Symmetric Key Cryptography
Cryptography Intro

- Provides confidentiality
- Securely exchange information in an insecure environment
- Process:

```
Plaintext → Encryption → Ciphertext → Decryption → Plaintext
```

Unreadable, “random” information
Cryptography Intro

- Encryption and Decryption are specified algorithms
- Only valid users know how to perform decryption
- Two approaches:
  - Keep encryption and decryption algorithms secret: “security through obscurity”
  - Add another variable, called a key that must be known to encrypt and decrypt a message
- Symmetric-key cryptography → same key used for both encryption and decryption
Cryptography Intro

• Symmetric-key cryptography requires the recipient of an encrypted message knows the key
  – Can’t send key in a separate message because an adversary could intercept it and know the key
  – Assume key is known
  – Will address key distribution protocols in lecture 3
Classical Cryptography

- Caeser Cipher
- Julius Caeser was the first to use this scheme
- Substitution cipher: replace individual characters with different characters
- Caeser Cipher shifted characters by 3

Plaintext: I HAVE A SECRET
Ciphertext: L KDYH D VHFUHW
Classical Cryptography

- Rotation ciphers: special case of substitution cipher
- Map letters to numbers
  - A=0, B=1, C=2, ..., Z=25
- Rotate by a constant $k$ for each character
  - $y = E(x) = x + k \pmod{26}$
- For Caesar Cipher, $k=3$
- Decryption:
  - $D(y) = y - k \pmod{26}$
  - $D(E(x)) = x + k - k \pmod{26} = x$
- Special Case: $k=13$
  - Known as ROT-13
  - $E() = D()$
Classical Cryptography

- Also known as Monoalphabetic ciphers
  - Same encryption algorithm used for each character
- Vulnerable to frequency analysis
  - Problem
    - Characters and words are not random
    - Context defines which are used when
    - Characters can be considered random variables with a certain unique probability distribution (not independent)
  - Attack
    - Compute probability distribution of characters in the encrypted message
    - Compare with distribution in target language
    - Map characters back
Classical Cryptography

• Frequency Analysis
  – Can look at individual letters or combinations of letters
  – The more ciphertext available, the better the result
Polyalphabetic Ciphers

• Change the substitution pattern on a per-character basis

• Example: Vigenère cipher
  – Caeser cipher on each character, but change “k” value
  – Word or phrase was used to determine “k” values, repeated over and over

<table>
<thead>
<tr>
<th>A B C D E F G H I J K L M N O P Q R S T U V W X Y Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>A B C D E F G H I J K L M N O P Q R S T U V W X Y Z</td>
</tr>
<tr>
<td>B C D E F G H I J K L M N O P Q R S T U V W X Y Z A</td>
</tr>
<tr>
<td>C D E F G H I J K L M N O P Q R S T U V W X Y Z A B</td>
</tr>
<tr>
<td>D E F G H I J K L M N O P Q R S T U V W X Y Z A B C</td>
</tr>
<tr>
<td>E F G H I J K L M N O P Q R S T U V W X Y Z A B C D</td>
</tr>
<tr>
<td>F G H I J K L M N O P Q R S T U V W X Y Z A B C D E</td>
</tr>
<tr>
<td>G H I J K L M N O P Q R S T U V W X Y Z A B C D E F</td>
</tr>
<tr>
<td>H I J K L M N O P Q R S T U V W X Y Z A B C D E F G</td>
</tr>
<tr>
<td>I J K L M N O P Q R S T U V W X Y Z A B C D E F G H</td>
</tr>
<tr>
<td>J K L M N O P Q R S T U V W X Y Z A B C D E F G H I</td>
</tr>
<tr>
<td>K L M N O P Q R S T U V W X Y Z A B C D E F G H I J</td>
</tr>
<tr>
<td>L M N O P Q R S T U V W X Y Z A B C D E F G H I J K</td>
</tr>
<tr>
<td>M N O P Q R S T U V W X Y Z A B C D E F G H I J K L</td>
</tr>
<tr>
<td>N O P Q R S T U V W X Y Z A B C D E F G H I J K L M</td>
</tr>
<tr>
<td>O P Q R S T U V W X Y Z A B C D E F G H I J K L M N</td>
</tr>
<tr>
<td>P Q R S T U V W X Y Z A B C D E F G H I J K L M N O</td>
</tr>
<tr>
<td>Q R S T U V W X Y Z A B C D E F G H I J K L M N O P</td>
</tr>
<tr>
<td>R S T U V W X Y Z A B C D E F G H I J K L M N O P Q</td>
</tr>
<tr>
<td>T U V W X Y Z A B C D E F G H I J K L M N O P Q R S</td>
</tr>
<tr>
<td>U V W X Y Z A B C D E F G H I J K L M N O P Q R S T</td>
</tr>
<tr>
<td>V W X Y Z A B C D E F G H I J K L M N O P Q R S T U</td>
</tr>
<tr>
<td>W X Y Z A B C D E F G H I J K L M N O P Q R S T U V</td>
</tr>
<tr>
<td>X Y Z A B C D E F G H I J K L M N O P Q R S T U V W</td>
</tr>
<tr>
<td>Z A B C D E F G H I J K L M N O P Q R S T U V W X Y</td>
</tr>
</tbody>
</table>
Polyalphabetic Ciphers

• Can’t apply frequency analysis directly because the translations change every character
• Fatal flaw: repeating key
• Guess key length, treat ciphertext as interwoven Caeser ciphers
• Compute probability distribution
• Rotate until it matches
Polyalphabetic Ciphers

• $V(0..25) =$ vector of target language probabilities

• For $L=1$ to maxLength
  – Partition message into $L$ sub-messages, $M(1) \ldots M(L)$
  – For $i=1$ to $L$
    • $H =$ Histogram($M(i)$)
    • Find value $k(L, i)$ that minimizes $e(L, i) = \text{sum}(|\text{rotate}(H, k(L, i)) - V|)$
  – $\text{error}(L) = \text{sum}(e(L,i))/L$

• Key is the values $k(L, .)$ that minimizes $\text{error}(L)$
Polyalphabetic Ciphers

- Enigma Machine: Famous Polyalphabetic cipher
- Per-character substitution cipher
- Each key press rotors turned to change the substitution pattern
- Result: 16,900 key strokes before the pattern repeated
- Flaws:
  - Characters never mapped to themselves
  - E() = D()
  - Operator errors
  - Known plaintext
- Given operator errors, analysts could determine key
Perfect Secrecy

- In 1948, Claude Shannon invented the basis of Information Theory in his publication “A Mathematical Theory of Communication”

- Let $P$=plaintext, $C$=ciphertext, $K$=key

- To be secure: $Pr(P|C) = Pr(P)$

- Bayes’ Theorem says: $Pr(P|C) = \frac{Pr(C|P)Pr(P)}{Pr(C)}$

- To be secure: $Pr(C|P) = Pr(C)$

- $Pr(C|P) = Pr(K)$, thus we must have $Pr(C) = Pr(K)$

- Thus the key must be just as random as the ciphertext
Perfect Secrecy

- One-Time Pad
- Non-repeating set of keys on pieces of paper glued together
- Sender and receiver need identical pads
- Sender uses Vigenère cipher
- Row used is based on key value from pad
- Never reuse key values
- Provably secure, but pads difficult to distribute and maintain
Modern Cryptography

• Modern symmetric-key cryptography can be characterized as:
  – Block Ciphers
  – Stream Ciphers
  – Message Integrity Codes
Block Ciphers

• Typically: \( N \) bits of plaintext and \( N \) bits of key yield \( N \) bits of ciphertext

• Assume \( K \) and \( C \) are uniformly random, \( P \) can have any distribution

• Called block cipher because it operates on block of \( N \) bits of data at a time
DES

- Digital Encryption Standard, adopted 1976
- In early 1970s, USG recognized the need for a standardized block cipher
  - Provide high level of security
  - Easy to understand and implement
  - Releasable and exportable
  - Validated by the public
- 1972: call for proposals from National Bureau of Standards (now NIST) received few responses
- 1974: second call
- IBM cipher “Lucifer” was accepted
- Eventually accepted by ISO as an international standard for encryption
DES

- 16 rounds of substitution and transposition
  - Substitution abstractly performed by “S-Box”
  - Permutation permutes bit sequences using “P-Box”
  - Product cipher alternating application of S-Box and P-Box

- Uses ideas of confusion and diffusion
  - Confusion: substitution
  - Diffusion: spread effects of change in the plaintext throughout ciphertext

- $E() = D()$
DES

• 64-bit plaintext, 56-bit key, 64-bit ciphertext
• Key really 64 bits long, with 8 bits parity
• Key length concerns
  – 56 bits long, try all keys until successful (C,P known)
  – Estimate in 1977 guessed that for $50M you could crack a key one day
  – Moore’s law: computers keep getting faster
  – In 1997 a distributed network of 3500 computers cracked the a DES key in 4 months
  – In 1998 for $130K you could crack a key in < 5 days
  – 2008: 10 years later, computers are 60x faster
    • Network of 3500 could crack in 2 days
**DES**

- 56-bit key too short, how can we make it more secure?
- Enter “Triple DES” aka “3DES”
- Uses 3 DES keys: $k_1, k_2, k_3$
- Computes: $C = DES(k_1, DES(k_2, DES(k_3, P)))$
DES

• Goal of 3DES: 3x the key length, $2^{112}$x the attack complexity

• 3DES attack: meet in the middle
  – Assume C, P known
  – Compute $X = E(E(P, k1), k2)$ for all $k1, k2$
  – Compute $Y = D(C, k3)$ for all $k3$
  – Find points where $X=Y$, verify with a second set of C, P

• Key length: 168 bits

• Key strength: 112 bits

• Attack requires $O(2^{n+1})$ crypto operations and $O(2^n)$ storage space
AES

• Advanced Encryption Standard
• DES system limited, longevity questioned
• January 1997, NIST issued a call for a new block cipher
• Requirements
  – Unclassified and publicly disclosed
  – Free from intellectual property restrictions worldwide
  – Usable with key sizes 128, 192, 256
• August 1998, five finalists chosen
  – MARS, RC6, Serpent, Twofish, Rijndael
  – Rijndael selected due to performance
AES

- Similar structure to DES
- Rounds depends on key size
  - 9, 11, 13 for 128, 192, 256
- Per-byte S-boxes (16 for 128-bit AES)
  - Confusion
- Permute bytes, mix together with XOR
  - Diffusion
- Add round-specific value generated from key
AES

• Much research analyzing the structure
  – Look for mathematical relationships between input and output
  – Allow you to mathematically solve for the key
  – Currently solving those equations more difficult than brute-forcing the key

• Currently no significant attacks possible against AES

• Embraced world-wide as the next-generation block cipher standard
Block Ciphers

- A few architectural flaws:
  - Susceptible to change (malleable)
    - User encrypts P to C and transmits
    - Adversary alters C to C’
    - User decrypts C’ to P’ ≠ P
    - For block ciphers, adversary cannot make deterministic changes to P due to diffusion
  - Messages larger than one block must be split up into blocks (known as ECB, Electronic Code Book)
  - Susceptible to block reordering, deletion, replay, etc
Block Ciphers

- Cipher Block Chaining (CBC) addresses some of these issues
- Invented by IBM in 1976
- Each plaintext block is XORed with previous ciphertext block before encryption
- Uses random initialization vector (IV) to ensure uniqueness
Block Ciphers

- CBC mode introduces performance problems
- Can’t parallelize the encryption
- New modes designed:
  - Offset Codebook Mode (OCB)
    - Phil Rogaway, UC Davis, 2002
  - Counter CBC-MAC Mode (CCM)
    - Russ Housley, then IETF Security Chair, 2003
- OCB covered by patents
- CCM created for use in WiFi without need to pay royalties (slightly less efficient)
- CCM embraced by WiMAX, 3G, etc
Block Ciphers

- AES-CCM: Uses AES to generate a “key stream”
  - Feed an incrementing counter into AES blocks
  - Each one computes a pseudo-random sequence
  - XOR with data

- Key stream blocks can be generated in parallel in hardware

\[
\begin{array}{cccc}
\text{Data Block 1} & \text{Data Block 2} & \ldots & \text{Data Block N} \\
\text{AES(key, IV || 1)} & \text{AES(key, IV || 2)} & \ldots & \text{AES(key, IV || N)} \\
\end{array}
\]

\[\text{CBC-MAC} \quad \text{AES(key, IV || 0)}\]

- Looses diffusion properties of AES
Stream Ciphers

• Lesson from one-time pads:
  – XOR plaintext with random sequence to obtain perfect secrecy

• Issue: distributing random sequences to both sender and receiver is difficult

• Solution: use pseudo-random sequences

• Result: stream ciphers
  – AES-CCM is a type of stream cipher
  – Many others exist, predating AES-CCM
Stream Ciphers

• Basic architecture:

\[ E() = D() \]
Stream Ciphers

• Theory behind the creation of keystream generators
  – Need to be provably secure pseudorandom number generators
  – Seed with the key and IV

• Typically based on linear-feedback shift registers

• Implement operations in a binary Galois Field
Stream Ciphers

- Basic idea:
  - Consider a binary Galois Field with irreducible polynomial \( x^3 + x + 1 \)
  - Elements in the field are:
    - Look at first bit in vector
    - Pseudorandom sequence
    - Repeats after \( 2^N - 1 \) cycles
- Implement as a shift register

<table>
<thead>
<tr>
<th>Power</th>
<th>Polynomial</th>
<th>Vector</th>
<th>Regular</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>(000)</td>
<td>0</td>
</tr>
<tr>
<td>( x^0 )</td>
<td>1</td>
<td>(001)</td>
<td>1</td>
</tr>
<tr>
<td>( x^1 )</td>
<td>( x )</td>
<td>(010)</td>
<td>2</td>
</tr>
<tr>
<td>( x^2 )</td>
<td>( x^2 )</td>
<td>(100)</td>
<td>4</td>
</tr>
<tr>
<td>( x^3 )</td>
<td>( x + 1 )</td>
<td>(011)</td>
<td>3</td>
</tr>
<tr>
<td>( x^4 )</td>
<td>( x^2 + x )</td>
<td>(110)</td>
<td>6</td>
</tr>
<tr>
<td>( x^5 )</td>
<td>( x^2 + x + 1 )</td>
<td>(111)</td>
<td>7</td>
</tr>
<tr>
<td>( x^6 )</td>
<td>( x^2 + 1 )</td>
<td>(101)</td>
<td>5</td>
</tr>
</tbody>
</table>
Stream Ciphers

- Keystream Generator used by GSM Cell Phone Encryption (A5/1)
Stream Ciphers

- Most popular stream cipher: RC4 (Rivest Cipher #4)
- Invented by Ron Rivest in 1987 at RSA Laboratories
- Trade secret until it was leaked to the Internet in 1994
- Basic algorithm:
  - Use key/IV to generate random S-Box (permutation of single bytes stored as a vector)
  - Algorithm uses two pointers
    - One incremented by one each time
    - One incremented by S-Box[i] each time (where i is the counter)
  - Each loop values of S-Box[i] and S-Box[j] are swapped

Not based on LFSR
Ideal for software implementation
Stream Ciphers

• Issue with RC4: first few bytes of keystream output is statistically correlated with the original key
  – Need to throw those first few bytes out
  – Results in attack against WiFi networks
Message Integrity

• Block and stream ciphers allow us to conceal the contents of a message
  – Achieves confidentiality
• Attacker can still change messages (either encrypted or unencrypted) and cause problems in a network
• Need a way to ensure a message has not been tampered with
• Message Integrity Codes (MIC)
  – aka Messages Authentication Codes (MAC)
Hash Functions

- MICs are based on hash functions
- Hash functions take an arbitrarily long input and output a uniform-length value

**Input**

- Fox
  - Hash function
  - Hash sum: DFCD3454

- The red fox runs across the ice
  - Hash function
  - Hash sum: 52ED879E

- The red fox walks across the ice
  - Hash function
  - Hash sum: 46042841
Hash Functions

• Properties of hash functions
  – Deterministic: always get the same output for the same input
  – Uniformity: output probability distribution should be uniform
  – Variable range: accept any input length

• Cryptographic hash functions
  – Pseudorandom: output indistinguishable from random
  – Diffusion: minor input change maps to a random output change
  – Preimage Resistant: given an output, it's difficult to compute an input that would give that output
  – Collision Resistant: given input and output, it's difficult to compute another input with same output
Hash Functions

• Simple ones
  – Internet Checksum
    • 16-bit sum of data
  – CRC-32
    • Linear feedback shift register in reverse
    • Push message into it followed by 32 zeros, values remaining in the register at the end are the CRC
    • CRC-32 uses a degree-32 polynomial
Hash Functions

• Cryptographic Hash Functions
  – Message Digest #5 (MD5)
    • Invented by Ron Rivest, 1992
    • Message padded, blocked into 512 bits
    • Permutation run 64 times for each block
      – 16 times on each 128-bit sub-block
    • Outputs added together to form 128-bit hash
  – MD5 Security
    • 1993: “pseudocollision” found
    • 1996: attack on compression function found (not full attack)
    • Mar 2004: effort to find a collision via the birthday paradox began
    • Apr 2004: collisions found requiring 1 hour on supercomputer
    • Mar 2005: forged digital signatures demonstrated
Hash Functions

- **Secure Hash Algorithm #1 (SHA-1)**
  - SHA-0 designed by NSA in 1993
    - Quickly withdrawn without explanation
  - SHA-1 designed by NSA, published in 1995
    - Supposedly fixed issues with SHA-0
  - US Federal Government Standard via NIST
  - Similar in operation to MD5
  - Produces 160-bit digest

- **SHA-1 Security**
  - Feb 2005: Attack demonstrated on reduced-round SHA-1 with complexity $2^{80}$
  - Aug 2005: Attack on full SHA-1 with complexity $2^{63}$
  - 2006: Attack on full SHA-1 with complexity $2^{35}$

- Replaced by SHA-2 (aka SHA-256), phase out by NIST by 2010

- Competition for SHA-3 underway
Hash Functions

• Impact of security flaws
  – Breaks the collision-resistance property
  – Does not break the preimage-resistance property
  – Therefore can still be useful as key derivation function
    • i.e. given key k1, compute new key k2 from it
    • k2 = hash(k1 || entropy)
Message Integrity Checks

- Use hash functions with a cryptographic key

- Basic idea:
  - Message M, Key K
  - Compute MIC = hash(K || M)
  - Send M || MIC
  - Receiver knows K, computes MIC, verifies
  - Receiver knows only someone knowing K could have sent the message
Message Integrity Checks

• Hash Message Authentication Code (HMAC)
  – Designed by Bellare, UCSD, 1996
  – Adopted by NIST as a USG standard

\[ \text{HMAC}_K(m) = h((K \oplus \text{opad}) || h((K \oplus \text{ipad}) || m)) \]

– Secure as long as hash is secure

– 2006: attack showing HMAC with MD4 and SHA-0 is insecure (MD5, SHA-1 still not shown to be vulnerable)
Message Integrity Checks

• Also can build MIC out of block cipher

• Cipherblock Chaining (CBC) MAC
• Similar to CBC encryption mode but without outputting intermediate values
• Security currently unchallenged, no known vulnerabilities
Authenticated Encryption

• Provide both encryption and integrity of a message
  – Guarantee confidentiality and integrity

• Current trend in the standards world

• AES-CCM is an example
  – AES encryption with built-in CBC-MAC
  – Other modes as well, OCB, EAX, GCM
  – GCM is NIST standard
Conclusion

• Current best practices:
  – AES-128 encryption
  – SHA-256 hash function
  – HMAC-SHA-256 or AES-CBC-MAC-128 message integrity
  – AES-CCM or AES-GCM for authenticated encryption