ECE 435 – Network Engineering Lecture 8

Vince Weaver

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

vincent.weaver@maine.edu

9 February 2026

Announcements

- HW#1 will be graded
- HW#2 due today
- HW#3 will be posted. Encryption. No coding.

Symmetric Key Algorithms

- Use same key for encryption and decryption
- Block ciphers, take block of data and encrypt it to same size block (why in blocks?)
- P-box (permutation), S-box (substitution)
- shift/permute/xor
- *very* important that the key is picked randomly.

Symmetric Key Implementations – Historical (DES)

- DES – Data Encryption Standard
- From 1976
- 64 bit key (56-bits used)
- NSA had say on key size.
- 19 stages based on Key
- widely used until broken.
- Competition to break various sizes.

Symmetric Key Implementations – Historical (3DES)

- 3DES (running DES three times)
- encrypt/decrypt/encrypt with only two keys?
- Why? 112 bits seen as enough, also if set keys to same then it's same as single-DES (back compat)

Symmetric Key Implementations – AES

- AES – Advanced Encryption Standard (2001)
  - replaces DES
  - NIST had a contest to find new standard
  - Rijndael won (Rhine-dahl)

    developed by two Belgian cryptographers Joan Daemen and Vincent Rijmen
  - NSA allows for classified data

    Intel chips have AES instructions

    Galois Field Theory (Gal-wah) interesting math guy

AES Notes

- Block size 128 bits (really, 4x4 array of bytes)
- 128, 192, 256 bits supported
- 10, 12 or 14 rounds based on size
- Each extra bit of key length doubles search space

AES Encryption

1. Key Expansion (using key schedule)
2. AddRound on initial key (add/xor on round key)
3. 9/11/13 rounds (depending on key size)
   (a) SubByte: non-linear substitution (w lookup table)
   (b) ShiftRows: transposition/row shift
   (c) MixColumns: mix columns (matrix multiply)
   (d) AddRound (xor again)
4. Final round: a,b,d again

AES Attacks

- In theory take billions of years to brute force
- "Attack" means finding some way to decode key faster than brute force
- Have been some but none really effective yet
  biclique attack reduce search for 128-bit to 2^{126} but that's still large
- Side Channel Attacks are possible though

AES Performance

- Pentium Pro 200MHz: 11 MBits/s
- Modern Intel/AMD with AES in hardware, multiple GB/s

Asymmetric / Public Key Encryption

- Asymmetric/Public Key
- Key exchange is weakest link of symmetric encryption, as both sides need it and if it leaks, all is lost
- Have a public key that anyone can use to encrypt a message. Can only be (easily) decrypted by a secret, private key
- Hard to solve math problems. Integer factorization, discrete logarithm, elliptic curves

Why not use Asymmetric for everything?

- Often only used to encrypt small amounts of data, i.e. used to encrypt a symmetric key used for longer transactions
- High overhead and requires high-quality random numbers, hard to use it for large amounts of data

Uses of Public Key Crypto

- public key encryption
  - public key used to encrypt message only holder of private key can decrypt
- digital signature
  - message signed with private key and anyone with access to public key can verify the original sender

RSA

- Rivest/Shamir/Adleman at MIT (1977)
  Discovered before by UK govt (1973) but classified
- Choose two large primes p and q (1024+ bits)
- Compute: n=p*q, z=(p-1)*(q-1)
- Choose number relatively prime to z: d
  (no common factors)
- Find e such that e*d mod z=1
- Divide plaintext into blocks $0 \leq P < n$, blocks of k bits
  where k largest $2^k < n$

- To encrypt, compute $C = P^e \ mod \ n$
- To decrypt, compute $P = C^d \ mod \ n$
- public key is e,n. private key is d,n
- Hard to break as you need to factor n (hard)
- How do you find p and q? Generate random number, then apply various tests to determine if prime (there are algorithms for that)

RSA Example

- Example from Tanenbaum Figure 8-17:
  Pick two large primes: p=3, q=11
  n=p*q=33, z=(p-1)*(q-1)=20
  d=7 (no common factors with 20)
  $7 * e\ mod\ 20 = 1$ so e=3
  private key=7,33 public key=3,33
  To encrypt "13", $13^3 = 2197, mod33 = 19$
  To decrypt "19", $19^7 = 893871739mod33 = 13$

Why RSA Not Used Anymore

- Needs really good random primes, if you pick bad primes can be easier to crack (if p and q too close together)
- Slow, so on low-power devices tempting to pick low value exponents
- Adding more bits only slowly adds better encryption
- No random element, so can tell if the same message sent twice because will encrypt to the same (or can brute force easier)

  Fix to this is random padding at end

- Improper padding can lead to “padding oracle” attack (if you get an invalid padding error on invalid cyphertext, can slowly work your way to the key)

RSA Replacements – DSA

- RSA 2048 bit but even that might not be enough
- DSA (NIST 1991 / FIPS 1993)
  - built on modular exponentiation / discrete logarithms
  - Roughly same security with keysize as RSA

RSA Replacements – ECDSA

- ECDSA – elliptic curve cryptography (ECC) (1999)
  - Algebraic structure of elliptic curves on finite fields
  - Same security with smaller keys than RSA/DSA
  - Endorsed by NSA
  - 1024 bit RSA equivalent to 160 bit ECC
- EdSA (not same as ECDSA)
  - Edwards curve, “Schnorr Signature”
  - github using ed25519: SHA-512 plus curve 25519 based on $y^2 = x^3 + 486662x^2 + x$ with prime $2^{255} - 19$

Quantum Computing

- Many of these algorithms (especially prime based) are vulnerable to quantum computing algortithms
- If quantum computing ever happens, then existing crypto is not strong
- There are new quantum-resistant algorithms
- Newer versions of ssh have started warning if you are using quantum vulnerable algos

```
** WARNING: connection is not using a post-quant
** This session may be vulnerable to "store now,
** The server may need to be upgraded. See https:
```

Cryptographic Hash Functions

- Maps a document of arbitrary size to a fixed size
- Easy to calculate, hard to reverse. Only real feasible way to reverse is brute-force search
- Break file up into chunks, do a series of operations to "compress" it, often shift, xor, or, add, and, not
- Small changes in document should lead to very different hashes

Hash Collisions

- Should not be able to find two different messages with same hash
- Two items with same hash are a collision
- Are collisions useful? If you can map documents of same filetype, or if somehow same document with lots of garbage on end

Cryptographic Hash Algorithms – md5

- md5 `md5sum` (Rivest) (1991, replacing md4)
- 128-bit md5 hashes, create checksum, almost uniquely ID file

  supposed to be unlikely to get collision
- Been broken, easy to defeat since 2007
  - Birthday attack, while creating two files with same sum hard, creating a huge number of files the likelyhood of getting two to be the same is more likely than you think

◦ Chosen-prefix attack – in this case take two differing start texts, by appending arbitrary data to each (in a comment section in some formats like PDF) can find match

HW#3 – md5 collision (extra credit) note

- md5 is broken, but it's still not possible to "reverse" a hash in a reasonable (less than billions of year) time. (i.e., given a hash you can't make a file that creates it)
- It is trivial (less than a second) to make two random files that have the same md5sum
- A chosen-prefix attack is also possible (hours?), where given two files you can append garbage at the end until you find a matching md5sum for them
- The "include md5sum in itself" and the "make

two images with same md5sum" use weaknesses in various file formats where parts of files can be ignored/commented and also tiny changes can trigger colors or patterns to change in a way that has the desired effect