Private Key Cryptography

Professor Yanmin Gong
08/24/2017
What is Cryptography

• Crypto core
  – Secret key establishment
  – Secure communication

confidentiality and integrity
But crypto can do much more

- Digital signatures
- Anonymous communication
- Anonymous digital cash
- Elections
But crypto can do much more (cont.)

- Private auctions
- Privately outsourcing computation

- Zero knowledge (proof of knowledge)
Outline

• History of Crypto
• Perfect Secrecy and One time pad
• Stream Ciphers
• Semantic secrecy
• Block Ciphers
• Hash
History of Crypto: 3000BC-1976

Design crypto system

Use for life or death applications

Believe impossible to break

System is broken

Fix crypto system

Die
Example 1: Mary's cipher

Mary queen of Scot planned to assassinate her cousin queen Elisabeth in 1587.

Communicated plot using substitution cipher

Sir Francis Walsingham broke it using frequency analysis
Example 2: Enigma
A typewriter* that based on wires and rotor setting would emit different letter for every keypress.

![Diagram of Enigma machine]

- **Current state**
- **Letter typed**
- **New state**
- **Letter output**

About $10^{113}$ possibilities to set the wirings and rotors. Lightspeed supercomputer will take \( \gg 10^{17} \) years to check them all.

Believed impossible to break by Germans. (universe is only $10^{10}$ years old)

Broken via heroic efforts by British at Bletchley park

- Cut German U-Boat success in sinking ships by \(~90\%\)
- Sank about 60% of German U-Boats in Mediterranean
Modern Cryptography (1976-)

“We stand today on the brink of a revolution in cryptography” Whit Diffie and Martin Hellman, 1976

<table>
<thead>
<tr>
<th></th>
<th>Data</th>
<th>Cycles</th>
<th>Person-years</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary’s cipher</td>
<td>$10^4$ bytes</td>
<td>N/A</td>
<td>1</td>
<td>Broken</td>
</tr>
<tr>
<td>Enigma</td>
<td>$10^7$ bytes</td>
<td>$10^{13}$</td>
<td>$10^5$</td>
<td>Broken</td>
</tr>
<tr>
<td>1976</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diffie-Hellman/RSA</td>
<td>$10^{18}$ bytes</td>
<td>$10^{25}$</td>
<td>$10^8$</td>
<td>Unbroken!</td>
</tr>
</tbody>
</table>

DH/RSA are *simpler* than Enigma, and allow *public* encryption key

Security through obscurity  →  Security through *simplicity*/mathematics
Principles of Modern Cryptography

• 1. Formal definitions
• 2. Precise assumptions
• 3. Proofs of Security
Formal Definitions: a secure encryption scheme

• Security Goal
  – Regardless of any information an attacker already has, a ciphertext should leak no additional information about the underlying plaintext
  – Mathematical formulation is still missing: prior knowledge, leak information,…

• Threat Model
  – What power does the attacker have?
  – E.g., ciphertext-only attack
Precise Assumptions

• Most modern cryptographic constructions cannot be proven secure unconditionally.
• The proof of security typically rely on assumptions in the theory of computational complexity.
• For validation of assumptions, comparison of schemes, and understanding the necessary assumptions.
Provable security VS Real-world Security

• Incorrect implementation

• To attack, focus on the definition or the underlying assumptions
  – explore how the idealized definition differs from the real-world environment
  – See whether assumptions hold
Cryptosystem

A cryptosystem is a 5-tuple consisting of

\[(E, D, M, K, C)\]

Where,

- **E** is an *encryption* algorithm
- **D** is an *decryption* algorithm
- **M** is the set of *plaintexts*
- **K** is the set of *keys*
- **C** is the set of *ciphertexts*
What is a key?

• A key is an input to a cryptographic algorithm used to obtain confidentiality, integrity, authenticity or other property over some data.
  – The security of the cryptosystem often depends on keeping the key secret to some set of parties
  – The key space is the set of all possible keys
  – Entropy is a measure of the variance in keys
    • typically measured in bits

• Keys are often stored in some secure place:
  – passwords, on disk keyrings, ...
  – TPM, secure co-processor, smartcards, ... ...

• and sometimes not, e.g., certificates
Encryption algorithm

- Algorithm used to make content unreadable by all but the intended receivers

\[ E(key, plaintext) = ciphertext \]
\[ D(key, ciphertext) = plaintext \]

- Algorithm is public, key is private
- Block vs. Stream Ciphers
  - Block: input is fixed blocks of same length
  - Stream: stream of input
Transposition Ciphers

- Scrambles the symbols to produce output
- The key is the permutation of symbols
Substitution Ciphers

- Substitutes one symbol for another (codebook)
- The key is the permutation
Example: Caesar Cipher

- Shift cipher, shift parameter used as the key
- Every character is replaced with the character three slots to the right

\[ E_n(x) = (x + n) \mod 26. \quad D_n(x) = (x - n) \mod 26. \]

[Image of the Caesar Cipher alphabet]

Cryptanalysis of Caesar Ciphers

• Goal: to find plaintext of encoded message
• Given: ciphertext
• Two possible attackers
  – Know it is substitution cipher: frequency analysis
  – Know it is ceasar: simply try all possible keys.
Vigenère cipher

• Using a series of interwoven Caesar ciphers based on the letters of a keyword.
• Plaintext: ATTACKATDAWN
• Key: LEMONLEMONLEMONLE
• Ciphertext: LXFOPVEFRNHR
• https://planetcalc.com/2468/
Cryptanalysis of Vigenère cipher

• Two observations:
  – Frequency analysis will not be successful.
  – Repeating nature of the key

• If a cryptanalyst correctly guesses the key's length, then the cipher text can be treated as interwoven Caesar ciphers, which individually are easily broken.
  – Repeated words may, by chance, sometimes be encrypted using the same key letters, leading to repeated groups in the ciphertext
  
  – Key:  
  
  – Plaintext:  
  
  – Ciphertext:  

ABCDABCDABCDABCDABCDABCDABCDABCD
CRYPTOISSHORTFORCRYPTOGRAPHY
CSASTPKVISIQTGQUCSASTPIUAQJB
Outline

- History of Crypto
- Perfect Secrecy and One time pad
- Stream Ciphers
- Semantic secrecy
- Block Ciphers
- Hash
Cryptographic Schemes

Kerkhoffs' principle: The cipher method must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience.
Cryptographic Schemes

\[\mathcal{E}:\] encryption algorithm \hspace{1cm} \[K_e: \text{encryption key}\]
\[\mathcal{D}:\] decryption algorithm \hspace{1cm} \[K_d: \text{decryption key}\]

**Settings:**

- **public-key (asymmetric):** \[K_e \text{ public, } K_d \text{ secret}\]
- **private-key (symmetric):** \[K_e = K_d \text{ secret}\]
Private key cryptography

• Traditional use of cryptography
• Symmetric keys, where a single key \((k)\) is used for \(E\) and \(D\)

\[
D(k, E(k, p)) = p
\]

• All (intended) receivers have access to key
• Note: Management of keys determines who has access to encrypted data
  – E.g., password encrypted email
• Also known as symmetric key cryptography
Perfectly Secret Encryption (by Shannon)

Attacker’s abilities: **Ciphertext only attack** (for now)

Possible security requirements:

- attempt #1: **attacker cannot recover secret key**
- attempt #2: **attacker cannot recover all of plaintext**

Shannon’s idea:

**CT should reveal no “info” about PT**
Shannon’s perfect secrecy (cont.)

- Definition: An encryption scheme \((\text{Gen}, \text{Enc}, \text{Dec})\) with message space \(M\) is perfectly secret if for every probability distribution over \(M\), every message \(m\) in \(M\), and every ciphertext \(c\) in \(C\) for which \(\Pr[C=c] > 0\):

\[
\Pr[M=m | C=c] = \Pr[M = m]
\]

- Requires that the probability distribution of the ciphertext does not depend on the plaintext:

\[
\Pr[\text{Enc}(m)=c] = \Pr[\text{Enc}(m') = c]
\]
Adversarial indistinguishability experiment

• Adversarial indistinguishability experiment
  – The adversary A outputs a pair of messages $m_0, m_1$
  – A key $k$ is generated using Gen, and a uniform bit $b = 0$ or $1$ is chosen. Ciphertext $c \leftarrow \text{Enc}_k(m_b)$ and given to A. $c$ is the challenge ciphertext.
  – A outputs a bit $b'$. The output of the experiment is defined to be $1$ if $b' = b$ and $0$ o.w.
  – If the output of the experiment is $1$, A succeeds

• Perfectly indistinguishable
  – for every A it holds that A’s probability to succeed is $\frac{1}{2}$.
  – Alternative definition for perfect secrecy.

• Formal Security Proof
Weak Ciphers

• Substitution cipher
• Caesar cipher
• Vigenere cipher
• ...

A Good Cipher: One Time Pad

Example where message $m$ and ciphertext $c$ are defined in $\{0, 1, 2, \ldots, 25\}$.

Encryption: $E(k, m) = k + m \mod 26$, $D(k, c) = c - k \mod 26$

<table>
<thead>
<tr>
<th>E</th>
<th>Q</th>
<th>N</th>
<th>V</th>
<th>Z</th>
<th>ciphertext</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>16</td>
<td>13</td>
<td>21</td>
<td>25</td>
<td>ciphertext</td>
</tr>
<tr>
<td>- 23</td>
<td>12</td>
<td>2</td>
<td>10</td>
<td>11</td>
<td>key</td>
</tr>
<tr>
<td>= 7</td>
<td>4</td>
<td>11</td>
<td>11</td>
<td>14</td>
<td>ciphertext - key</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>E</td>
<td>L</td>
<td>O</td>
<td>ciphertext - key (mod 26)</td>
</tr>
</tbody>
</table>

When $M=C=K=\{0,1\}^n$

Modular addition can be substituted as bitwise exclusive-or (XOR), denoted as $\oplus$

$E(k, m) = k \oplus m$, $D(k, c) = k \oplus c$
Cryptanalysis of OTP

Suppose Eve intercepts Alice's ciphertext: "EQNVZ". If Eve had infinite time, she would find that the key "XMCKL" would produce the plaintext "HELLO", but she would also find that the key "TQURI" would produce the plaintext "LATER", an equally plausible message.

**Lemma:** OTP has perfect secrecy (i.e. no CT only attacks)

**Proof:** assume each bits in m is equally likely, then you have no information to work with.
Bad News

• Bad news:
  – Truly random one-time pad values, which is a non-trivial requirement.
  – Secure generation and exchange of the one-time pad values, which must be at least as long as the message.
  – Preventing reuse in whole or part, hence “one time”.
Conventional symmetric encryption algorithms use complex patterns of substitution and transpositions. For the best of these currently in use, it is not known whether there can be a cryptanalytic procedure which can reverse (or, usefully, partially reverse) these transformations without knowing the key used during encryption. Asymmetric encryption algorithms depend on mathematical problems that are thought to be difficult to solve, such as integer factorization and discrete logarithms. However, there is no proof that these problems are hard, and a mathematical breakthrough could make existing systems vulnerable to attack.
Outline

• History of Crypto
• Perfect Secrecy and One time pad
• Computational Secrecy and Stream Ciphers
• Block Ciphers
• Hash
Classical cryptography vs modern cryptography

• Classical cryptography
  – Before mid-1970s and 1980s
  – Precise mathematical definitions, but no unproven computational assumptions.

• Modern Cryptography
  – Computational science: Security of a “practical” system must rely not on the impossibility but on the computational difficulty of breaking the system

Rather than:

“It is impossible to break the scheme”

We might be able to say:

“No attack using \( \leq 2^{160} \) time succeeds with probability \( \geq 2^{-20} \)”

I.e., Attacks can exist as long as cost to mount them is prohibitive, where \( \text{Cost} = \text{computing time/memory, } $$$ \) \((t, \varepsilon)\)-secure
Two relaxations

• Threat model
  Security parameter $n$
  - Randomized algorithms running in time polynomial in $n$

• Adversaries can potential succeed with some Negligible probability
Asymptotic Approach for computational security

- Integer-valued security parameter (denoted by n)
  - Parameterizes both cryptographic schemes and involved parties
  - Allow the honest party to tune the security of a scheme
- Randomized algorithms running in time polynomial in n
  - There is some polynomial p such that the adversary runs for at most p(n) when the security parameter is n
- Small probability of success
  - Success probabilities smaller than any inverse polynomial in n (negligible probability).
- PPT: probabilistic polynomial-time
- Definition of asymptotic approach
  - A scheme is secure if any PPT adversary succeeds in breaking the scheme with at most negligible probability.
- Translate to concrete security in practice when deployed.

More details in Chapter 3.1, 3.2, Introduction to Modern Cryptography
Adversarial indistinguishability experiment

- Adversarial indistinguishability experiment
  - The adversary $A$ is given input $1^n$, outputs a pair of messages $m_0, m_1$, $|m_0| = |m_1|$.
  - A key $k$ is generated using $\text{Gen}$, and a uniform bit $b = 0$ or $1$ is chosen. Ciphertext $c \leftarrow \text{Enc}_k(m_b)$ and given to $A$. $c$ is the challenge ciphertext.
  - $A$ outputs a bit $b'$. The output of the experiment is defined to be $1$ if $b' = b$ and $0$ o.w.
  - If the output of the experiment is $1$, $A$ succeeds.

- A private-key encryption scheme $(\text{Gen}, \text{Enc}, \text{Dec})$ has indistinguishable encryptions in the presence of an eavesdropper, or is EAV-secure
  - if for all probabilistic polynomial-time adversaries $A$ there is a negligible function $\text{negl}$ such that for all $n$, $\Pr[A \text{ succeeds}] \leq \frac{1}{2} + \text{negl}(n)$.

- Semantic security definition
Building blocks for private key encryption

- Pseudorandom generator
- Stream cipher
Stream Ciphers: making OTP practical

• idea: replace “random” key by “pseudorandom” key
• Generating randomness
  – Ideal: Unlimited supply of independent, unbiased random bits
  – Modern random-number generation
    • a pool of high-entropy data is collected;
    • the high-entropy data is processed to yield a sequence of nearly independent and unbiased bits (e.g., coin flipping with probability p)
  – Pseudorandom generator
    • Use a small amount of randomness to generate a large amount of pseudorandomness.
    • Pseudorandomness is a property of a distribution on strings. Informally call a string sampled from the pseudorandom generator as a pseudorandom string.
    • Formal definition (Definition 3.14 in Katz and Lindell’s book)
• Stream ciphers are used in practice to instantiate pseudorandomness

Question

• Can stream ciphers have perfect secrecy?

No, since the key is shorter than the message
What is the security for stream cipher?

Stream ciphers cannot have perfect secrecy !!

• Need a different definition of security

• Security will depend on specific PRG
PRG must be unpredictable

If PRG is predictable, there could be a problem when you know part of the plaintext:

\[ c \oplus m |_{1,\ldots,i} \rightarrow G(k) |_{i+1,\ldots,n} \]

We say that \( G : K \rightarrow \{0,1\}^n \) is **predictable** if:

There exists an \textit{“eff”} algorithm \( A \) and \( \exists \ 0<i<n-1 \) s.t. \( \Pr[A(G(k)) |_{1,\ldots,i} = G(k) |_{i+1}] > \frac{1}{2} + \varepsilon \) for non-negligible \( \varepsilon \)

**Def:** PRG is **unpredictable** if it is not predictable

\[ \Rightarrow \ \forall i: \text{no “eff” adv. can predict bit (i+1) for “non-neg” } \varepsilon \]
Weak PRGs

• Linear congruential generator

```python
def lcg(modulus, a, c, seed):
    while True:
        seed = (a * seed + c) % modulus
        yield seed
```

Don’t use for crypto!

• glibc random():

```python
r[i] ← ( r[i-3] + r[i-31] ) % 2^{32}
output r[i] >> 1
```

The linear relationships between the outputs bits are exactly what enable an easy attack.
Negligible vs Non-negligible

• In practice: $\varepsilon$ is a scalar and
  - $\varepsilon$ non-neg: $\varepsilon \geq \frac{1}{2^{30}}$ (likely to happen over 1GB of data)
  - $\varepsilon$ negligible: $\varepsilon \leq \frac{1}{2^{80}}$ (won’t happen over life of key)
Attack 1: \textbf{two time pad is insecure} !!

Never use stream cipher key more than once !!

\[ C_1 \leftarrow m_1 \oplus \text{PRG}(k) \]
\[ C_2 \leftarrow m_2 \oplus \text{PRG}(k) \]

Eavesdropper does:
\[ C_1 \oplus C_2 \rightarrow \]

Enough redundancy in English and ASCII encoding that:
\[ m_1 \oplus m_2 \rightarrow m_1, m_2 \]
Real world examples

Never use stream cipher key more than once !!

• Network traffic: negotiate new key for every session (e.g. TLS)

• Disk encryption: typically do not use a stream cipher
Real world examples

- **Project Venona: Russian vs US**
- **MS-PPTP (windows NT)**
- **802.11b WEP**

**Don’t use same pad twice**
Need different keys for $C \rightarrow S$ and $S \rightarrow C$

40,000 frames are enough to recover the keys
Don’t use related keys

**IV**
ciphetext

**PRG( IV II k )**

**m**

CRC(m)
Attack 2: no integrity  (OTP is malleable)

• Modifications to ciphertext are undetected and have **predictable** impact on plaintext

\[
m \oplus k \rightarrow \text{dec}(\oplus k) \rightarrow (m \oplus k) \oplus p
\]
Real world examples of stream ciphers

• RC4
  – Used in HTTPS and WEP
  – Weaknesses:
    1. Bias in initial output:  \[ \Pr[\, 2^{\text{nd}} \text{ byte } = 0 \,] = \frac{2}{256} \]
    2. Prob. of (0,0) is  \[ \frac{1}{256^2} + \frac{1}{256^3} \]
    3. Related key attacks
  Not recommended

• CSS
  – DVD encryption (CSS):  2 LFSRs
  – GSM encryption (A5/1,2):  3 LFSRs
  – Bluetooth (E0):  4 LFSRs
  Badly broken

• Modern stream ciphers: eStream
Modern stream ciphers: eStream

PRG: \( \{0,1\}^s \times R \rightarrow \{0,1\}^n \)

Nonce: a non-repeating value for a given key.

\[ E(k, m; r) = m \oplus \text{PRG}(k; r) \]

The pair \((k, r)\) is never used more than once.

Chapter 6.1.3 Trivium, Introduction to Modern cryptography
Block Cipher

• A block cipher is generally only considered “good” if the best known attack has time complexity roughly equivalent to a brute-force search for the key

• Designed to behave as strong pseudorandom permutations.
Abstractly: PRPs and PFRs

• Pseudorandom functions
  – Random looking strings -> random looking functions
  – Function $F_k$ is indistinguishable from a function randomly selected from the set of functions.

Abstractly: PRPs and PFRs

- **Pseudo Random Function (PRF)** defined over \((K, X, Y)\):

  \[ F: K \times X \rightarrow Y \]

  such that there exists “efficient” algorithm to evaluate \(F(k, x)\)

- **Pseudo Random Permutation (PRP)** defined over \((K, X)\):

  \[ E: K \times X \rightarrow X \]

  such that:
  1. There exists “efficient” deterministic algorithm to evaluate \(E(k, x)\)
  2. The function \(E(k, \cdot)\) is one-to-one
  3. There exists “efficient” inversion algorithm \(D(k, y)\)
• Example PRPs: 3DES, AES, ...

  AES: $K \times X \rightarrow X$ where $K = X = \{0,1\}^{128}$

  3DES: $K \times X \rightarrow X$ where $X = \{0,1\}^{64}$, $K = \{0,1\}^{168}$

• Functionally, any PRP is also a PRF.
  – A PRP is a PRF where $X=Y$ and is efficiently invertible.
Secure PRFs

• Let \( F: K \times X \rightarrow Y \) be a PRF

\[
\begin{align*}
\text{Funs}[X,Y]: & \quad \text{the set of all functions from } X \text{ to } Y \\
S_F & = \{ F(k, \cdot) \quad \text{s.t.} \quad k \in K \} \quad \subseteq \quad \text{Funs}[X,Y]
\end{align*}
\]

• Intuition: a PRF is secure if a random function in \( \text{Funs}[X,Y] \) is indistinguishable from a random function in \( S_F \)
Block Cipher

Canonical examples:

1. 3DES: $n = 64$ bits, $k = 168$ bits
2. AES: $n = 128$ bits, $k = 128, 192, 256$ bits
Block Ciphers Built by Iteration

R(k, m) is called a round function

for 3DES (n=48),  for AES-128 (n=10)
Data Encryption Standard (DES)

• Introduced by the US NBS (now NIST) in 1972
• Signaled the beginning of the modern area of cryptography
• Block cipher
  – Fixed sized input
• 8-byte input and a 8-byte key (56-bits+8 parity bits)
DES: core idea – Feistel Network

Given functions \( f_1, \ldots, f_d : \{0,1\}^n \rightarrow \{0,1\}^n \)

Goal: build invertible function \( F : \{0,1\}^{2n} \rightarrow \{0,1\}^{2n} \)
Claim: for all \( f_1, \ldots, f_d: \{0,1\}^n \rightarrow \{0,1\}^n \)

Feistel network \( F: \{0,1\}^{2n} \rightarrow \{0,1\}^{2n} \) is invertible

Proof: construct inverse
DES Round

• Initial round permutes input, then 16 rounds
• Each round key (subkey) \((k_i)\) is 48 bits of input key
• Function \(f\) is a substitution table (\(s\)-boxes)
Substitution Box (S-box)

• A substitution box (or S-box) is used to obscure the relationship between the plaintext and the ciphertext
  – Shannon's property of confusion: the relationship between key and ciphertext is as complex as possible.
  – In DES S-boxes are carefully chosen to resist cryptanalysis

• Thus, that is where the security comes from.
Cryptanalysis of DES

- DES has an effective 56-bit key length
  - Wiener: $1,000,000 - 3.5$ hours (never built)
  - July 17, 1998, the EFF DES Cracker, which was built for less than $250,000 < 3$ days
  - January 19, 1999, Distributed.Net (w/EFF), 22 hours and 15 minutes (over nearly 100,000 machines)
  - September 2002, FPGA implementation, 12 hours.
- We all assume that NSA and agencies like it around the world can crack (recover key) DES in milliseconds
- What now? Give up on DES?
Exhaustive Search for block cipher key

**Goal:** given a few input output pairs \((m_i, c_i = E(k, m_i))\) \(i=1,..,3\) find key \(k\).

For two DES pairs \((m_1, c_1=DES(k, m_1))\), \((m_2, c_2=DES(k, m_2))\)

unicity prob. \(\approx 1 - 1/2^{71}\)

For AES-128: given two inp/out pairs, unicity prob. \(\approx 1 - 1/2^{128}\)

\(\Rightarrow\) two input/output pairs are enough for exhaustive key search.
Variants of DES

DESX (two additional keys \(\sim= 118\)-bits)

Triple DES (three DES keys \(\sim= 112\)-bits)

Keys \(k_1, k_2, k_3\)

\[c = E(k_3, D(k_2, E(k_1, p)))\]
Attacks on the implementation

1. Side channel attacks:
   - Measure **time** to do enc/dec, **measure power** for enc/dec

2. Fault attacks:
   - Computing errors in the last round expose the secret key $k$

$\Rightarrow$ do not even implement crypto primitives yourself ...
Advanced Encryption Standard (AES)

- Result of international NIST bakeoff between cryptographers
  - Intended as replacement for DES
  - Rijndael (pronounced “Rhine-dall”)
  - Currently implemented in many devices and software, but not yet fully embraced
  - Cryptography community is actively vetting the theory and implementations (stay tuned)
AES is a Subs-Perm network (not Feistel)

Key mixing; substitution; permutation
AES-128 schematic

10 rounds

(1) ByteSub
(2) ShiftRow
(3) MixColumn

key expansion:
16 bytes → 176 bytes

invertible

16 bytes	key

4

input

4

k₀

k₁

k₂

k₉

k₁₀

output

4
The round function

- **ByteSub:** a 1 byte S-box. 256 byte table (easily computable)

- **ShiftRows:**

- **MixColumns:**
Hash Functions and Applications

• Message Authentication using hash function
• Generic attacks on hash function
• Random-Oracle Model
• Additional Applications
Hash Algorithms

• Compression of data into a hash value
  – E.g., $h(d) = \text{parity}(d)$
  – Such algorithms are generally useful in systems (speed/space optimization)

• ... as used in cryptosystems
  – One-way: computationally hard to invert $h()$, i.e., compute $h^{-1}(y)$ where $y = h(d)$
  – Collision resistant: hard to find two data $x_1$ and $x_2$ such that $h(x_1) == h(x_2)$
Construction of Hash Function

• Collision-resistant compression handling fix-length inputs
  – Block ciphers

• Domain extension to handle arbitrary-length inputs
  – The Merkle-Damgard Transform
**Compr. func. from a block cipher**

E: $K \times \{0,1\}^n \rightarrow \{0,1\}^n$ a block cipher.

The **Davies-Meyer** compression function:

$$h(H, m) = E(m, H) \oplus H$$

**Thm:** Suppose E is an ideal cipher (collection of $|K|$ random perms.). Finding a collision $h(H,m) = h(H',m')$ takes $O(2^{n/2})$ evaluations of (E,D).

Best possible !!
Suppose we define \( h(H, m) = E(m, H) \)

Then the resulting \( h(.,.) \) is not collision resistant:

to build a collision \((H,m) \) and \((H',m')\)
choose random \((H,m,m')\) and construct \(H'\) as follows:

A. \( H' = D(m', E(m,H)) \)

B. \( H' = E(m', D(m,H)) \)

C. \( H' = E(m', E(m,H)) \)

D. \( H' = D(m', D(m,H)) \)

A. \( E(m', H') = E(m,H) \)
Case study: SHA-256

- Merkle-Damgard function
- Davies-Meyer compression function
- Block cipher: SHACAL-2

Diagram:
- 512-bit key
- SHACAL-2
- 256-bit block
The Merkle-Damgard iterated construction

Given \( h: T \times X \rightarrow T \) (compression function)
we obtain \( H: X^{\leq L} \rightarrow T \). \( H_i \) - chaining variables

PB: padding block

If \( h \) is collision resistant, then so is \( H \).

If no space for PB add another block
MD collision resistance

**Thm:** if \( h \) is collision resistant then so is \( H \).

**Proof:** collision on \( H \) \( \Rightarrow \) collision on \( h \)

Suppose \( H(M) = H(M') \). We build collision for \( h \).

\[
IV = H_0 , \quad H_1 , \quad ... , \quad H_t , \quad H_{t+1} = H(M) \\
IV = H_0' , \quad H_1' , \quad ... , \quad H_r' , \quad H_{r+1}' = H(M')
\]

\[
h(H_t, M_t \| PB) = H_{t+1} = H_{r+1}' = h(H_r', M_r' \| PB')
\]
Hash Functions

• How do you design a “strong cryptographic hash function?”

• No formal basis
  ▶ Concern is backdoors

• MD2, MD4, MD5 (128bit):
  ▶ Broken, Broken, Broken
  ▶ MD4, MD5: Similar, but complex functions in multiple passes

• SHA-1 (160 bit)
  ▶ “Complicated function”
  ▶ Theoretical weaknesses

• SHA-2 (224, 256, 384 or 512-bit)

• SHA-3 (224, 256, 384 or 512-bit)
Applications of Hash Function

• MAC
• Hash Chain
• Random Oracle Model
• Commitment
Message Authentication Code

• MAC
  – Used in protocols to authenticate content, authenticates integrity for data d

• To simplify, hash function $h()$, key $k$, data $d$
  – $\text{MAC}(k,d) = h(k \oplus d)$
  – An example of the Merkle-Damgard construction.

• Why does this provide integrity?
  – Cannot produce $\text{mac}(k,d)$ unless you know $k$ and $d$
  – If you could, you could invert $h()$. 
A simple proof

• Setup: you know $d$ and have an algorithm $X(d)$ that produces $\text{MAC}(k,d)$ without $k$ (assume $d$ known).

• Suppose $X()$ exists:
  \[ d = 0 \]
  then, $X(d) = h(k \oplus 0) = h(k)$

• There are two possible explanations

  • $k$ is constant (which it is not)
  
  • $X(d)$ receives $k$ from input (which by definition it does not)

  • ... a contradiction.
HMAC

• MAC that meets the following properties
  – Collision-resistant
  – Attacker cannot compute proper digest without knowing K
    • Even if attacker can see an arbitrary number of digests H(k+x)
• Simple MAC has a flaw
  – Block hash algorithms mean that new content can be added
  – Turn H(K+m) to H(K+m+m’) where m’ is controlled by an attacker
    – SSL 2.0 was vulnerable to such an attack.
• HMAC(K, d) = H(K_2 + H(K_1 + d))
  – Attacker cannot extend MAC as above
HMAC in pictures

Similar to the NMAC PRF. main difference: the two keys $k_1, k_2$ are dependent
Using hash values as authenticators

• Consider the following scenario
  – Alice is a teacher who has not decided if she will cancel the next lecture
  – When she does decide, she communicates to Bob the student through Mallory, her evil TA
  – She does not care if Bob shows up to a cancelled class
  – Alice does not trust Mallory to deliver the message

• She and Bob use the following protocol
  – Alice invents a secret t
  – Alice gives Bob h(t), where h() is a crypto hash function
  – If she cancels class, she gives t to Mallory to go to Bob
  – If does not cancel class, she does nothing
  – If Bob receives the token t, he knows that Alice sent it
Hash Authenticators

• Why is this protocol secure?
  – t acts as an authenticated value because Mallory could not have produced t without inverting h()
  – Note: Mallory can convince Bob that class is occurring when it is not by simply not delivering h(t)

• Here hash preimages are as good as authenticators

• Bob got the original value h(t) from Alice directly
Now consider the case where Alice wants to do the same protocol, only for all 26 classes (the semester)

Alice and Bob use the following protocol

- Alice invents a secret $t$
- Alice gives Bob $H^{26}(t)$, where $H^{26}()$ is 26 repeated applications of $H()$.
- If she cancels class on day $d$, she gives $H^{(26-D)}(t)$ to Mallory, e.g.,
  - If cancels on day 1, she gives Mallory $H^{25}(t)$
  - If cancels on day 2, she gives Mallory $H^{24}(t)$
  - ......
  - If cancels on day 25, she gives Mallory $H^{1}(t)$
  - If cancels on day 26, she gives Mallory $H^{0}(t)$
- If does not cancel class, she does nothing

If Bob receives the token $t$, he knows that Alice sent it
Hash Chain (cont.)

- Why is this protocol secure?
  - On day $d$, $H^{(26-d)}(t)$ acts as an authenticated value ( Authenticator) because Mallory could not create $t$ without inverting $H()$ because for any $H^k(t)$ she has $k>(26-d)$
  - That is, Mallory potentially has access to the hash values for all days prior to today, but that provides no information on today’s value, as they are all post-images of today’s value
  - Note: Mallory can again convince Bob that class is occurring by not delivering $H^{(26-d)}(t)$

- Chain of hash values are ordered authenticators

- Important that Bob got the original value $H^{26}(t)$ from Alice directly (was provably authentic)
Birthday Attack

- $H: \{0,1\}^* \rightarrow \{0,1\}^l$
- Trivial collision-finding attack running in time $O(2^l)$
- Birthday attack
  - Assume $y_i = H(x_i)$ is uniformly distributed in $\{0,1\}^l$
  - Choose values $y_1, \ldots, y_q \in \{0,1\}^l$ uniformly at random, what is the probability to find $i,j$ with $y_i = y_j$?
  - Take $q \sim 2^l/2$ yields a collision with probability roughly $1/2$
Random Oracle Model

- Treats a cryptographic hash function $H$ as a truly random function.
- $H$ is evaluated only by “querying” an oracle, which can be thought of as a “black box” that returns $H(x)$ when given input $x$.
- Random oracle model versus “standard model”.
Formal methodology based on random-oracle model

- Step 1: a scheme is designed and proven secure in the random-oracle model
- Step 2: when we want to implement the scheme in the real world, a random oracle is not available. Instead, the random oracle is instantiated with an appropriately designed cryptographic hash function $H$.
- Note: a pseudorandom function is not a random oracle because it is only pseudorandom if the key is secret. In the random-oracle model, all parties need to be able to compute the function and there can be no secret key.
Commitment

• A commitment scheme allows one party to commit to a message m by sending a commitment value com, while obtaining the following seemingly contradictory properties
  – Hiding: the commitment reveals nothing about m
  – Binding: it is infeasible for the committer to output a commitment com that it can later open as two different messages m, m’.

• Similar as a digital envelope
Secure commitment scheme from a random oracle $H$

- Sender: choose uniform $r \in \{0,1\}^n$ and outputs $\text{com} := H(m || r)$. 
Key management

Problem: n users. Storing mutual secret keys is difficult

Total: $O(n)$ keys per user
A better solution

Online Trusted 3\textsuperscript{rd} Party (TTP)

\[\text{Every user only remembers one key.}\]
Generating keys: a toy protocol

Alice wants a shared key with Bob. Eavesdropping security only.

Bob \((k_B)\)  
Alice \((k_A)\)  
TTP

```
| Choose random \(k_{AB}\)  
| \(E(k_A, "A,B"\|k_{AB})\)  
| \(E(k_B, "A,B"\|k_{AB})\)  
| Ticket  
| \(k_{AB}\)  
| \(k_{AB}\)  
| \((E,D)\) a CPA-secure cipher  
```
Generating keys: a toy protocol

Alice wants a shared key with Bob. Eavesdropping security only.

Eavesdropper sees: $E(k_A, \text{"A, B" II } k_{AB})$; $E(k_B, \text{"A, B" II } k_{AB})$

$(E,D)$ is CPA-secure $\Rightarrow$ eavesdropper learns nothing about $k_{AB}$

Note: TTP needed for every key exchange, knows all session keys.
   (basis of Kerberos system)
Toy protocol: insecure against active attacks

Example: insecure against replay attacks

Attacker records session between Alice and merchant Bob

– For example a book order

Attacker replays session to Bob

– Bob thinks Alice is ordering another copy of book
Key question

Can we generate shared keys without an online trusted 3rd party?

Answer: yes!

Starting point of public-key cryptography: