CYENG 312/GECE 594 : Trusted Operating System (OS)

Lecture 05: Cryptography

Instructor: Shayan (Sean) Taheri, Ph.D.

Assistant Professor

The Department of Electrical and Cyber Engineering (ECE)

The Institute for Health and Cyber Knowledge (I-HACK)

The Gannon University (GU)

GANNON UNIVERSITY

Personal Information

- Name: Shayan (Sean) Taheri.
- Date of Birth: July/28/1991.
- Past Position: Postdoctoral Fellow at University of Florida.
- Ph.D. Degree: Electrical Engineering from the University of Central Florida.
- M.S. Degree: Computer Engineering from the Utah State University.
- University Profile: https://www.gannon.edu/FacultyProfiles.aspx?profile=taheri001

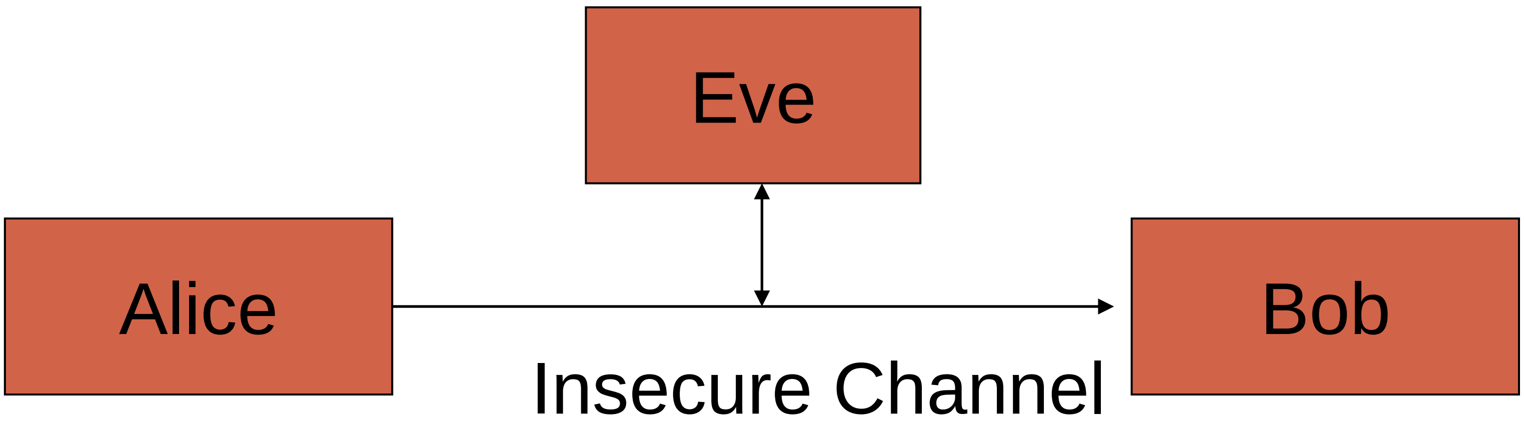
Open Problems

- Alternatives to passwords?
  - *The secret should be easy to remember, difficult to guess, and easy to enter into the system.*
- Better ways to make user choose stronger passwords?
- Better ways to use other devices for authentication
- Effective 2-factored and/or out of band authentication for the Web
- Phishing defense

Definitions

- Cryptography = the science (art) of encryption
- Cryptanalysis = the science (art) of breaking encryption
- Cryptology = cryptography + cryptanalysis

Cryptography Goals



- Encryption – Prevent Eve from intercepting message
- Authentication – Prevent Eve from impersonating Alice

Symmetric (Secret) Key

- Alice and Bob share a secret key, K_{ab}
- Encryption – Plaintext message is encrypted and decrypted with K_{ab}
- Authentication – Alice proves to Bob that she knows K_{ab} (e.g. a password)

Public Key Encryption

- Bob generates 2 keys, K_{eb} and K_{db}
- Bob publishes K_{eb} (public key)
- Alice encrypts:
  ciphertext $C = E(K_{eb}, \text{plaintext } P)$
- Bob decrypts: $P = D(K_{db}, C)$
- It must not be possible to compute K_{db} (private key) from K_{eb}

Digital Signatures

- Alice generates K_{ea} and K_{da}
- Alice publishes K_{ea}
- Alice signs plaintext P: $(P, S = D(K_{da}, P))$
- Alice sends P, S to Bob
- Bob verifies that $E(K_{ea}, S) = P$
  (since only Alice knows K_{da})

Combining Public Key Encryption and Authentication

- Alice encrypts with Bob's public key:
  $C = E(K_{eb}, P)$
- Alice signs with her secret key:
  $S = D(K_{da}, C)$
- Alice sends S, C to Bob
- Bob verifies $E(K_{ea}, C) = C$
- Bob decrypts: $P = D(K_{db}, C)$

Cryptographic Attacks

- Ciphertext only: attacker has only ciphertext.
- Known plaintext: attacker has plaintext and corresponding ciphertext.
- Chosen plaintext: attacker can encrypt messages of his choosing.
- Distinguishing attack: an attacker can distinguish your cipher from an ideal cipher (random permutation).
- A cipher must be secure against all of these attacks.

Kerckhoffs' Principle

- The security of an encryption system must depend only on the key, not on the secrecy of the algorithm.
- Nearly all proprietary encryption systems have been broken (Enigma, DeCSS, zipcrack).
- Secure systems use published algorithms (PGP, OpenSSL, Truecrypt).

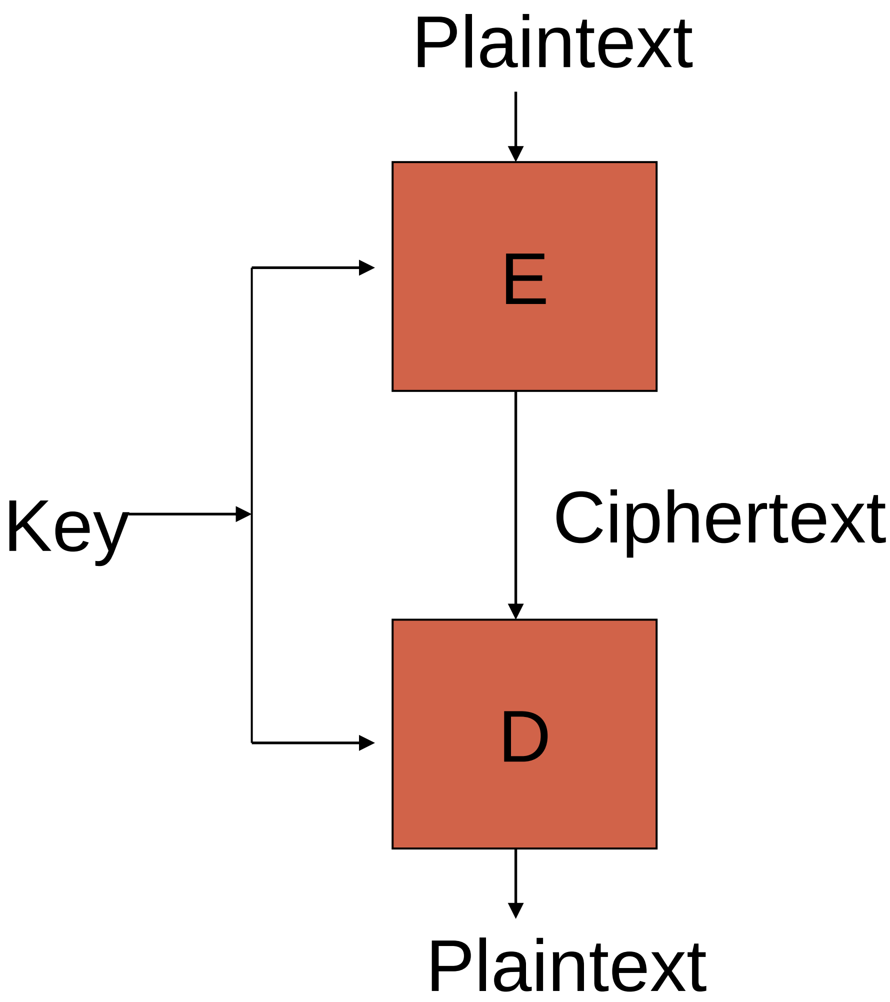
Provable Security

- There is no such thing as a provably secure system.
- Proof of unbreakable encryption does not prove the system is secure.
- The only provably secure encryption is the one time pad: C = P + K, where K is as long as P and never reused.
- Systems are believed secure only when many people try and fail to break them.

Cryptographic Algorithms

- Block ciphers (secret/symmetric key)
- Hashes
- MAC (keyed hashes)
- Diffie-Hellman key exchange
- RSA (public key encryption and digital signature)
- ElGamal digital signature

Block Ciphers



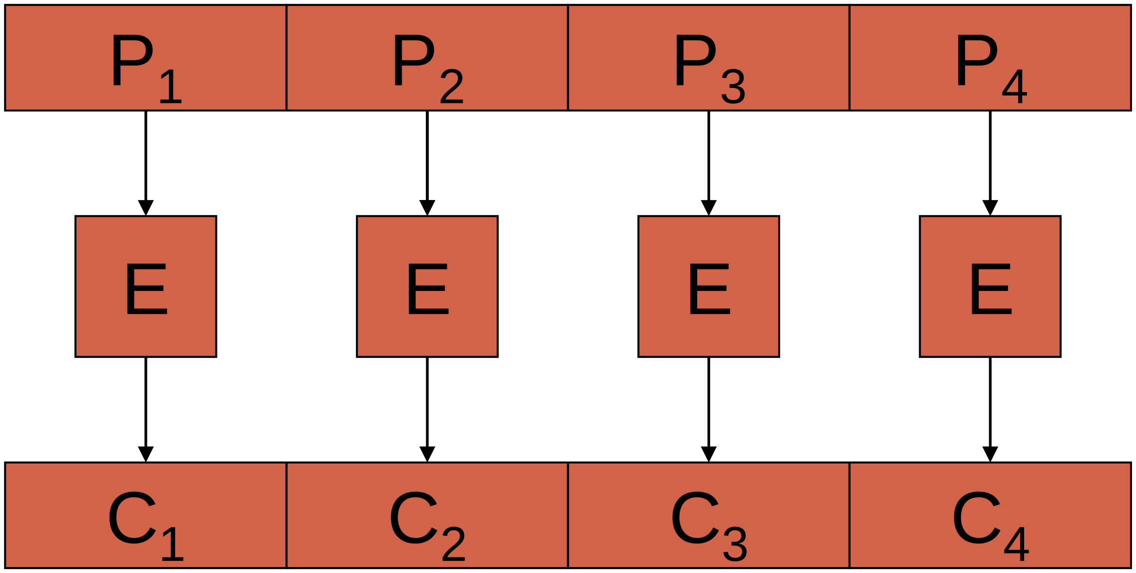
- AES
- DES
- 3DES
- Twofish
- Blowfish
- Serpent
- RC4
- IDEA
- Etc.

Encryption Modes

- ECB – Electronic Code Book
- CBC – Cipher Block Chaining
- OFB – Output Feedback
- CTR – Counter

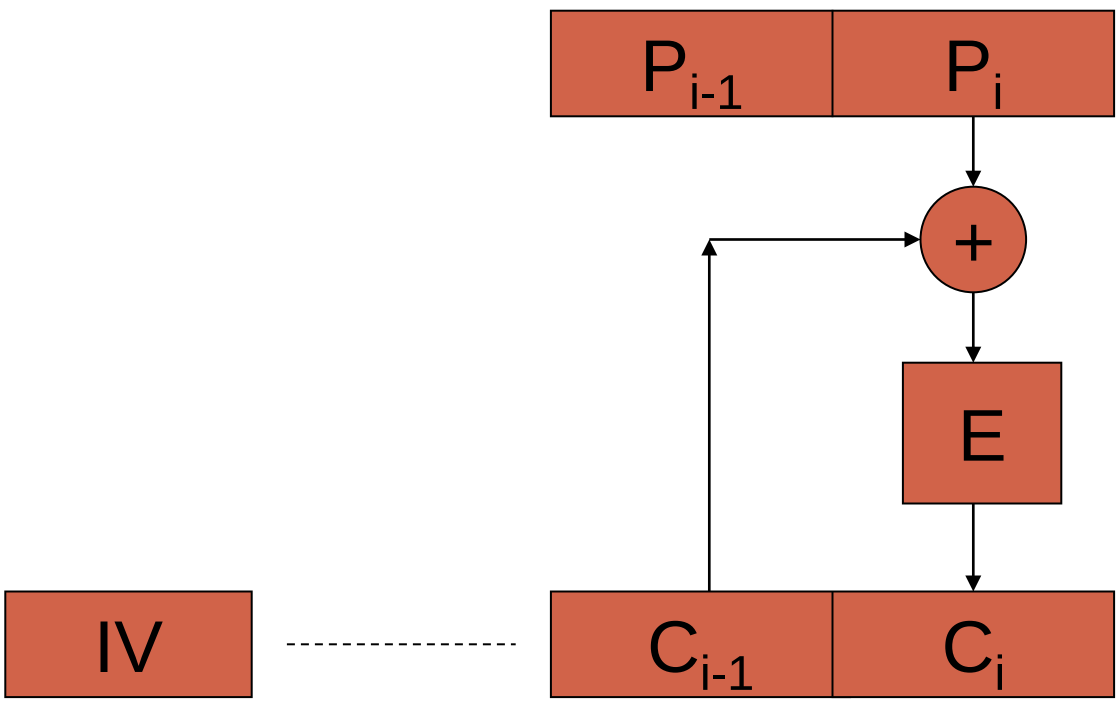
ECB Mode

- $C_i = E(K, P_i)$
- Insecure (ciphertext blocks may repeat)



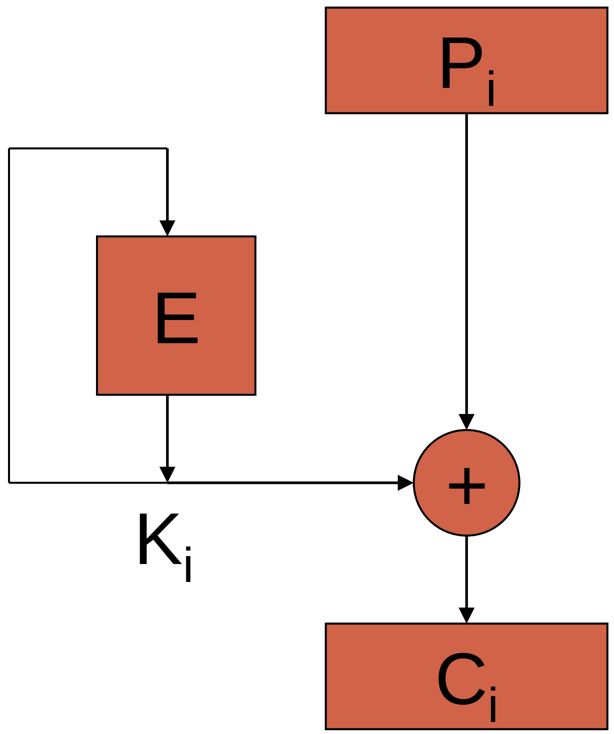
CBC Mode

- $C_i = E(K, P_i \text{ xor } C_{i-1})$
- C_0 = IV (initialization Vector) (fixed, random, counter, or nonce)
- Most popular mode



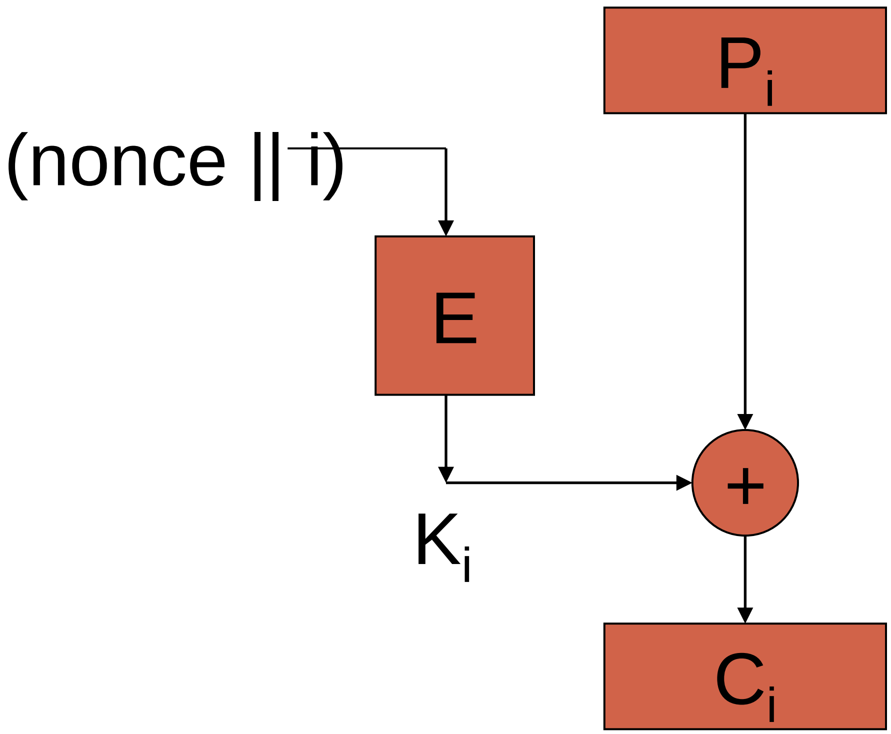
OFB Mode

- K_0 = IV (nonce = number used once)
- $K_i = E(K, K_{i-1})$
- $C_i = P_i$ xor K_i
- Not tamper resistant



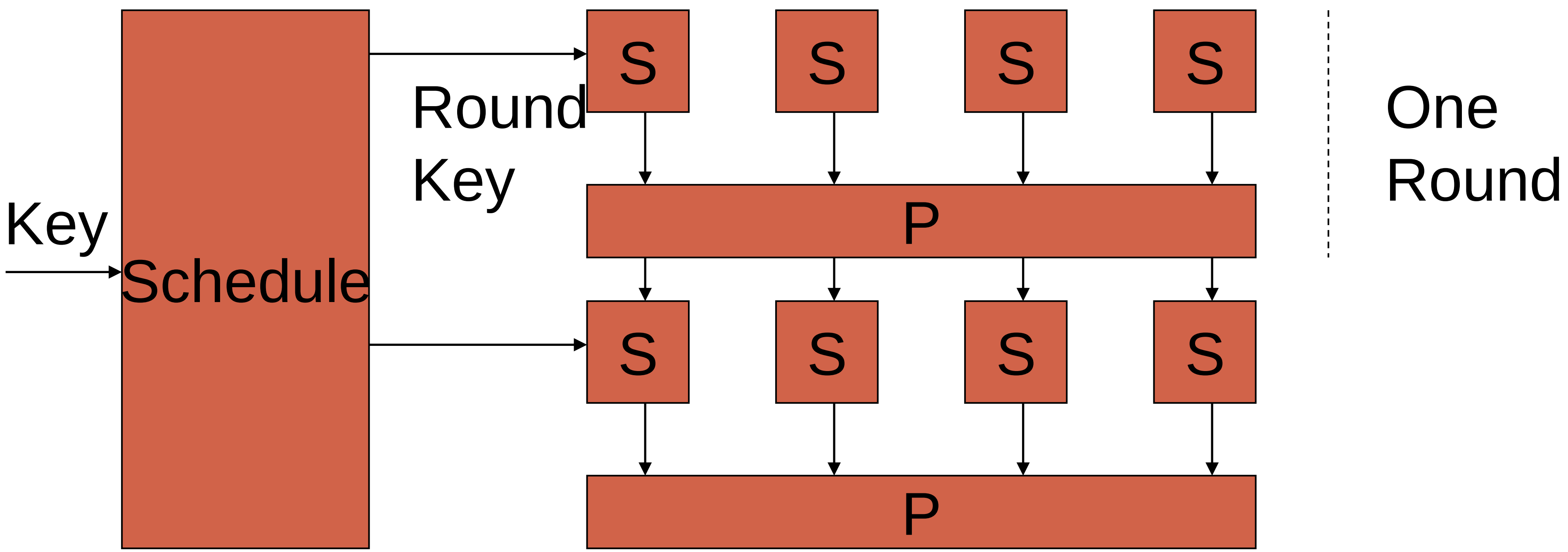
CTR Mode

- K_i = E(K, nonce || i)
- $C_i = P_i$ xor K_i
- Not tamper resistant



Block Cipher Components

- S boxes – invertible lookup tables, depends on key
- P boxes – reorder bits (may also depend on key)
- Key schedule – function of key (e.g. bit selection or simple hash)



Substitution by Itself is Weak

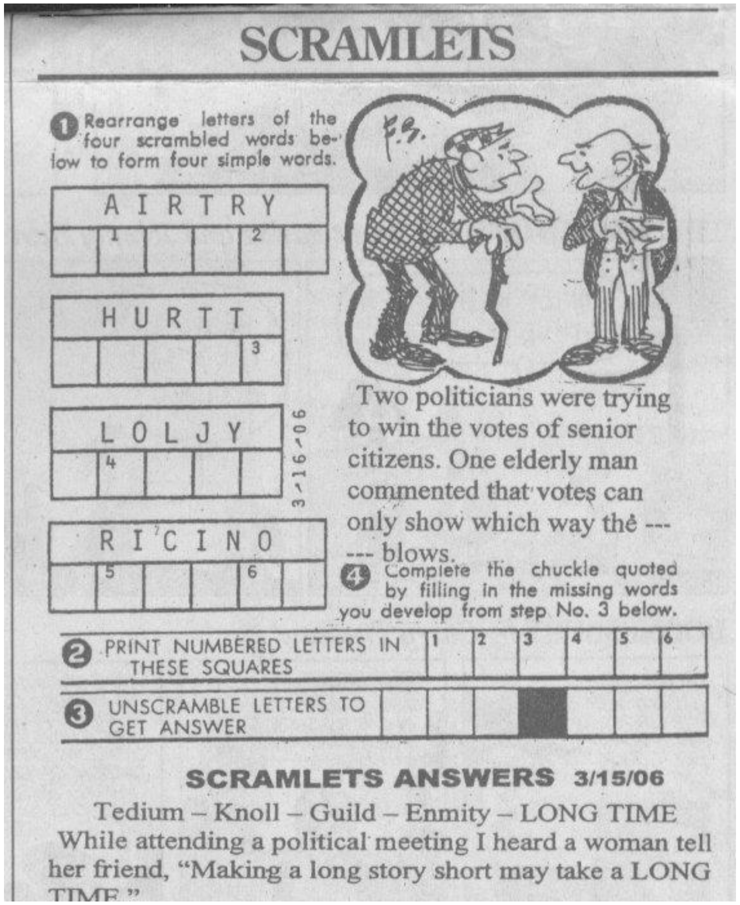
CRYPTOQUIP

Q H U Q S T V G B H T E I B T P Q U I B H V V K V L E O P I Q K B V H P Q H T E T X P H H G S L H P T L X B V U I S T L E O E T L K .

Yesterday's Cryptoquip: IF YOU HAD A LOT OF RUBBER SWATTING GIZMOS IN YOUR HOME, WOULD IT BE A NO-FLY ZONE?

Today's Cryptoquip Clue: H equals T

Permutation by Itself is Weak (Contd.)



SCRAMLETS

1 Rearrange letters of the four scrambled words below to form four simple words.

A I R T R Y

H U R T T

L O L J Y

R I C I N O

3-16-06

Two politicians were trying to win the votes of senior citizens. One elderly man commented that votes can only show which way the --- --- blows.

4 Complete the chuckle quoted by filling in the missing words you develop from step No. 3 below.

2 PRINT NUMBERED LETTERS IN THESE SQUARES

3 UNSCRAMBLE LETTERS TO GET ANSWER

SCRAMLETS ANSWERS 3/15/06

Tedium – Knoll – Guild – Enmity – LONG TIME

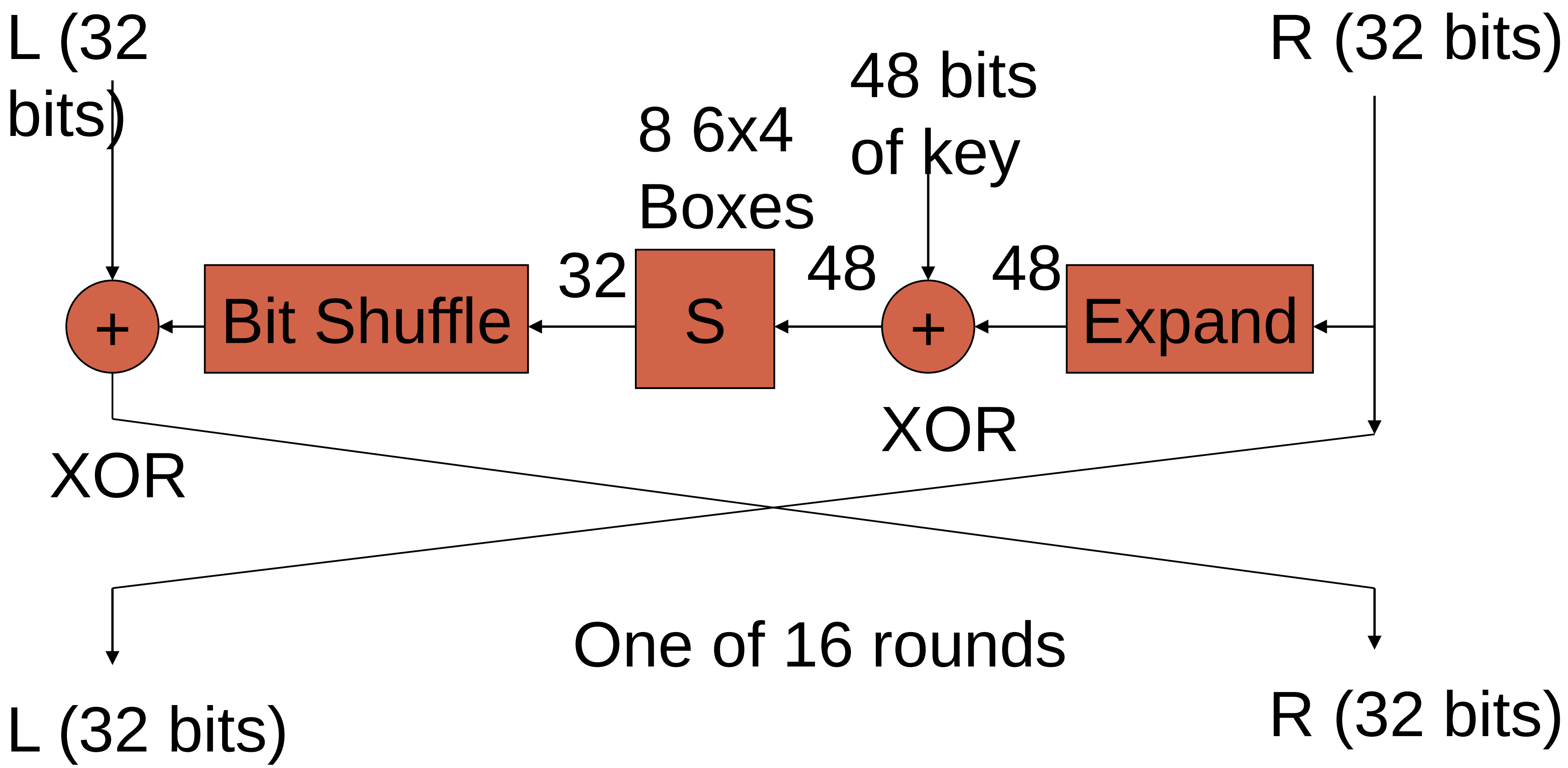
While attending a political meeting I heard a woman tell her friend, "Making a long story short may take a LONG TIME."

- But combining many rounds of substitution and permutation *might* build a strong cipher.

Data Encryption Standard (DES)

- 64 bit block
- 56 bit key
- 16 round Feistel network
- Designed by NSA and IBM in 1976 for unclassified data
- Considered obsolete due to small key and block size
- 3DES increases key to 112 bits:
  $C = E(K_1, D(K_2, E(K_1, P)))$
- http://www.itl.nist.gov/fipspubs/fip46-2.htm

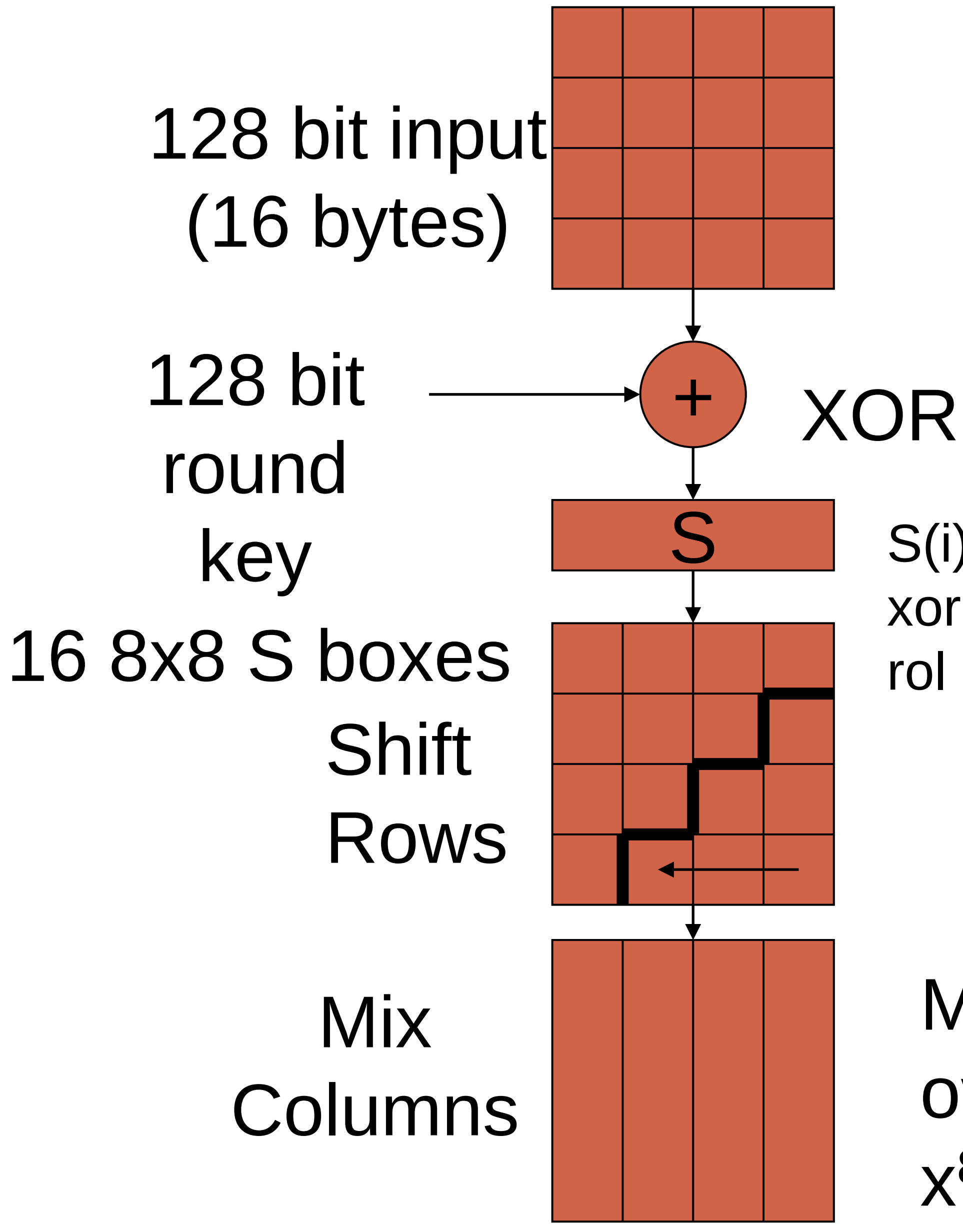
DES Feistel Network



AES - Advanced Encryption Standard (Rijndahl)

- Replaces DES
- Selected by competition by NIST in 2001
- Reviewed by NSA and approved for classified data in 2003
- 128 bit block size
- 128, 192, or 256 bit key
- 10, 12, or 14 rounds of a substitution-permutation network
- http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf

AES Round

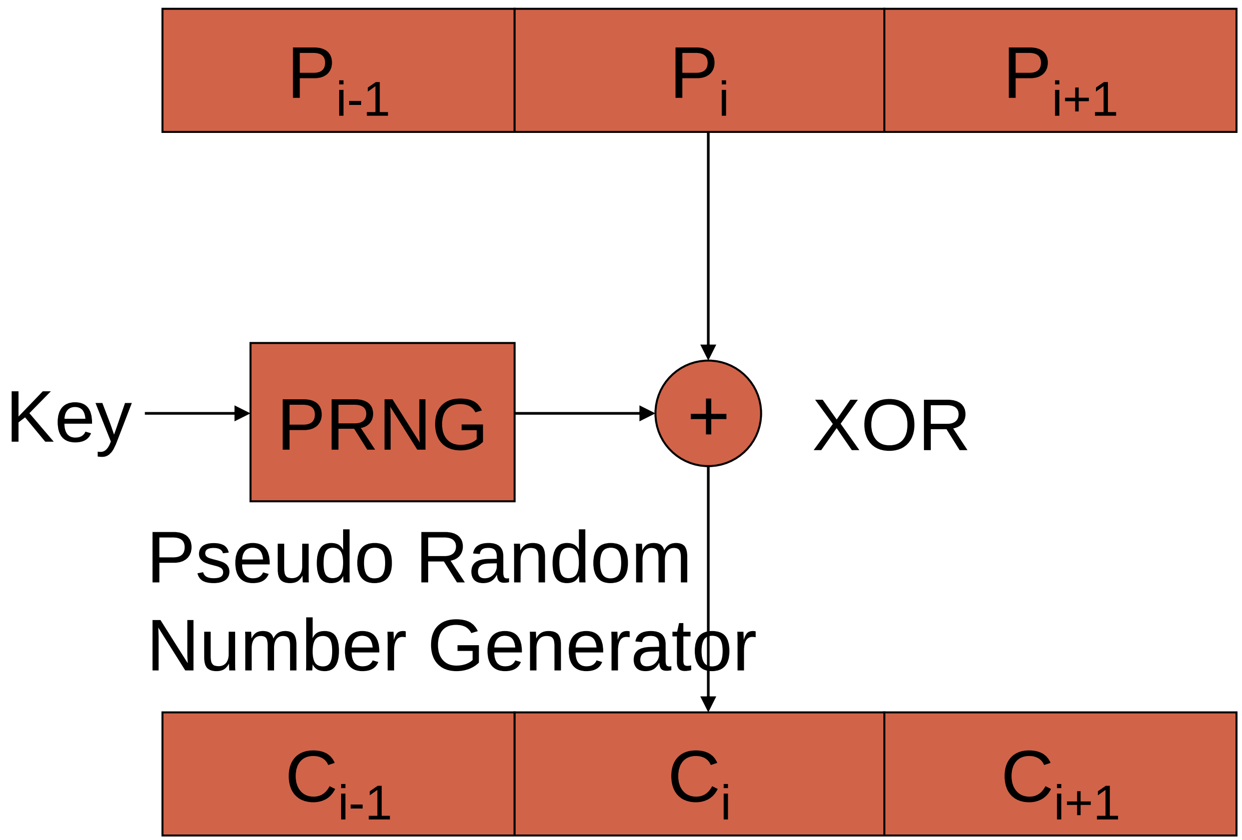


S(i) = k xor rol(k,4) xor rol(k,5) xor rol(k,6) xor rol(k,7) xor 0x63, where $k = i^{-1}$ in $GF(2^8)$, rol = rotate byte left

Multiply by M in $GF(2^8)$ over polynomial $x^8+x^4+x^3+x+1$ where M=

$$\begin{matrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{matrix}$$

Stream Ciphers



RC4 Stream Cipher

- **Key Schedule**
  - *for i from 0 to 255 S[i] := i*
  - *j := 0*
  - *for i from 0 to 255*
    - j := (j + S[i] + key[i mod keylength]) mod 256
    - swap(S[i],S[j])
- **Keystream Generation**
  - *i := 0, j := 0*
  - *while GeneratingOutput:*
    - i := (i + 1) mod 256
    - j := (j + S[i]) mod 256
    - swap(S[i],S[j])
    - output S[(S[i] + S[j]) mod 256]

RC4 Weaknesses

- Not tamper resistant.
  - *Solution: use a MAC.*
- XOR of ciphertexts with same key yields XOR of plaintexts.
  - *Solution: hash key with nonce.*
- Fluhrer, Mantin and Shamir Attack
  - *Initial keystream is non-random.*
  - *If key is simply concatenated with nonce, then key can be recovered.*
  - *Used to break WEP encryption used by 802.11b wireless networks.*

Secure Hash Functions

Message m (any size) → h → n-bit hash h(m)

- Goals
  - *Collision resistance: it takes $2^{n/2}$ work to find any m_1, m_2 such that $h(m_1) = h(m_2)$.*
  - *First preimage resistance: given $h(m)$ it takes 2^n work to find m.*
  - *Second preimage resistance: given m_1 it takes 2^n work to find m_2 such that $h(m_1) = h(m_2)$.*

Hash Applications

- Faster digital signatures: Alice signs h(P) instead of P.
- Password verification (e.g. UNIX) without storing passwords.
- Strong pseudo-random number generation.
- Message Authentication Code (MAC).

Hash Examples

- *MD2, MD4, MD5 – 128 bits (broken, http://eprint.iacr.org/2004/199.pdf http://eprint.iacr.org/2006/105.pdf)*
- *SHA-1 – 160 bits*
- *SHA-256, 384, 512 bits*

http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf

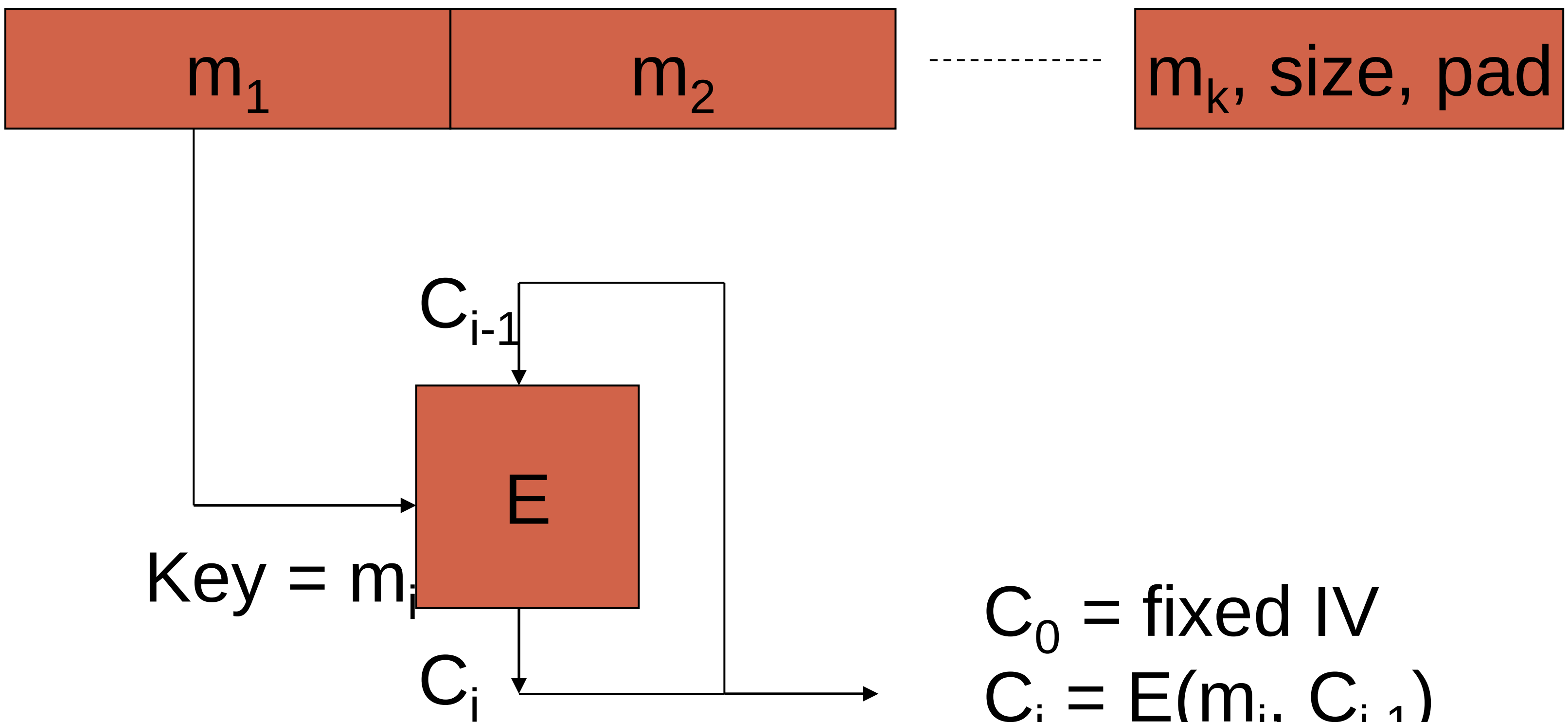
- *Whirlpool – 512 bits*
- *Tiger – 192 bits*

- Many proposed hashes have been broken.

http://paginas.terra.com.br/informatica/paulobarreto/hflounge.html

Hash Construction from a Block Cipher

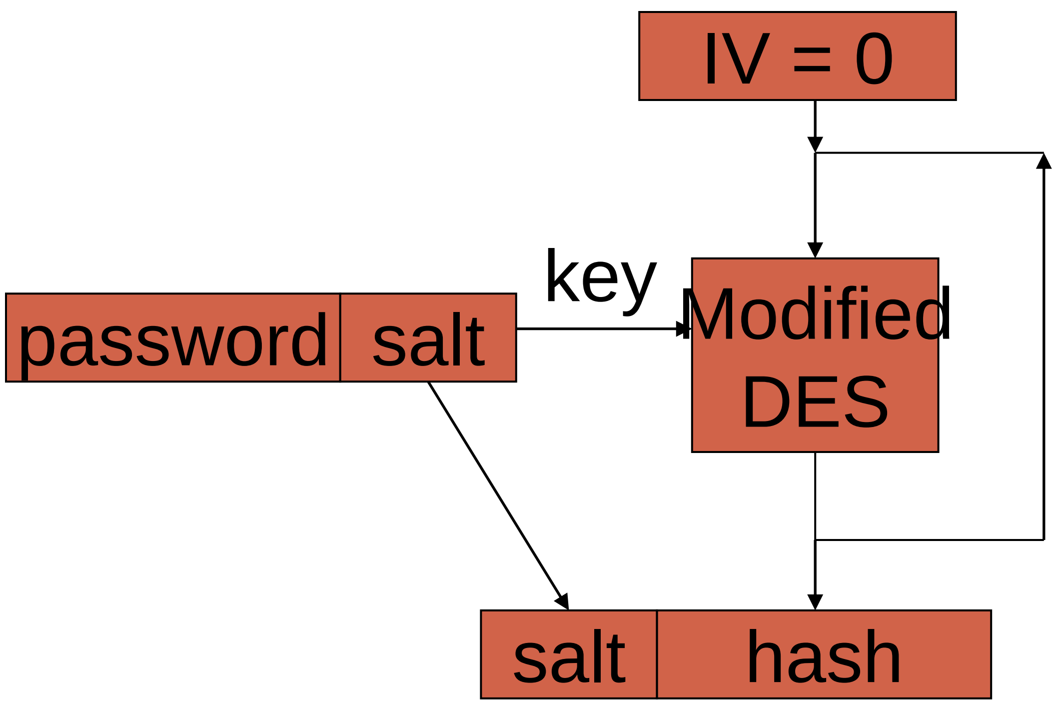
- Whirlpool uses a cipher called W, based on AES but with a 512 bit block and 512 bit key.



C_0 = fixed IV
$C_i = E(m_i, C_{i-1})$
$h(m) = C_k$

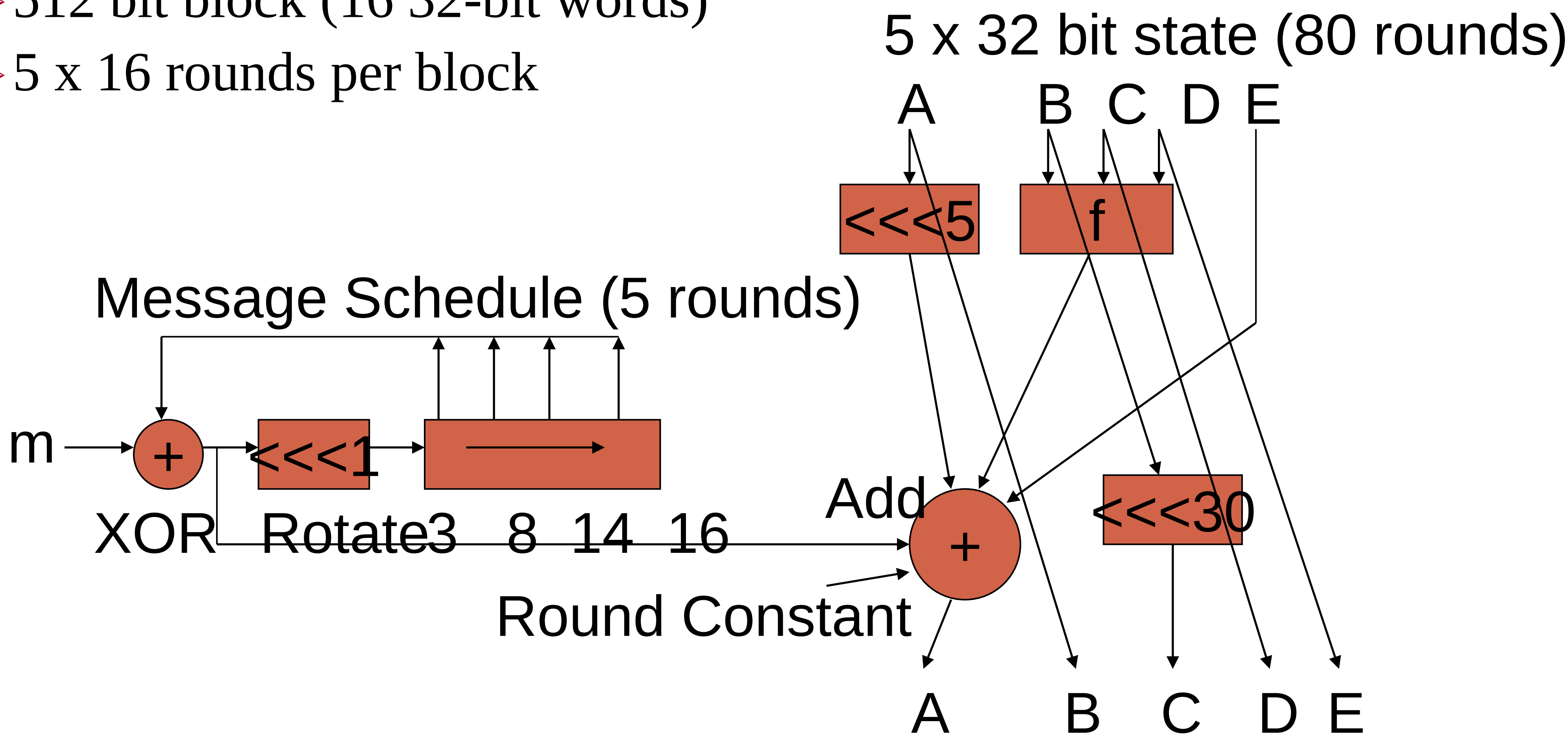
UNIX Password Hash

- Hash is stored in /etc/passwd (public) or /etc/shadow (readable by root)
- 8 byte ASCII password is used as 56-bit key to modified DES
- Iterated thousands of times to slow down brute force guessing
- 12 bit salt used to thwart table lookup and detection of reused passwords
- DES modified to thwart hardware acceleration
- Newer systems now use MD5 to overcome password length limit



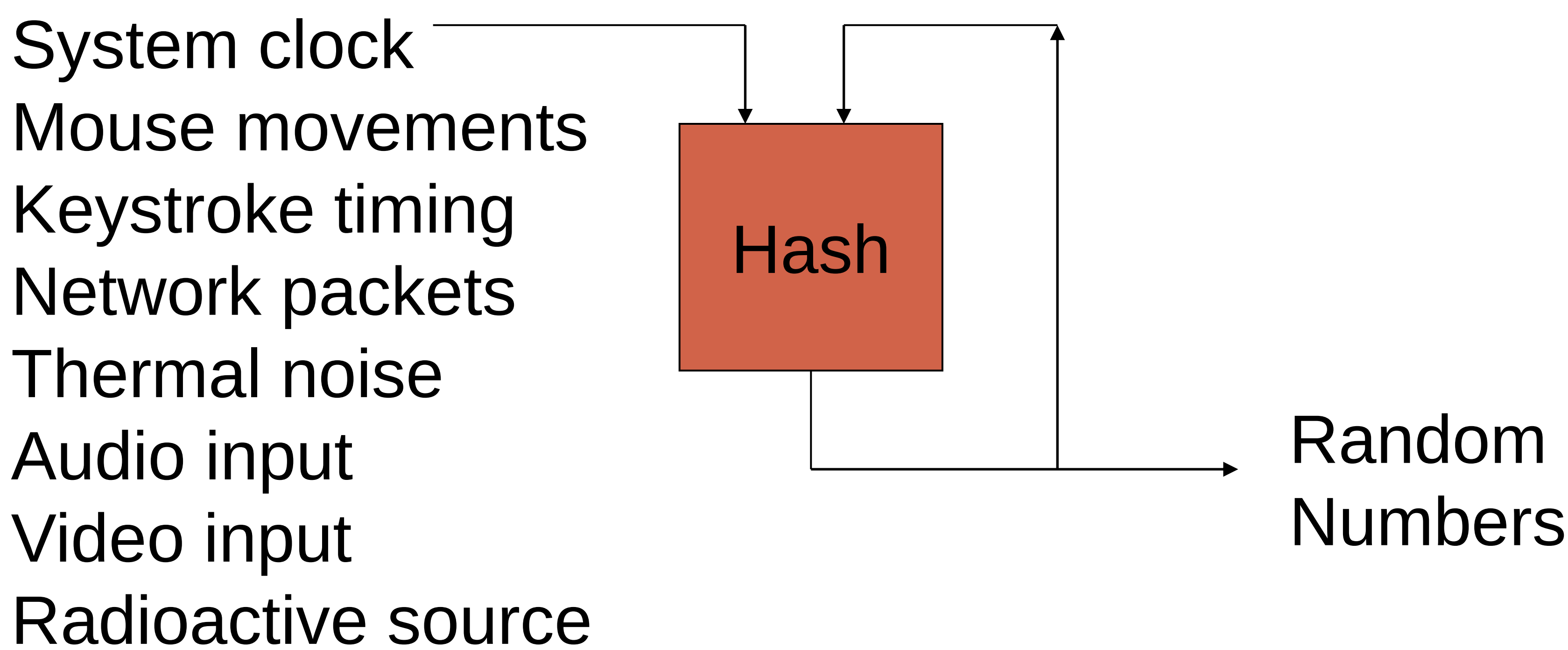
SHA-1 (RFC 3184)

- 160 bit hash
- 512 bit block (16 32-bit words)
- 5 x 16 rounds per block



Random Number Generation

- Random = not guessable by an attacker.
- Requires a hardware source of entropy.



Message Authentication Code (MAC)

- HMAC(K, m) =
  h(K xor 0x5c5c…|| h(K xor 0x3c3c… || m))
  - *h = SHA-1 or MD5*
  - *K = key*
  - *m = message*
- Can only be computed if you know K.
- FIPS Pub 198

Diffie-Hellman Key Exchange

- DH allows Alice and Bob to agree on a key over an insecure channel.
  - *Let p be a large prime number (2K-4K bits)*
  - *Let g be a generator of* Z_p^*
    - g is a generator iff for all $0 < x \neq y < p$, $g^x \neq g^y \pmod{p}$.
  - *Alice chooses random x,* $1 < x < p-1$, *and sends* $g^x \pmod{p}$ *to Bob.*
  - *Bob chooses random y,* $1 < y < p-1$, *and sends* $g^y \pmod{p}$ *to Alice.*
  - *Alice and Bob use* $K = (g^x)^y = (g^y)^x = g^{xy}$
  - *Eve cannot compute* g^{xy} *from p, g,* g^x *and* g^y.
    - Computing x from $g^x \pmod{p}$ (discrete logarithm problem) is believed (but not *proven*) to be hard.

DH Man in the Middle Attack

- Alice -> Eve: g^x (intercepts message to Bob)
- Eve -> Bob: g^v (pretends to be Alice)
- Bob -> Eve: g^y (intercepts message to Alice)
- Eve -> Alice: g^w (pretends to be Bob)
- Eve now knows Alice's key g^{xw} and Bob's key g^{yv}

RSA Public Key Cryptography

- Originally discovered by GCHQ in 1973 but kept secret.
- RSA = Rivest, Shamir, Adelman, published 1978.
- Patented in 1983, expired in 2000.
- Alice chooses:
  - *two random primes, p and q, 1K-2K bits each,*
  - $n = pq$,
  - $t = lcm(p-1, q-1)$,
  - *e and d, such that* $ed = 1 \pmod t$ *(usually e is a small odd number),*
  - *Alice's public key is (n,e) and private key is (p,q,t,d).*
- Bob encrypts: $C = P^e \pmod n$
- Alice decrypts: $P = C^d \pmod n$

Security of RSA

- Computing P from P^e (mod n) is believed to be hard (discrete logarithm).
- Computing d from e and n is believed to be hard (requires factoring n to find p, q).
- Neither problem has been *proven* to be hard.
- Numbers up to 663 bits have been factored.
- A theoretical attack exists using a quantum computer.
  - *Shor's algorithm solves both the discrete logarithm and factoring.*

RSA Considerations

- Small message/exponent attack
  - *If $m^e < n$, then m is easy to find.*
  - *m should be padded with random data.*
- Factoring
  - *If p and q have only small factors, then n is easy to factor.*
  - *If p is close to q then n is easy to factor.*

RSA Man in the Middle Attack

- Bob -> Eve: my public key is (n_b, e_b)
- Eve -> Alice: my public key is (n_e, e_e) (pretending to be Bob)

$$
\begin{tikzcd}
\text{Alice} \arrow[r, "P^{ee} \pmod{n_e}"] & \text{Eve} \arrow[r, "P^{eb} \pmod{n_b}"] & \text{Bob}
\end{tikzcd}
$$

Eve deciphers P and encrypts with n_b, e_b

ElGamal Signature Algorithm

- Key Generation
  - *p = a large prime (at least 1K bits)*
  - *g = a generator of Z_p^* (g^i mod p generates all values from 1 to p-1)*
  - *x = secret key, $1 < x < p-1$*
  - *$y = g^x \pmod p$, public key*
- To sign message m
  - *Choose random k, $0 < k < p-1$, $\gcd(k, p-1)=1$*
  - *$r = g^k \pmod p$*
  - *$s = (h(m) - xr)k^{-1} \pmod{p-1}$, $s > 0$, h = hash function*
  - *Signature is (r,s)*
- To verify
  - *$0 < r < p$, $0 < s < p-1$?*
  - *$g^{h(m)} = y^r r^s \pmod p$?*
- Forgery requires finding x (discrete log) or finding a hash collision.
- Reusing k allows an attacker to find x. p and g may be reused.

Digital Signature Algorithm (DSA)

- Avoids RSA patent
- Defined in FIPS 182-2
- ElGamal signature is twice size of p
- DSA reduces signature to 320 bits (mod $q < p$)
- Parameters:
  - *p = 1024 bit prime*
  - *q = 160 bit prime, $qz + 1 = p$ for some integer z*
  - *h = SHA-1*
- FIPS 182-3 proposes larger primes and hashes

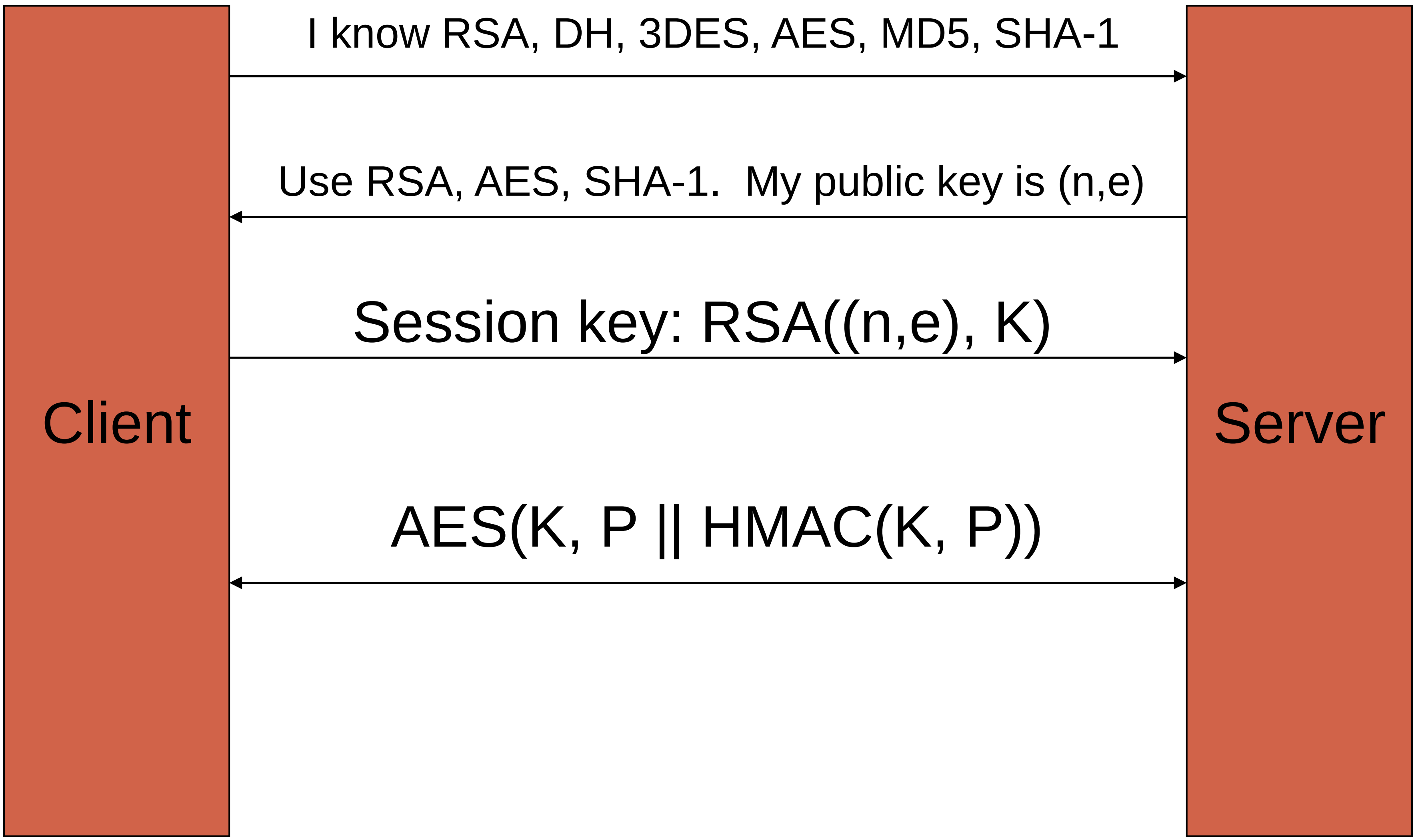
DSA

- Key Generation
  - g = generator in Z_p^* (choose h: $g = h^z > 1 \pmod p$)
  - x = randomly chosen secret key
  - $y = g^x \pmod p$
  - Public key is (p, q, g, y), private key is x
- Signing m:
  - Choose random secret k, $0 < k < q$
  - $r = (g^k \bmod p) \bmod q$, $r > 0$
  - $s = (h(m) + xr)k^{-1} \pmod q$, $s > 0$
- Verifying
  - $0 < r < q$, $0 < s < q$?
  - $u_1 = h(m)s^{-1} \pmod q$
  - $u_2 = rs^{-1} \pmod q$
  - $r = (g^{u1}y^{u2} \bmod p) \bmod q$?

Secure Sockets Layer (SSL)

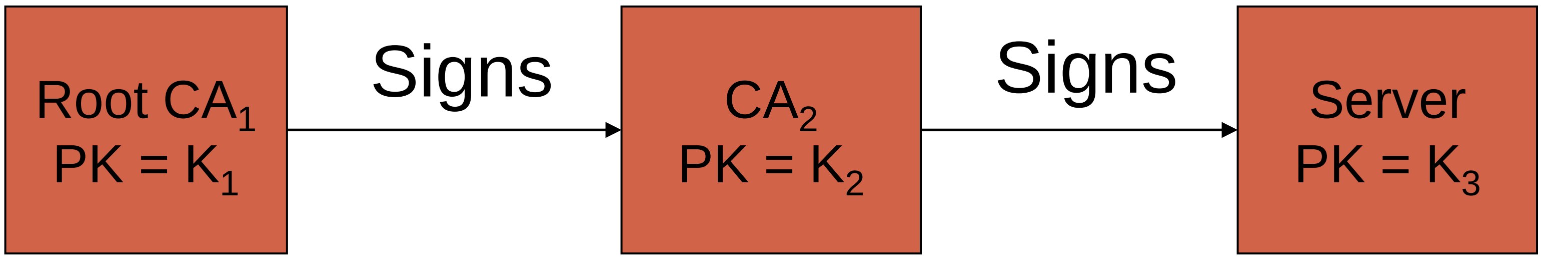
- https protocol (secure channel)
- Version 3.0 developed by Netscape in 1996
- Also known as TLS 1.0 (Transport Layer Security)
- Supports many algorithms
  - *Public Key: RSA, DH, DSA*
  - *Symmetric Key: RC2, RC4, IDEA, DES, 3DES, AES*
  - *Hashes: MD5, SHA*
- Public keys are signed by CA (Certificate Authority) using X.509 certificates.

SSL Example


I know RSA, DH, 3DES, AES, MD5, SHA-1
Use RSA, AES, SHA-1. My public key is (n,e)
Session key: RSA((n,e), K)
Client
Server
AES(K, P || HMAC(K, P))

X.509 Certificates

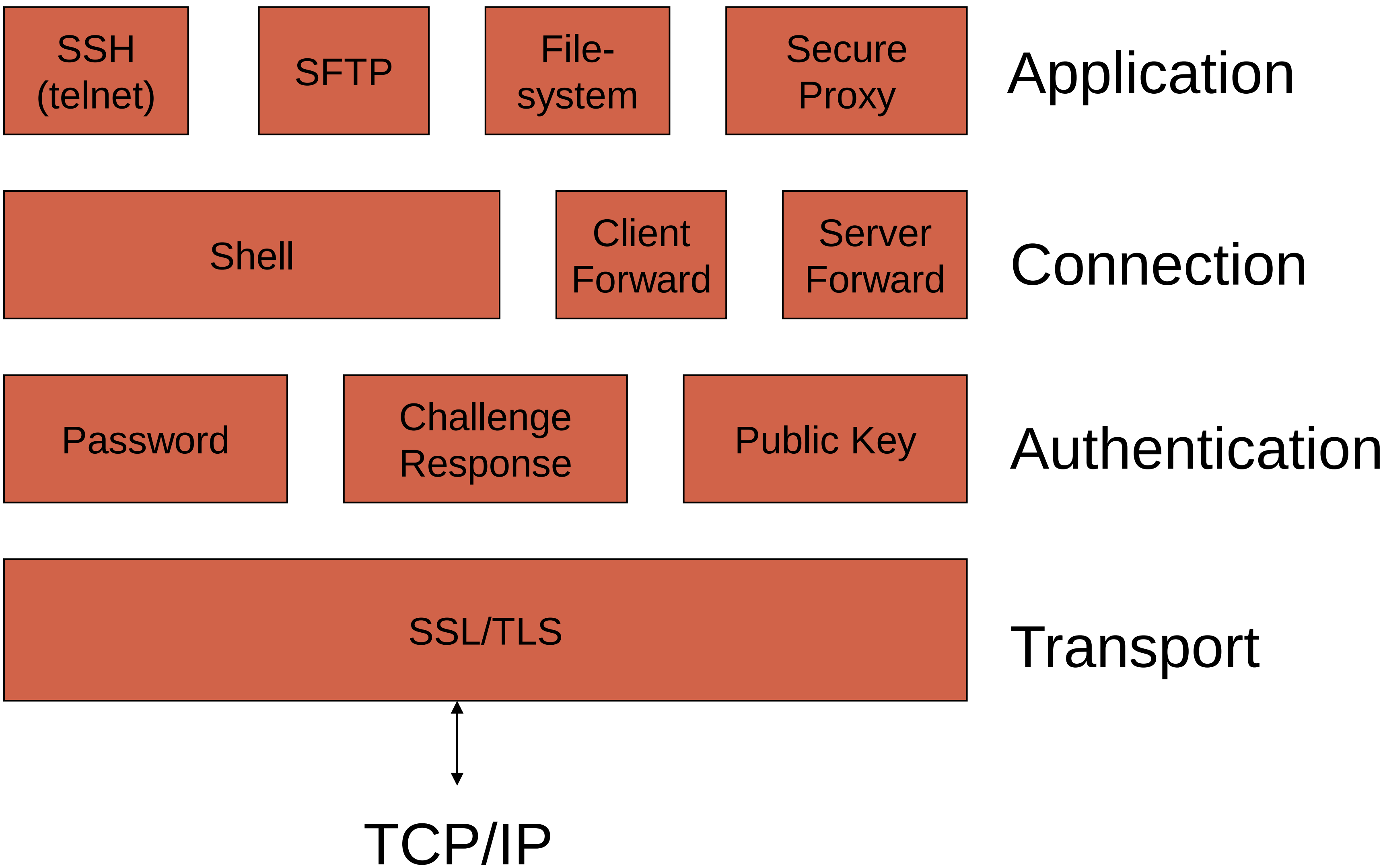
- Goal: prevent man in the middle attacks.
- Binds public keys to servers (or clients).
- Signed by a “trusted” certificate authority (CA).
- Chains to a root CA.



X.509 Weaknesses

- Not well understood by users (which CA's do you trust?)
- CA private key could be leaked.
- Certificates using MD5 can be forged.
  http://www.win.tue.nl/~bdeweger/CollidingCertificates/

SSH Layered Architecture



- Groups Z_p, Z_p^*
- Algorithms for Modular Arithmetic
  - *gcd*
  - *Extended Euclid (inverse mod p)*
  - *Chinese Remainder Theorem (CRT)*
  - *Exponentiation*
  - *Rabin-Miller prime testing*
- Fields $GF(p^n)$

Groups

- A set G and a binary operation + that is:
  - *Closed: If a and b are in G then a + b is in G.*
  - *Associative: (a + b) + c = a + (b + c).*
  - *An identity element 0: a + 0 = 0 + a = a.*
  - *Inverses: -a + a = a + -a = 0.*
- Examples:
  - *Integers under addition.*
  - *Reals except 0 under multiplication.*
  - *Right multiplication of nonsingular matrices.*

Modular Groups

- $Z_n = (\{0,1,\ldots,n-1\}, + \bmod n), n > 0$
  - *Additive group of order (size) n.*
  - *Identity element is 0.*
  - *Inverse of a is -a mod n.*
- $Z_p^* = (\{1,2,\ldots,p-1\}, \times \bmod p)$, p prime.
  - *Multiplicative group of order p – 1.*
  - *Identity element is 1.*
  - *Inverses can be found using extended Euclid's algorithm.*

Euclid's GCD Algorithm

- Greatest Common Divisor of a, b ≥ 0
- gcd(a, b) =
  - *while (a $\neq$ 0) do*
    - (a, b) = (b mod a, a)
  - *return b*
- lcm(a, b) = ab / gcd(ab)
- If gcd(a, b) = 1 then we say a and b are *relatively prime*.

Extended Euclid's Algorithm

- Finds a^{-1} in Z_p^*
- ExtendedGCD(a, p) =
  - *u := 1, v := 0*
  - *while (a ≠ 0) do*
    - q := ⌊p/a⌋
    - (a, p) := (p – qa, a)
    - (u, v) := (v – qu, u)
  - *return* $a^{-1} = v$

Chinese Remainder Theorem (CRT)

- CRT: (x mod p, x mod q) uniquely represents x in Z_{pq}, p, q prime.
- Given a = x mod p, b = x mod q, Garner's formula finds x:
  - $x = (((a - b)(q^{-1} \bmod p)) \bmod p)q + b) \bmod q$
- Can be extended to any number of prime modulus.

Efficient Exponentiation

- Compute by repeated squaring
- a^x mod n:
  - *if x = 0 return 1*
  - *else if x = 1 return a*
  - *else if x is even return $a^{x/2}\, a^{x/2}$ (mod n)*
  - *else x is odd, return $a\, a^{x-1}$ (mod n)*

Fermat's Little Theorem

- If p is prime, a > 0, then $a^{p-1} = 1 \pmod{p}$.

Subgroups

- A subgroup is a subset of a group that is also a group.
- The order of a subgroup of Z_p^* divides p – 1.
- Example: ({1,2,4}, x mod 7) is a subgroup of Z_7^*.
  - *This subgroup has order 3, which divides 7 – 1.*

Generators

- g is a generator of G if powers of g generate all elements of G.
- For all g in Z_p*, g generates either Z_p^* or a subgroup.
- Therefore g is a generator of Z_p^* iff for all factors $f < p - 1$ of $p - 1$, $g^f \neq 1 \pmod{p}$.

Fermat Test for Primes

- Testing by factoring is not possible for large primes.
- Test is probabilistic.
  - *Can only prove a number is composite.*
  - *Error can be made arbitrarily small.*
- Uses Fermat's little theorem.
  - *If* $a^{n-1} \neq 1 \pmod{n}$ *then n is composite.*
  - *If n is composite, then* $a^{n-1} = 1 \pmod{n}$ *for at most ¼ of a,* $0 < a < n$.
  - *If* $a^{n-1} = 1 \pmod{n}$ *for many a, then n is probably prime.*

Rabin-Miller Test for Primes

- Optimizes Fermat test to reduce number of modular multiplications:
- Write n as $2^t s + 1$, s odd
- Repeat 64 times
  - *Pick random a, $1 < a < n$*
  - *$v = a^s \bmod n$ (slow step)*
  - *While $t > 0$ and $v \neq 1$ and $v \neq -1$ do*
    - $v = v^2 \bmod n$
    - $t := t - 1$
  - *If ($v \neq 1$ and $v \neq -1$) or ($t = 0$ and $v \neq 1$) then return n is composite*
- Return n is prime with probability $1 - 2^{-128}$

Fields

- A field is a set G and *two* operators, + and x.
- (G,+) is a group with identity 0.
- (G\0, x) is a group with identity 1.
- Distributive: a(b + c) = ab + ac.
- Examples
  - *Real numbers over + and x.*
  - *Polynomials over* $GF(p^n)$

Galois Fields

- $GF(p^n)$, p prime
- Set is $\{0,1,\ldots,p-1\}^n$, vector of n polynomial coefficients
- + is polynomial addition mod p.
- x is polynomial multiplication mod p mod an irreducible polynomial.
  - *A polynomial is irreducible if it has no factors but 1 and itself.*

GF(2^8) (from AES S-boxes)

- Elements are bytes.
  - *e.g.* $0x63 = 01100011 = x^6 + x^5 + x + 1$.
- Addition is mod 2 (xor).
- Multiplication is reduced over $x_8 + x_4 + x_3 + x + 1$.
  - *Multiply by shift and xor to 15 bits.*
  - *xor with shifted reduction polynomial 100011011 to cancel high bits.*
- AES uses GF(2^8) to resist certain differential attacks.

Summary

- Cryptography is hard, why?
  - *Security can not be proven.*
  - *Even expertly designed systems have weaknesses.*
  - *Designing your own encryption algorithm would be foolish.*
- Cryptography is not the best answer. Why?
  - *Most attacks do not involve breaking encryption.*
- Prevent, Detect, Recover?
  - *Cryptography is only for prevention.*

Further Reading

- Practical Cryptography, Ferguron & Schneier
  - *A practical approach to building secure systems.*
- Cryptography, Theory and Practice, Stinson
  - *Mathematics of cryptography and cryptanalysis.*
- Handbook of Applied Cryptography
  - *Free online reference, very theoretical.*
- Wikipedia
- sci.crypt

Questions?