UMSSIA

LECTURE V: SECRETS OF THE CRYPT...
The goal of cryptography is to provide a “secure channel” between Alice and Bob.

A secure channel:
1. Leaks NO information about its contents
2. Delivers only messages from Alice and Bob
3. Delivers messages in order or not at all
The goal of cryptography is to provide a “secure channel” between Alice and Bob. The channel should remain secure even though Eve can inspect, modify or drop any message, and control all but one bit of the channel contents.
There exist some physically “hard to break” channels.

Symmetric cryptography uses a temporary or slow secure channel to establish a secret key.

Knowing the key is what lets Alice and Bob keep secrets from Eve.
CAESAR’S CIPHER

To write a secret message,

1) Write down each plaintext letter

ATTACK AT DAWN

2) Count ahead three letters from each plaintext letter to get a ciphertext letter.

ATTACK AT DAWN

B U U B D L B U E B X O
C V V C E M C V F C Y P
D W W D F N D W G D Z Q
SUBSTITUTION

How many possible keys?

26 £ 25 £ 24 £ ... = 4.03 £ 10^{26}
BREAKING SUBSTITUTION

MOST COMMON ENGLISH LETTERS: ETAOINSHR...

FREQUENCY ANALYSIS: GUESS MOST COMMON LETTER = E, T, A, O, I, N, S, H, R...
POLYALPHABETIC ROTATION

... uses a different rotation for each letter.

Plaintext: ATTACKATDAWN
Key: CYA
Ciphertext: CRTCAKCRDCUN

How would you break polyalphabetic rotation?

“Modern” variant: the Vigenere cipher uses XOR.

Special case: The One Time Pad has |key| = |msg|

Any plaintext can encrypt to any ciphertext!
STREAM CIPHERS

... are computerized approximations to the OTP.

A stream cipher combines a short key and IV to make a long, pseudorandom keystream.

If no one can tell the stream apart from random without the key, Eve can’t read the ciphertext.

Key → Stream Generator → Keystream

IV → Stream Generator

Plaintext

Keystream

0xA6C2777014777EE9813

0xDEADBEEF45A1E218400

= 0x786FC99F51D69CF1C13
**EXAMPLES**

**RC4** is a popular stream cipher because its code is short.

RC4 has distinguishing attacks for 64MB files.

There are many ways to use RC4 insecurely, e.g. WEP.

LFSRs are popular in hardware crypto, due to low gate count.

They are usually trivial to break.

**NESSIE: No approved stream ciphers**

(Figures from Wikipedia)
... take as input a key and a block of bits. The output is a random-looking block of bits.

With the key, it is easy to invert the cipher.

Without guessing the key, it is impossible to tell the inputs and outputs apart from random bits.
ENCRYPTION MODES

... build encryption schemes for any input length out of block ciphers for fixed input lengths.

Example: Electronic Codebook Mode

ECB-Encrypt (Block[] P, Block[] C) :
for(i = 0; i < P.len; i++)  C[i] = Encrypt(P[i]);

ECB-Decrypt (Block[] P, Block[] C) :
for(i = 0; i < C.len; i++)  P[i] = Decrypt(C[i]);

If you see a product using ECB, run - don’t walk - away!
SECURE MODES

**CBC-Encrypt** (Block[] P, Block[] C):

C[0] = RandomBlock();
for(i = 0; i < P.len; i++)
    C[i+1] = Encrypt(P[i] XOR C[i]);

**CBC-Decrypt** (Block[] P, Block[] C):
for(i = 1; i < C.len; i++)
    P[i-1] = C[i-1] XOR Decrypt(C[i]);

**CTR-Encrypt** (Block[] P, Block[] C):

C[0] = RandomBlock();
for(i = 0; i < P.len; i++)
    C[i+1] = Encrypt(C[0]+i) XOR P[i];

**CTR-Decrypt** (Block[] P, Block[] C):
for(i = 1; i < C.len; i++)
    P[i] = Encrypt(C[0]+i) XOR C[i];
The **Advanced Encryption Standard**, or AES, was chosen as the result of a five-year competition.

AES has security proofs against most known attacks on block ciphers.

AES is fast in hardware, fast in software, and friendly to embedded processors.
The goal of cryptography is to provide a “secure channel” between Alice and Bob.

A secure channel:
1. Leaks NO information about its contents
2. **Delivers only messages from Alice and Bob**
3. Delivers messages in order or not at all
INTEGRITY

... Is the property that a message has not been modified by anyone but the sender.

A channel provides integrity if Eve cannot make new, valid messages even after seeing Alice and Bob send many of them.
HASH FUNCTIONS

... map long input strings to short output strings.

\[ H(x, y, z) = x + y + z \]

\[ H(x) = x \mod 65537 \]

Hash functions may have a key, e.g.:

\[ H_{a, b, p}(x) = ax + b \mod p \]

When \((a, b, p)\) are used only once, \(H_{a, b, p}(x)\) can be appended to \(x\) to prevent modification.
... Are fixed hash functions with outputs that “look random” to an adversary.

A cryptographic hash function should resist the following attacks:

1. “inversion” or preimage: given $y$, find any $x$ so that $H(x) = y$.

2. **Targeted** collisions: given $x$, $H(x)$, find $x' \neq x$ where $H(x') = H(x)$.

3. **Free** collisions: find $x' \neq x$ where $H(x) = H(x')$. 

CRYPTO HASH FUNCTIONS
**HASH FUNCTION STRENGTH**

For **any** function $H$ with $k$-bit outputs, how long (at most) does it take to find:

1. Preimages?  
   - $2^k$ hashes.

2. Targeted Collisions?  
   - $2^k$ hashes.

3. Free Collisions?  
   - $2^{k/2}$ hashes.

Because of this **birthday attack** a $k$-bit hash function has at most “$k/2$ bits of security” against free collisions.
HASH FUNCTION WEAKNESS

We know some good hash functions (SHA-256, Whirlpool) and there is work on others.

... but sending \((m, H(m))\) does not create a channel with integrity against adversaries.

For that, we need something with a key: a Message Authentication Code, or MAC.
A MAC takes a key and message and outputs a random-looking “tag” of fixed length.

The verification function checks that a tag matches a message and key.

It should be hard to find a new correct tag without knowing the key.

Even if the adversary gets to see many messages and tags.
CBC-MAC builds a MAC out of a block cipher in CBC mode. It is only secure if all messages use the same number of blocks.

HMAC builds a MAC out of a crypto hash function. 

$$\text{HMAC}_K(m) = H(0x5c \oplus K, H(0x36 \oplus K, m))$$

Breaking HMAC involves finding collisions in the Hash function.
SECURE CHANNELS

A secure channel:
1. Leaks NO information about its contents
2. Delivers only messages from Alice and Bob
3. Delivers messages in order or not at all

To send $M_i$, do $C = \text{Encrypt}(M)$; $T = \text{Tag}(C, i)$. Send $(i, C, T)$. 
WHY INTEGRITY?

Without integrity protection, chosen ciphertext attacks become a threat.

Alice sends $C = \text{RC4-Encrypt}_K(8\text{CharPassword})$.

Bob checks that `strlen(password)` is exactly 8.

If not, replies with warning: "8 letters only!"
Else, does expensive password check.

Eve sends $C\oplus 0x0100000000000000$ – quick response means first char = $0x01$! Else try $0x02$, $0x03$… $0xFF$.

Integrity protects such attacks – Eve can’t make new ciphertexts to try!
CHEAP INTEGRITY

One objection to MACing ciphertexts is that the cost of encryption doubles.

To circumvent this limitation, there exist authenticated encryption modes that encrypt N blocks with less than 2N block cipher calls.

The new NIST standard for such modes is Galois Counter Mode (GCM). GCM is fast and free.
Given a shared secret we can build a channel that:
1. Leaks NO information about its contents
2. Delivers only messages from Alice and Bob
3. Delivers messages in order or not at all

WHAT IF NO SHARED SECRET IS AVAILABLE?
PUBLIC-KEY CRYPTO

Bob publishes **public** (encryption) **key** PK
keeps secret **secret** (decryption) **key** SK

Alice
Get Bob’s Public Key
compute \( c = E(PK, m) \)

Eve
Even though Eve knows PK and all but one bit about \( m \), she should not learn the contents of \( p \).

Bob
Compute \( p = D(SK, c) \)
PUBLIC KEY SCHEMES

• “Vanilla” or “Textbook” RSA:
  - Keys: choose p, q prime.
    N = pq
    e £ d mod (p-1)(q-1) = 1
    PK = (N, e); SK = (N,d)
  - Encrypt: c = m^e mod N
  - Decrypt: p = c^d mod N

• Vanilla RSA is **insecure**!
El Gamal: Let \( p, q \) be prime and \( p = 2q+1 \)
- Keys: pick \( x \mod q \).
  \( SK = x; PK = y = g^x \mod p \)
- Encrypt(\( m < q \)): pick \( r \mod q \).
  if \( m^q \mod p = -1 \), set \( m = p-m \).
  let \( z = y^r, v=g^r, w = mz \mod p \)
  \( c = (v, w) \)
- Decrypt(\( c \)): \( z = v^x \quad ( = (g^r)^x = g^{rx} = y^r ) \)
  \( m = w/z \mod p \)
  if \( m > q \), set \( m = p-m \).

El Gamal is secure if Eve can’t modify or inject packets.
HYBRID ENCRYPTION

- AES encryption is fast. (~1Gbps in Software; ~10Gbps in Hardware)
- RSA and El Gamal are slow.
  - Software: ~100Kbps (RSA encrypt ~1Mbps)
  - Hardware: ~1Mbps (RSA encrypt ~10Mbps)
  - Elliptic curves: £ 10
- Hybrid Encryption schemes get public key properties with private key speed:
  - Pick random K
  - encrypt K with PK scheme,
  - Encrypt m with symmetric key K
**SECURE HYBRID RSA**

**RSA-KEM:**
- **Keys:** choose \( p, q \) prime; \( N = pq \)
  
  \( e \leq d \mod (p-1)(q-1) = 1 \)

  \( PK = (N, e); SK = (N,d) \)

- **Encrypt:** pick \( r \mod N \)
  
  \( K = H(r) \)

  \( c_1 = r^e \mod N \)

  \( c_2 = E(K, m) \)

- **Decrypt:** \( K = H(c_1^d \mod N) \)

  \( p = D(K, c_2) \)

- \((E, D)\) must be integrity-aware secure symmetric scheme

- **Also:** RSA-OAEP, RSA-PKCS2
SECURE HYBRID ELGAMAL

• Keys: pick $x \mod q$
  $SK = x; \ \ PK = y = g^x \mod p$

• Encrypt: pick $r \mod q$
  $z = y^r \mod p; \ \ c_1 = g^r \mod p$
  $K1 = HMAC(z, 1)$
  $c_2 = E(K1, m)$
  $c_3 = HMAC(z, 0||c_1||c_2)$

• Also ECDHIES, Cramer-Shoup
IDENTITY-BASED ENCRYPTION

TRENT publishes MASTER key MPK and keeps secret master secret key MSK

\[ c = E_{MPK}("Bob", m) \]

\[ p = D(BSK, c) \]
Diffie-Hellman Key Exchange

Discrete Logarithm Problem:
Hard to find $a$ given $A$, $p$

$A = g^a \mod p$
$B = g^b \mod p$

$K = B^a \mod p$
$K = A^b \mod p$

Shared key: $K = g^{ab} \mod p$
MAN IN THE MIDDLE

$K = B'^a \mod p$

$A = g^a$

$B' = g^{b'}$

$A' = g^{a'}$

$B = g^b$

$K' = A'^b \mod p$

$K' \neq K$ but Alice & Bob think they are!
STATIC DIFFIE HELLMAN

Official Public Keys
Alice...................A
Bob...................B
Carol.................C
Dave...................D

Shared key: $K_{ab} = g^{ab} \mod p$

SK = a
$K_{ab} = B^a$

SK = b
$K_{ab} = A^b$
ULTIMATE DIFFIE HELLMAN

- Alice and Bob have public keys $A = g^a, B = g^b$
- Want to exchange “session” key $K$ so that:
  - $K$ will be secret even if $a$ and $b$ leak.
  - Prevent Man in the middle: Only Alice and Bob know $K$; Mallory can’t force $K \neq K’$
  - If Mallory learns $K$, can’t figure out $a$ or $b$.
  - If Mallory learns $a$, he can’t fool Alice
- The “state of the art” protocol is HMQV
  
- But how do we know that $A$ and $B$ are really Alice and Bob’s public keys?
DIGITAL SIGNATURES

Alice publishes **verification key** VK
keeps secret **signing key** SK

Alice
Computes signature on $m$
$s = S(SK,m)$

Anyone
Verify: what’s $VK_{Alice}$?
Verify: is $s$ valid on $m$?
$V(VK,m,s) = YES/NO$

Signatures provide **nonrepudiation**: only Alice (or whoever knows SK) could have said $m$. 
IMPORTANT

Digital signatures are NOT "encryption with the private key"
DSA: prime $p = 2q+1$
- **Keys:** pick $x \mod p$.
  
  $SK = x \mod p$; $VK = y = g^x \mod p$
- **Sign:** pick $k \mod q$.
  
  let $r = g^k$, $w = (h(m)+xr)/k \mod q$
  
  $s = r,w$
- **Verify:** $g^{h(m)}y^r \mod p = r^w \mod p$

**Also:** ECDSA, KCDSA, Schnorr, BLS…
DIGITAL SIGNATURE SCHEMES

- **RSA-FDH:** let $H$ output $n-1$ bits
  - Keys: choose $p$, $q$ prime.
    
    $2^n > N = pq > 2^{n-1}$
    
    $v \equiv s \pmod{(p-1)(q-1)} = 1$
    
    $VK = (N, v)$; $SK = (N, s)$
  - Sign: $S = H(m)^s \pmod{N}$
  - Verify: $S^v \pmod{N} = H(m)$?

- “Vanilla RSA” and RSA-PKCS#1 are not secure.

- Also: RSA-FDH, RSA-PSS, RSA-KW
CERTIFICATES

- Digitally signed:
  - Encryption PK
  - Signature VK
  - "Identity"
  - (opt) Signer Name
  - (opt) Expiry Date
  - Cert. Type
- How do we verify the cert?

Oct 5, 2005

I hereby certify that the hash of the public key of Alice@ms.net Is:

Signed,
MacroSquash Corp.
Oct 5, 2005

I hereby certify that the cert-signing key of

MacroSquash Corp

Is:

0xcbb7ccaddbe7692cb7a4290244479c3c

Signed,

Squash Group Intl

Oct 5, 2005

I hereby certify that the hash of the public key of

Alice@ms.net

Is:

0x15fd06ee754b36697c524c21fae25d1f

Signed,

MacroSquash Corp.
Oct 5, 2005

I hereby certify that the cert-signing key of Squash Group Intl
Is:

0x33666c857616c7f5afdbfc525b9d68e8

Signed, Squash Group Intl

Oct 5, 2005

I hereby certify that the cert-signing key of MacroSquash Corp
Is:

0xcbb7ccadd8e7692cb7a4290244479c3c

Signed, Squash Group Intl

Oct 5, 2005

I hereby certify that the hash of the public key of Alice@ms.net
Is:

0x15f06ee754b36697c524c21f6e25d1f

Signed, MacroSquash Corp.
**PUBLIC KEY INFRASTRUCTURE**

**PKI** a generic term for any method of binding PKs and VKs to names.

**PKI** is also used as a reference to a specific, hierarchical approach:

- Public Key Certificates are signed by Certification Authorities (CAs)
- CAs themselves have certificates that specify their authority (eg *.cs.umn.edu) and are signed by “Parent CAs” (eg *.umn.edu)
- There are a few “Root CAs” (self-certified), whose signing keys are “well-known” and ship with OSes, are published in newspapers, etc.
- CAs are trusted. Are they trustworthy?
PGP AND THE WEB OF TRUST

• PGP is a program and standard for email encryption with no CAs.
• Each user has a keyring of trusted keys
  – Some are personally verified (e.g. face to face)
  – Some are pre-trusted keys
• Public Keys are published on PGP key servers
• Users sign each other’s keys, including a level of “confidence” in the correctness of the binding.
• The decision whether to accept a PK: Name binding is local. For example, PGP recommends counting the number of disjoint key sig. paths from your keyring to PK.
Decisions about public key bindings can be made by any local policy: eg length of paths, number of paths, minimum cert level along paths...

Keyservers help by doing path computations.

E.g. Find paths from Wietse Venema to Phil Zimmerman: 7 paths; max length 4, min length 2.
REVOCATION

• One big problem for PKI is what to do when a Certificate is wrong:
  – Secret Keys may be compromised, lost or destroyed
  – Bindings may become invalid (employees leave companies, get new positions, etc.)

• One solution is to **revoke** or invalidate the certificate.

• The most common mechanism is via Certificate Revocation Lists:
  – Every Cert includes a “revocation server”
  – Lists of revoked keys are published by Root CAs
  – PGP pushes revocations to Key Servers.
  – Validating certificates requires contacting revocation servers.