- Should everyone be allowed to encrypt things?
- Should governments be allowed backdoors to decrypt things?
- Strong encryption was illegal for a while
  - more than 40 bit encryption an exportable munition
  - Separate browser downloads for US vs foreign
  - Could still export via paper and re-type in
  - people got "illegal" RSA algorithm in perl t-shirts and tattoos