Euclidean Alg.

$a^{-1} \bmod b$; a & b must be relatively prime!

$b = a(x) + c_1$

$a = c_1(y) + c_2$

until $c_n = 1$

then flip eqtn to be $1 = 2 - (z)\#$

Ex: $5^{-1} \bmod 7$

$7 = 5(1) + 2$

$5 = 2(2) + 1$

$2 = 1(2) + \emptyset$

$1 = 5 - 2(2)$

$2 = 7 - 5(1)$

$1 = 5 - [2(2)]$

$1 = 5 - 7(2) + 5(2)$

$1 = 5(1) - 7(2) + 5(2)$

$1 = 5(3) - 7(2)$

$1 = 15 - 14$ → $(a^{-1}) = 3$

Negative factor

- 'Add' to modulus; result is a^{-1}

Ex: Modulus 11

"inverse factor" = -4

$11 - 4 = (7 = a^{-1})$

Knapsack

GK = Hard to solve

SIK = Easy; attacks > each sum of prior wts

Line up binary to SIK

(~~ESB to Right~~)

compute GK from SIK

$SIK_1 \cdot n \bmod n = GK_1$

$SIK_2 \cdot n \bmod n = GK_2$

. . . .

$n^{-1} \bmod n = a$

Private key = (a, SIK, n)

Public key = GK

E → Line up Binary to GK

D → Subtract from C/t; 1 = used, 0 = unused

RSA

Rivest, Shamir, Adleman

Public key = (N, e)

Private key = d

Message = M

Encrypt → $C = M^e \bmod N$

Decrypt → $M = C^d \bmod N$

Signature

↳ $S = M^d \bmod N$

Verify

↳ $M' = S^e \bmod N$

M' should equal M

Key Distro Problem

Key safety paramount

↳ Msg only secure as protection method

Kerckhoff's

- System known to Attkr
- only secret is the key

Not Algrtm

- secret Algrtm's never stay secret
- find weaknesses in lab, NOT wild/production

Secure Crypto Sys

- Best known Atk is Brute-force (no shortcut exists)

Insecure

↳ shortcut exists/known!

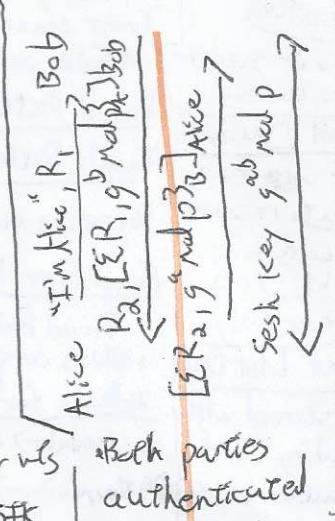
Avalanche Effect

1 bit change inputs about 1/2 output bits

CSRNG

★ statistically random

Pattern cannot exist/be predicted



- Both parties authenticated
- Comm session secured
- PFS established

Both have shown they're active

$g^{ab} \bmod p$ isn't used anywhere else (anti-snoop)

Combined Encryption & Digital Sigs

Confusion

obfuscation of msg

Diffusion

jumbling/mixing of msg

Substitution Ciphers

confusion only

Letter Frequency Analysis breaks this "cipher"

Ex: Caesar's Cipher

Transposition Cipher

Diffusion only for 3x3 Grid Msg

Ex: (2,3,1) (3,1,2)

mix Rows & Columns

cipher text = jumbled msg

One Time Pad (OTP)

★ only diffusion! But one of few provably secure Algrtm provided it's never reused

- key length == msg length

Any msg can be extracted but 0% accurate as there's nothing to compare against

Use only once then use new one time pad

Stream Cipher

- based on OTP

key stretched into keystream, which is used like OTP

Block Cipher

- creates codebook from each key

Both Conf. & Diff.

AS 1 is one

Iterated Block Cipher

Plain- & cipher text composed of fix-size

C/t = (p/t) qty Iterated Rnds

Rnd Input → Key & prior Rnds output

★ typically S/W implement

Feistel Cipher (type of Iterated Block)

Splits P/t in-half (L & R)

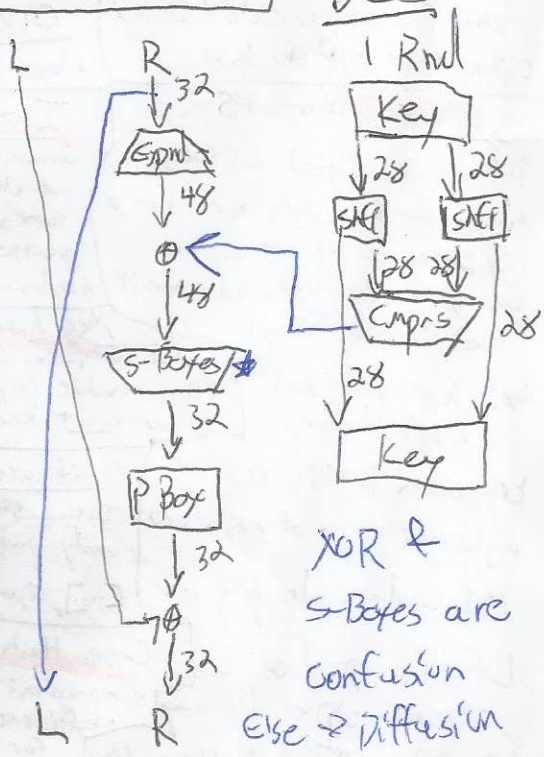
Computes i → n (up thru n) each Rnd

$L_i = R_{i-1}$ $\quad R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$

f → Rnd function $\quad K_i$ → subkey

$C/t = (L_n, R_n)$

DES



XOR & S-Boxes are Confusion

Else → diffusion

S-Boxes are special

what make DES, 56-bit key used)

RECALL: OTP & XOR are both confusion

AES (Rijndael)

Block → 128 bits

Rnds → 10 to 14 ← Key lengths dependent 128, 192, 256

Each Rnd: 4 functions across 3 layers

LFSR (4-bit ex)

Tap Seq. = bits to use for function

Tap S_4 [3, 0]

0	1	2	3

Initial Key (1, 1, 0, 1)

Plaintext = 1010

tap seq

	0	1	2	3	KSS	P	C
X	1	1	0	1	\	\	\
xt1	0	1	1	0	1	1	0
xt2	0	0	1	1	0	0	0
xt3	1	0	0	1	1	1	0
xt4	0	1	0	0	1	0	1

function is XOR in this ex.

X = initial key bits

Tap. function executes, bits shift, advance to xt1, write bits to table & calculate xt2 (as xt1 is calc'd @ X)

Cipher text = P ⊕ KSS

AES Rnd functions

ByteSub → nonlinear; Confusion

ShiftRow → linear mix; diffusion

MixColumn → nonlinear; Confusion

Add Rnd Key → key addition

underline indicates layer

E.C.B. Mode

Encrypts each block independently

But subject to "cut & paste" attack; can shuffle blocks but like blocks still the same

C.B.C. Mode

chains blocks

★ uses I.V. to initialize L Rand, not secret

Akin to codebook w/ Additive

$C_0 = E(IV \oplus P_0, K)$ Encrypt

$P_0 = IV \oplus D(C_0, K)$ Decrypt

$C_1 = E(C_0 \oplus P_1, K)$ Encry

$P_1 = C_0 \oplus D(C_1, K)$ Decry

CTR Mode

★ Good for Rand Acc

• Uses block cipher akin to stream

A5/1 An ~~LFBR~~ LFSR

Different sized LFSRs

Majority bit: 2/3 are 1 or 0

↳ If majority bit agrees on given Register, advance it else Register stays

Ex: X=1 y=0 Z=1

X & Z advance

Keystream Bit is XOR of rightmost bits of registers

not values dropping off

LFSRs produce a stream cipher this way

★ fast in H/W; Hard/Slow S/W

Each step (thru LFSR) produces a bit

RC4 stream cipher

★ self modifying lookup tbl

Fast in S/W

Each step produces byte

PKE → utilizes one-way "trapdoor" Easy computation 1-way, v. hard other way

Integrity

detect unauthorized write (modification of data) Confidentiality comes from encryption (prevention of unauthorized disclosure)

Msg Authentication Code

useful for integrity, computed from C.B.C. residue (final cipher block) MAC

★ Recv must know key 'k'

Certificate Authority (CA)

Trusted 3rd Party (TTP) to verify public certs w/

$[\ldots]_x$ signature $\{\ldots\}_x$ Encryption

Crypto Hash function h(x) must:

• Compression: small output length

• Efficient: h(x) simple comptn for any x

• one-way: Given y, infeasible to find an x such h(x)=y

• Weak collision Resist: Given x & h(x) infeasible to find y≠x such h(y) = h(x)

• Strong collision Resist: infeasible any x & y, w/ x≠y such h(x) = h(y)

★ Desires Avalanche Effect

Crypto Hash consists of arty # of Rounds ★ more speed and security; Avalanche Effect after a few simple Rnds ideally

Perfect Fwd Secrecy

prevents session keys being compromised even if other secrets of exchange compromised

g^ab mod p never transmits clear text

Authentication

★ cannot do simply by recieving!

★ Multiple comm attempts can be open simultaneously

Passwords Passphrases

Proving Identity

• Knowledge (something you know)

Ex: 3 day, password, etc.

• Membership (what you are)

Ex: fingerprint, Retina Scan

• Possession (what you have)

Ex: phone recieving verify text or Authenticator application

Authentication verifies ID of user/service

Authorization determines if suc/user has rights to access

Access Control Matrix

Full D.B. of who can access what resource

ACLs & C-Lists are slices of this; ACM grows exponentially

Access Control List (ACL)

permissions stored with files detailing what level(s) of access user(s) possess

Capabilities Lists

permissions stored with users; avoids Confused Deputy

Confused Deputy

★ privilege escalation issue ★

User doesn't have permissions but process/service does

Uses it's permissions instead of the user's permissions passed to it

C-Lists avoid this situation

Password Hashes

Should be stored, not clear text password!

Salt random string pre-pended or appended to password prior to being hashed. Due to Avalanche Effect, vastly different resulting hash is generated

★ helps defend vs Brute-force, Dictionary, and Hash Table (Rainbow Table) attacks

Biometrics (something you are)

• Iris scans fakeable via AI (can make amalgam from public image searches of target).

↳ Retinal scan more reliable also more expensive, takes longer

★ All Biometrics require obtaining a baseline (enrollment period) ★

• Hand width == good measure (tiny variations when compared over time) ★ Exception: enrollment of child/teenager/elderly person

Generally better than passwords

↳ very hard to spoof/falsify

Not widely used due to expense & time (most scanners aren't easy)

★ false positives are an issue

Fraud Rate A misauthenticated as B

Insult Rate A not authenticated as A

Decrease one, Increases other

Equal Error Rate point @ which Fraud Rate == Insult Rate

• allows comparing different biometrics

Ideals for Biometrics

Universal → applies to all population

Distinguishing → high lvl of reliability

Permanent → miniscule change over time

Collectable → easy to get measurements

2FA

2 of 3 categories used to authenticate

• Biometric • Knowledge • Possession

ACM (pg 2)

subject = Rows

objects = columns

example

	OS	Account prgm	Account Data	Ins	payrl
Bob	rx	rx	r	—	—
Alice	rx	rx	r	rw	rw
Sam	rwx	rwx	r	rw	rw
Account prgm	rx	rx	rw	rw	rw

256-bit key to decrypt

very long key; high impossible to memorize. User will store insecurely; may as well not even have this

Non-repudiation

inability to refute or adherence to obligation

Ex: legal signature on a contract

Incomplete Mediation

* Failure to validate input

Ex: Client (web browser) validates form data but Server doesn't Trudy could alter form data prior to Server's receipt

Software Reverse Engineer

Anti-Disassembly: prevent or delay reverse engineering of code

Anti-Debugging → intended to prevent anlysis

Tamper-Resist → difficult for Atkr to modify code

Code Obfuscation → Decrypted EXE segments and/or nonsensical code present to prevent/delay

Collisions (sarcasm)

feature, not a problem!

- Hashes map any input to a fixed-length output, thus output could "collide" with another output, even though inputs differ

weak collision Resist

Ex: stored passwords in DB. Creator of password knows it, but we store hash not cleartext

(weak) => infeasible ~~to~~ find another input for same output

(Strong) => infeasible any other input generates identical output

* Data integrity checker
- Seen w/ digitally signing docs

MLS Compartments

"keys" within ~~clearance~~ classifictn levels Ex: TS {Dog}

Those with the clearance corresponding to this key

If multiple keys Req'd @ classification level, must possess them all

Ex: TS {Cat, Dog} but only have {Cat} clearance? cannot access!

Static Analysis Examining malware w/o running it

- Basic → No instruction inspection but can miss key behaviors, fails on advanced malware but it's quick
- Advanced → Complex, requires understanding Assembly

Dynamic Analysis run malware, monitor effects

Ex: Hybrid Analysis website (VM sandbox runs s/w)

- Basic → ineffective on some malware (anti-Debug) can be easy but need safe isolated environment
- Advanced → examines internal (ie running) state of malware, runs debugger

Buffer Overflow

Data in buffer exceeds storage; floods into other locations

Ex of smashing the stack

Diffie Hellman Weaknesses

CPU intense WRT large keys

very susceptible to MitM Atks

- Relies on mutual trust or another means of exchanging secure session key (P.F.S.)

Canary

Run-time stack check

Dead? Indicates potential tampering of the stack

Incident Response (IR)

* Root-Cause Analysis → How did attacker get in? what did they do/change? How long did they have access? Do THEY STILL??!

- Dissecting malware is critical component-

Address Space Layout Rand.

Randomizes where code is placed in memory/stack

Buffer overflow becomes probability based now

Sandboxing safe/isolated environment for testing [malware, unknown code, new program before it goes to production] - typically a VM rather than separate full system (cheaper)

Akin to Roach Motel or our inner box from SRE & Malware lab; locked down, limited access machine to test on w/o letting malware into network / the wild

Multilevel Security (MLS)

Classifications → resources/objects

clearances → people / subjects

* Req'd when subject/object @ different levels using same system

Code Reviews

- Collaborative review of code across development teams - Akin to "spot-check" of code for errors
- Helps facilitate codebase knowledge IRL

Ethically, just because we can reverse engineer software, doesn't mean we should [w.r.t. proprietary software]

Bell-LaPadula (BLP)

Intended to express essential reqs for implementing an MLS

- focus = confidentiality; prevent unauthorized reading of objects

* No Read Up, No Write Down *

- intended to encapsulate objects-

Memory Safe Lang.

Indirectly alters memory (unlike C)

Ex: Java or C#

- Could also use memory-safe function Ex: strncpy in C rather than strcpy

Biba Model focus = data integrity prevention of unauthorized writes

* Dual of BLP *

No Read Down, No Write Up

Crypto Hash Function

Compressed → small output

Efficient → easy to compute hash for any input

one-way/Trap door → infeasible to perform the math in reverse

Attack Trees

* Means of organizing info & describing connections btwn possible Atk pos
- allows describing security sub/systems
- can build automatic D.B. describing system security
- Can capture & re-use expertise (fix issue once, can cc. elsewhere!
- ~~allows~~ facilitates decision-making on improving security or effects of new security

Root == Atk goal

Leaf == attack vector

Both required for vulnerability

Blue indicates cross-applicability
- solutions to 'Get Admin Access' also apply to mitigating 'Get another's credentials'