Crypto

Def'n
- Confidentiality: Adversary cannot read/determine M; no information leaked
- Integrity: Adversary can't modify/tamper with M w/o detection
- Authenticity: Allows us to determine who created message
  - Requires integrity to hold
  - Idiom: Alice sends message w/ corresponding signature to Bob; signature can't be generated by anyone other than Alice
- Deniability: No guarantee over person created message, e.g. Alice & Bob use same key to generate msg; could've come from either party
- Symmetric: Alice & Bob both know and use some secret key
- Public Key: Bob have private key and corresponding public key; Alice encrypts under this public key, ↪ slower than sym. key enc.
- Asymmetric: Public Key (digital) Signatures provide integrity + authenticity
  - Alice uses her private key to gen. signature per message; appends to msg.
  - Bob uses Alice's public key to verify no tampering
- Kerckhoffs Principle: Sys should remain secure when internal details are known to attacker (aside secret key)

	Symmetric	Asym.
Confidentiality	Blk w/chain AES-CBC	Public Key Enc ElGamal, RSA
Integrity & Auth.	MACs AES-CBC-MAC	Digital Signatures RSA sig.

Threat Models

- Ciphertext only: Eve only observes single encrypted msg
- Known plaintext: Eve along w/partial information about plaintext msg
- Replay: Eve, w/o knowing msg, can repeatedly send encrypted msg
- Chosen-plaintext: Eve can have Alice encrypt arbitrary msg
- Chosen-ciphertext: Eve can trick Bob into decrypting arbitrary msg

IND-CPA: Indistinguishable under plaintext attack
1] Eve sends M_0, M_1 to Alice
2] Alice randomly chooses one to encrypt & send back to Eve
3] Eve may ask for encryptions of chosen plain text (excluding M_0/M_1)
4] Eve then guesses what msg was encrypted
- For IND-CPA, no information should leak, thus Eve has $50\% + \varepsilon$ of guessing correctly
- Any deterministic scheme isn't IND-CPA

Hashes: Deterministic func $\{0,1\}^* \to \{0,1\}^n$ (any len, looks random)
- One-way: Given $y = H(x)$, it's hard to find any x s.t. $y = H(x)$
- 2^{nd} Preimage Resistant: Given x, hard to find $x' \neq x$ s.t. $H(x) = H(x')$
- Collision Resistant: Hard to find any pair (x, x') s.t. $H(x) = H(x')$
- ⇑
- SHA2 susceptible to length extension as it's internal state is revealed
  Given $H(M)$, can produce $H(M \| M')$ $\forall M'$

Public Key

El Gamal: Public $\begin{cases} p: \text{large prime} \\ g \in [2, p-1) \end{cases}$
- Bob's secret = b; public = $B = g^b \bmod p$
- Alice's " = $r \xleftarrow{\$} [0, p-1)$; " = $R = g^r \bmod p$
  ↪ sends $\{C_1, C_2\} = \{R, M \times B^r \bmod p\}$
  ↪ Decrypts $C_2 \cdot C_1^{-b} = [M \times B^r \bmod p] \times R^{-b}$
  $= M \times g^{br} \times g^{-br}$
  $= M \bmod p$

Diffie-Hellman: Key exchange protocol based on El Gamal
1] Alice & Bob randomly choose (secret) $a, b \in \{0, 1, \dots, p-2\}$
2] They announce $A = g^a \bmod p$, $B = g^b \bmod p$
3] Symmetrically compute $S = B^a = A^b = g^{ab} \bmod p$
- Holds following discrete log problem: if $F(x) = g^x \bmod p$ is given, it's infeasible to determine x since $F(x)$ is one-way
- DH lacks integrity & auth, susceptible to MITM attacks; resolve w/ digital signatures

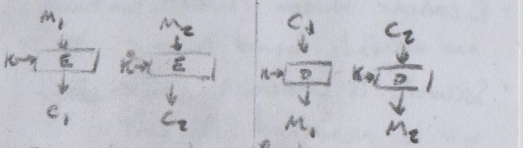
Trapdoor func: Like one way; given y, it's hard to find x s.t. $x = f(y)$ unless special backdoor K is given
- RSA: $N = pq$; given $c = m^e \bmod N$ and e, it's hard to find m w/o knowing p or q
- Disc Log: With a or b, it's easy to find $g^{ab} \bmod p$ (given $g, p, A = g^a \bmod p$)

Message Authentication Codes (MACs)

- Hashes are unkeyed, anyone can make them providing no integrity, motivating MACs
- Provides no confidentiality, only int. & auth.
  ↪ Motivates Authenticated Encryption:
  - Enc-then-MAC: $\langle Enc_K(M), MAC_{K_2}(Enc_{K_1}(M)) \rangle$
    ↪ Verify MAC, then decrypt
  - MAC-then-Enc: $E_{K_1}(M \| MAC_{K_2}(M))$
    ↪ Side channel attacks possible
    ↪ Doesn't provide ciphertext integrity

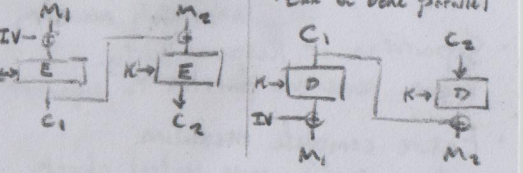
Block Cipher Modes of Operation

ECB: Electronic code Block
Break M into n-bit blks $M_1 \cdots M_\ell$
$C_i = E_K(M_i)$ $\quad M_i = D_K(C_i)$

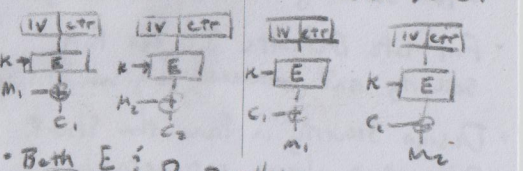


- Block independence leaks redundancy (across blks, possibly across messages)
- Not IND-CPA bc. determinism

CBC: Cipher Block Chain
Choose & send random IV ⇒ IND-CPA
$C_0 = IV$; $C_i = E_K(C_{i-1} \oplus M_i)$ $\quad M_i = D_K(C_i) \oplus C_{i-1}$
↪ Not parallelizable ↪ Can be done parallel

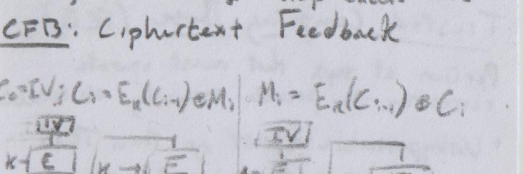


CTR: Counter
Init counter to IV; increment each blk
↪ Use ⇒ key stream, one-time pad
$C_i = E_K(IV+i) \oplus M_i$ $\quad M_i = E_K(IV+i) \oplus C_i$



- Both E & D parallelizable
- Padding not needed - just drop excess bits

CFB: Ciphertext Feedback
$C_0 = IV$; $C_i = E_K(C_{i-1}) \oplus M_i$ $\quad M_i = E_K(C_{i-1}) \oplus C_i$



- No padding needed

Security Principles

- Know your threat model. Who, what resources they have
- Consider Human Factors. Laziness and accidents bound to occur
- Security is Economics. Attack effort will be proportional to cost
  ↳ Focus only improving weakest link
- Detect if can't Prevent. Min. response time; indicate/alert to tampering
- Defense in Depth. Layer various types of defenses, requiring all must be breached
- Least Privilege. Only as much as absolutely necessary
- Separation of Responsibility. Require numerous parties to sign-off
- Ensure complete mediation. When entering access control, check every access to every object
- Shannon's Maxim. Attacker knows details about sys. they're attacking
  ↳ No security through obscurity
- Fail-safe Defaults. If sys fails, security isn't weakened (e.g. whitelisting)
- Design Security in From the Start. Difficult to change retroactively

Trusted Computing Base (TCB):

Portion of sys that must operate correctly to provide security assurances

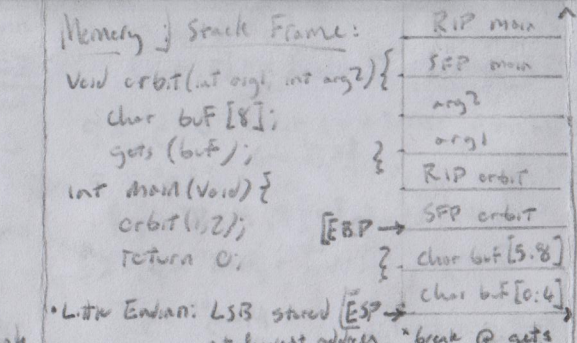
- Unbypassable - must go thru TCB
- Tamper-resistant - external influences shouldn't invalidate integrity
- Verifiable - Correctness should be assertable

Time-of-Check Time-of-Use:

Non-atomic ops lead to race conds
Value changes after check but before being used

Memory & Stack Frame:

```
void orbit(int arg1, int arg2) {
    char buf[8];
    gets(buf);
}
int main(void) {
    orbit(1,2);
    return 0;
}
```



- Little Endian: LSB stored at lowest address
  ↳ Address 0xDEADBEEF ⇒ EF | BE | AD | DE
- Global vars (outside fcns) put on heap
  ↳ Last defined has highest address
- Structs have 1st member at lowest addr.
- char[M] is length M/4 Words = M bytes
- int [L] " " L Words = 4L bytes

Off-by-one:

- Change SFP LSB s.t. points to 'A'×4 + &SHELLCODE
- 1st rt will move EBP to address of 'A'×4
- 2nd rt moves EBP to 'A'×4 (don't care!) then EIP to &SHELLCODE

Integer conversion: xFFFF = $2^{16}-1 = 65536$ when unsigned (i.e. size_t) but -1 when interpret as signed

- int = -1 will pass [int] > K ∀ K ≥ 0
- Also consider overflow

String Format: Specifier tells # bytes
Begins 8 bytes above printf's RIP

- %c : character (read 4, but print 1 byte)
- %[0]s: print [0]-bytes starting from arg
- %s : dereference arg, print string value at that addr until '\0'-byte
- %n : dereference arg, write # chars printed thus far at said address

Calling Convention:

Setup
1. Push args to stack, reverse order
2. Push old EIP onto stack (RIP)
3. Move EIP to new fcn code
4. Push old EBP (SFP)
5. Move EBP down to ESP
6. Move ESP down for new frame

Teardown
7. Execute fcn
8. Move ESP up to EBP
9. Restore EBP; pop SFP
10. Restore EIP; pop RIP

Tidy
11. Remove arg(s) from stack

Mitigations: For mem.-safety vulns

Memory safe lang: Many modern langs can prevent 100% of mem vulns

Writing Mem. safe Code: Defensive programming
Defn pre & post conditions, invariants!

Building Secure Software: Use tools to analyze/patch vulns; Find mem leaks, Fuzz

Exploit Mitigations: Cause crashes when unsafe behavior occurs; defense in depth + failsafe

- Non-executable pages. Either writable or executable memory; limit where EIP can point
  ↳ Counter: return to libc: overwrite RIP w/library fcn we can harness, ex execv
  ↳ Counter: return-oriented programming: put chain of return addresses at RIP that end in 'ret'
- Stack Canary. Place dummy vals below saved registers, above local vars
  - If val changes during execution, crash
  - Dummy val determined at runtime
  ↳ Counter: Brute force 2^{24} values (last byte = '\0'; Feasible for 32-bit)
  ↳ Counter: Leak canary, eg. format string vuln to print the stack
- Pointer Authentication (PAC). 64-bit has unused bytes for ptrs; store dummy val. here
- Address-space Layout Randomization (ASLR)
  - Shuffle locations of stack/heap/code/static causing absolute addresses of regs/local vars/code (Libs) to be random each run
  ↳ Counter: Guess. 32-bit has 16 bit entropy (1/2^{16} chance)
  ↳ Counter: Leak addrs, eg. print stack and use relative offsets

GDB:

- Info frame: Lists stack addresses of SFP & RIP
- printing: 1st byte will be lowest address; rightmost byte in word

2^n	n_2	n_{16}
1	0000	0
2	0001	1
4	0010	2
8	0011	3
16	0100	4
32	0101	5
64	0110	6
128	0111	7
256	1000	8
512	1001	9
1024	1010	A
2048	1011	B
4096	1100	C
8192	1101	D
16384	1110	E
32768	1111	F