Hash Functions

Hash Function Motivation

- Suppose Alice signs M
  - Alice sends M and $S = [M]_{\text{Alice}}$ to Bob
  - Bob verifies that $M = \{S\}_{\text{Alice}}$
  - Aside: Is it OK to just send S?
- If M is big, $[M]_{\text{Alice}}$ is costly to compute
- Suppose instead, Alice signs $h(M)$, where $h(M)$ is much smaller than M
  - Alice sends M and $S = [h(M)]_{\text{Alice}}$ to Bob
  - Bob verifies that $h(M) = \{S\}_{\text{Alice}}$

Crypto Hash Function

- Crypto hash function $h(x)$ must provide
  - **Compression** — output length is small
  - **Efficiency** — $h(x)$ easy to computer for any x
  - **One-way** — given a value y it is infeasible to find an x such that $h(x) = y$
  - **Weak collision resistance** — given x and $h(x)$, infeasible to find $y \neq x$ such that $h(y) = h(x)$
  - **Strong collision resistance** — infeasible to find any x and y, with $x \neq y$ such that $h(x) = h(y)$
  - Lots of collisions exist, but hard to find one

Pre-Birthday Problem

- Suppose N people in a room
- How large must N be before the probability someone has same birthday as me is $\geq 1/2$
  - Solve: $1/2 = 1 - (364/365)^N$ for N
  - Find $N = 253$

Birthday Problem

- How many people must be in a room before probability is $\geq 1/2$ that two or more have same birthday?
  - $1 - 365/365 \cdot 364/365 \cdot\cdot\cdot (365-N+1)/365$
  - Set equal to 1/2 and solve: **N = 23**
- Surprising? A paradox?
- Maybe not: "Should be" about sqrt(365) since we compare all **pairs** x and y

$N!/((2!)(N-2!)) = N(N-1)/2 \approx N^2 \leq 365$, $N \approx 19$

Of Hashes and Birthdays

- If $h(x)$ is N bits, then 2^N different hash values are possible
- $\text{sqrt}(2^N) = 2^{N/2}$
- Therefore, hash about $2^{N/2}$ random values and you expect to find a collision
- **Implication:** secure N bit symmetric key requires 2^{N-1} work to "break" while secure N bit hash requires $2^{N/2}$ work to "break"

Non-crypto Hash (1)

- Data $X = (X_0, X_1, X_2, \ldots, X_{n-1})$, each X_i is a byte
- $\text{hash}(X) = X_0 + X_1 + X_2 + \ldots + X_{n-1}$ mod 256
- Is this secure?
- Example: $X = (10101010, 00001111)$
- Hash is 10111001
- But so is hash of $Y = (00001111, 10101010)$
- Easy to find collisions, so **not** secure…

Non-crypto Hash (2)

- Data $X = (X_0,X_1,X_2,\ldots,X_{n-1})$
- Suppose hash is
  - $h(X) = nX_0+(n-1)X_1+(n-2)X_2+\ldots+1\cdot X_{n-1}$
- Is this hash secure?
- At least
  - $h(10101010,00001111)\neq h(00001111,10101010)$
- But hash of $(00000001,00001111)$ is same as hash of $(00000000,00010001)$
- This hash is used in the (non-crypto) application.

Non-crypto Hash (3)

- Cyclic Redundancy Check (CRC)
- Essentially, CRC is the remainder in a long division problem
- Good for detecting burst **errors**
- But easy to construct collisions
- CRC sometimes mistakenly used in crypto applications (WEP)

Popular Crypto Hashes

- **MD5** — invented by Rivest
  - 128 bit output
  - Note: MD5 collision recently found
- **SHA-1** — A US government standard (similar to MD5)
  - 180 bit output
- Many others hashes, but MD5 and SHA-1 most widely used
- Hashes work by hashing message in blocks

Crypto Hash Design

- Desired property: **avalanche effect**
  - Change to 1 bit of input should affect about half of output bits
- Crypto hash functions consist of some number of rounds
- Want security and speed
  - Avalanche effect after few rounds
  - But simple rounds
- Analogous to design of block ciphers

HMAC

- Can compute a MAC of M with key K using a "hashed MAC" or **HMAC**
- HMAC is an example of a keyed hash
  - Why do we need a key?
- How to compute HMAC?
- Two obvious choices
  - h(K,M)
  - h(M,K)

HMAC

- Should we compute HMAC as $h(K,M)$?
- Hashes computed in blocks
  - $h(B_1,B_2) = F(F(A,B_1),B_2)$ for some F and constant A
  - Then $h(B_1,B_2) = F(h(B_1),B_2)$
- Let $M' = (M,X)$
  - Then $h(K,M') = F(h(K,M),X)$
  - Attacker can compute HMAC of M' without K
- Is $h(M,K)$ better?
  - Yes, but… if $h(M') = h(M)$ then we might have $h(M,K)=F(h(M),K)=F(h(M'),K)=h(M',K)$

The Right Way to HMAC

- Described in RFC 2104
- Let B be the block length of hash, in bytes
  - B = 64 for MD5 and SHA-1 and Tiger
- ipad = 0x36 repeated B times
- opad = 0x5C repeated B times
- Then

$$\text{HMAC}(M,K) = H(K \oplus \text{opad}, H(K \oplus \text{ipad}, M))$$

Hash Uses

- Authentication (HMAC)
- Message integrity (HMAC)
- Message fingerprint
- Data corruption detection
- Digital signature efficiency
- Anything you can do with symmetric crypto

Online Auction

- Suppose Alice, Bob and Charlie are bidders
- Alice plans to bid A, Bob B and Charlie C
- They don't trust that bids will stay secret
- Solution?
  - Alice, Bob, Charlie submit **hashes** h(A), h(B), h(C)
  - All hashes received and posted online
  - Then bids A, B and C revealed
- Hashes don't reveal bids (one way)
- Can't change bid after hash sent (collision)

Spam Reduction

- Spam reduction
- Before I accept an email from you, I want proof that you spent "effort" (e.g., CPU cycles) to create the email
- Limit amount of email that can be sent
- Make spam much more costly to send

Spam Reduction

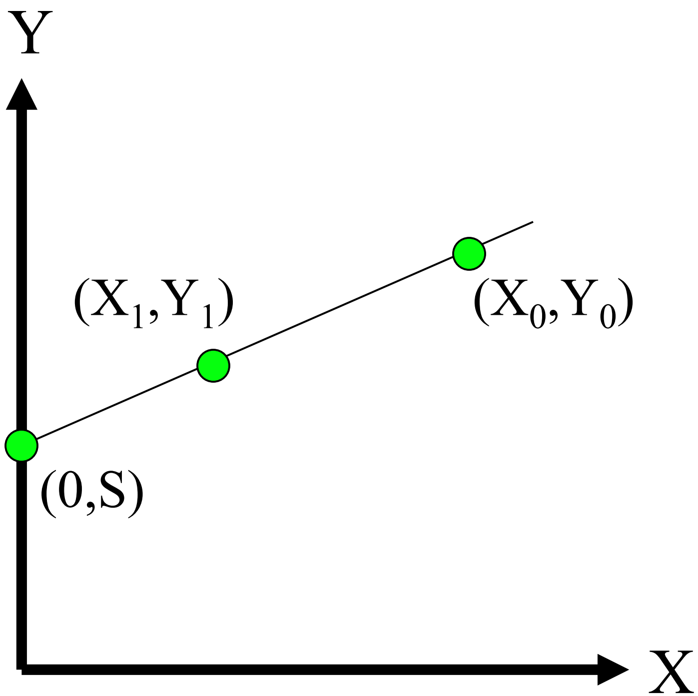
- Let M = email message
- Let **R** = value to be determined
- Let T = current time
- Sender must find **R** such that
  - hash(M,**R**,T) = (00…0,X), where
  - N initial bits of hash are **all zero**
- Sender then sends (M,**R**,T)
- Recipient accepts email, provided
  - hash(M,**R**,T) begins with N zeros

Spam Reduction

- Sender: hash(M,R,T) begins with N zeros
- Recipient: verify that hash(M,R,T) begins with N zeros
- **Work for sender:** about **2^N hashes**
- **Work for recipient: 1 hash**
- Sender's work increases exponentially in N
- Same work for recipient regardless of N
- Choose N so that
  - Work acceptable for normal email users
  - Work unacceptably high for spammers!

Secret Sharing

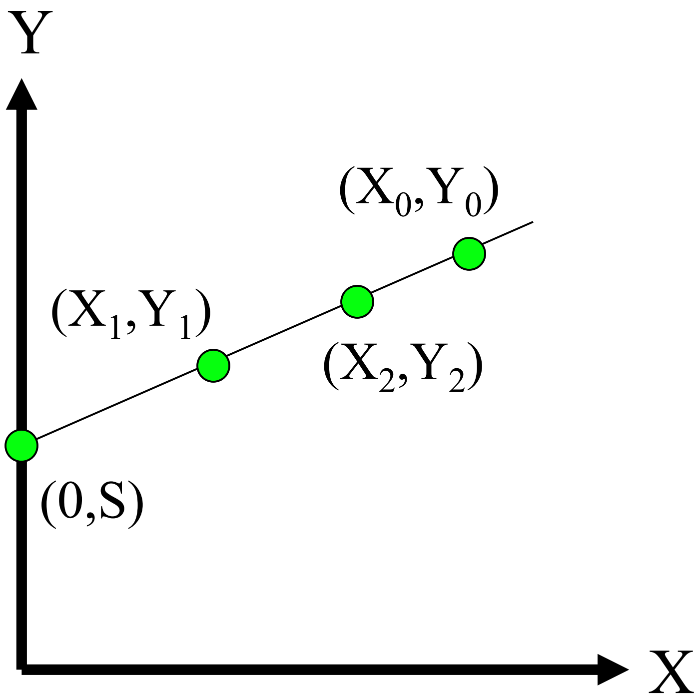
Shamir's Secret Sharing



2 out of 2

- Two points determine a line
- Give (X_0,Y_0) to Alice
- Give (X_1,Y_1) to Bob
- Then Alice and Bob must cooperate to find secret S
- Also works in discrete case
- Easy to make "m out of n" scheme for any $m \leq n$

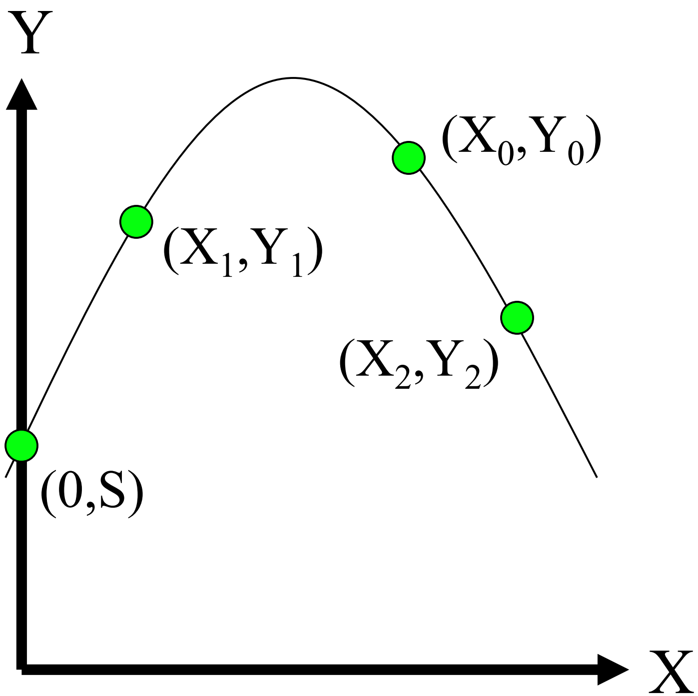
Shamir's Secret Sharing



2 out of 3

- Give (X_0,Y_0) to Alice
- Give (X_1,Y_1) to Bob
- Give (X_2,Y_2) to Charlie
- Then any two of Alice, Bob and Charlie can cooperate to find secret S
- But no one can find secret S
- A "2 out of 3" scheme

Shamir's Secret Sharing



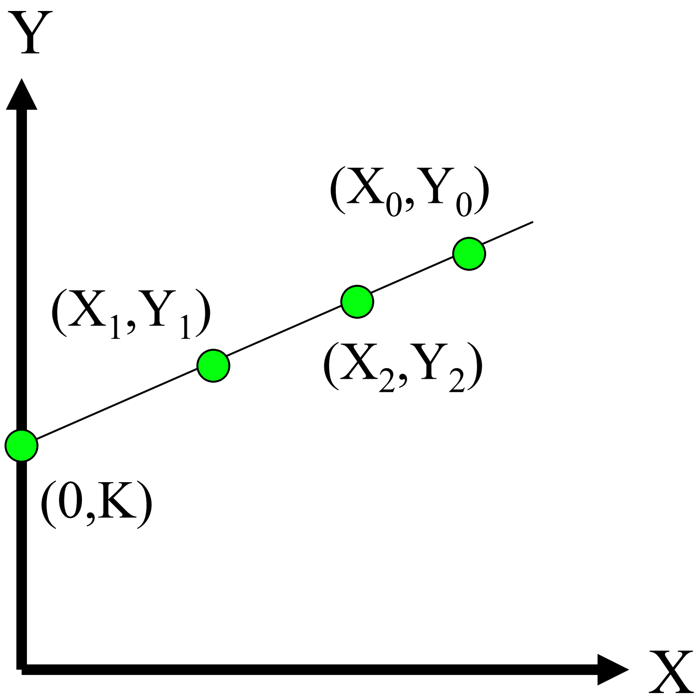
3 out of 3

- Give (X_0,Y_0) to Alice
- Give (X_1,Y_1) to Bob
- Give (X_2,Y_2) to Charlie
- 3 points determine a parabola
- Alice, Bob **and** Charlie must cooperate to find secret S
- A "3 out of 3" scheme
- Can you make a "3 out of 4"?

Secret Sharing Example

- **Key escrow** — required that your key be stored somewhere
- Key can be used with court order
- But you don't trust FBI to store keys
- We can use secret sharing
  - Say, three different government agencies
  - Two must cooperate to recover the key

Secret Sharing Example



- Your symmetric key is K
- Point (X_0,Y_0) to FBI
- Point (X_1,Y_1) to DoJ
- Point (X_2,Y_2) to DoC
- To recover your key K, two of the three agencies must cooperate
- No one agency can get K

Lagrange Interpolation Formula

Polynomial: $f(x) = s + a_1x + \ldots + a_{t-1}x^{t-1}$

Point: n pairs (x_i, y_i)'s

At least t pairs can use Lagrange interpolation formula to reconstruct unique polynomial as follows:

$$f(x) = \sum_{i=1}^{t} y_i \prod_{1 \le j \le t,\, j \ne i} \frac{x - x_j}{x_i - x_j}$$

$$f(0) = \sum_{i=1}^{t} y_i \prod_{1 \le j \le t,\, j \ne i} \frac{0 - x_j}{x_i - x_j} = s$$

Example 1:

Polynomial: $f(x) = s + a_1x + a_2x^2$

Point: 3 points (1, 4), (2, 5), (3, 10)

Use Lagrange interpolation formula to reconstruct the polynomial.

$$f(x) = \sum_{i=1}^{t} y_i \prod_{1 \le j \le t,\, j \ne i} \frac{x - x_j}{x_i - x_j}$$

$$\begin{aligned} f(x) &= 4\frac{(x-2)(x-3)}{(1-2)(1-3)} + 5\frac{(x-1)(x-3)}{(2-1)(2-3)} + 10\frac{(x-1)(x-2)}{(3-1)(3-2)} \\ &= 2(x^2 - 5x + 6) - 5(x^2 - 4x + 3) + 5(x^2 - 3x + 2) \\ &= 2x^2 - 5x + 7 \end{aligned}$$

Example 2:

Polynomial: $f(x) = s + a_1x + a_2x^2$

Point: 3 points (0, -9), (1, 2), (2, 21)

Use Lagrange interpolation formula to reconstruct the polynomial.

$$f(x) = \sum_{i=1}^{t} y_i \prod_{1 \le j \le t,\, j \ne i} \frac{x - x_j}{x_i - x_j}$$

$$f(x) =$$

Example 2:

Polynomial: $f(x) = s + a_1x + a_2x^2$

Point: 3 points (0, -9), (1, 2), (2, 21)

Use Lagrange interpolation formula to reconstruct the polynomial.

$$f(x) = \sum_{i=1}^{t} y_i \prod_{1 \le j \le t,\, j \ne i} \frac{x - x_j}{x_i - x_j}$$

$$f(x) = (-9)\frac{(x-1)(x-2)}{(0-1)(0-2)} + 2\frac{(x-0)(x-2)}{(1-0)(1-2)} + 21\frac{(x-0)(x-1)}{(2-0)(2-1)}$$

$$= (-9)(x^2 - 3x + 2)/9 - 2(x^2 - 2x) + 6(x^2 - x)/2$$

$$= 4x^2 + 7x - 9$$

Random Numbers in Cryptography

Random Numbers

- Random numbers used to generate **keys**
  - Symmetric keys
  - RSA: Prime numbers
  - Diffie Hellman: secret values
- Random numbers used for nonces
  - Sometimes a sequence is OK
  - But sometimes nonces must be random
- Random numbers also used in simulations, statistics, etc., where numbers only need to be "statistically" random

Random Numbers

- Cryptographic random numbers must be statistically random and **unpredictable**
- Suppose server generates symmetric keys
  - Alice: K_A
  - Bob: K_B
  - Charlie: K_C
  - Dave: K_D
- Spse Alice, Bob and Charlie don't like Dave
- Alice, Bob and Charlie working together must **not** be able to determine K_D

Bad Random Number Example

- Online version of Texas Hold 'em Poker
  - ASF Software, Inc.

- Random numbers used to shuffle the deck
- Program did not produce a random shuffle
- Could determine the shuffle in real time!

Card Shuffle

- There are $52! > 2^{225}$ possible shuffles
- The poker program used "random" 32-bit integer to determine the shuffle
  - Only 2^{32} distinct shuffles could occur
- Used Pascal pseudo-random number generator (PRNG): Randomize()
- Seed value for PRNG was function of number of milliseconds since midnight
- Less than 2^{27} milliseconds in a day
  - Therefore, less than 2^{27} possible shuffles

Poker Example

- Poker program is an extreme example
  - But common PRNGs are predictable
  - Only a question of how many outputs must be observed before determining the sequence
- Crypto random sequence is not predictable
  - For example, keystream from RC4 cipher
- But "seed" (or key) selection is still an issue!
- How to generate initial **random** values?
  - Applies to both keys and seeds

Randomness

- True randomness is hard to define
- **Entropy** is a measure of randomness
- Good sources of "true" randomness
  - Radioactive decay — though radioactive computers are not too popular
  - Hardware devices — many good ones on the market
  - Lava lamp — relies on chaotic behavior

Information Hiding

Information Hiding

- Digital Watermarks
  - Example: Add "invisible" identifier to data
  - Defense against music or software piracy
- Steganography
  - Secret communication channel
  - A kind of **covert channel**
  - Example: Hide data in image or music file

Watermark

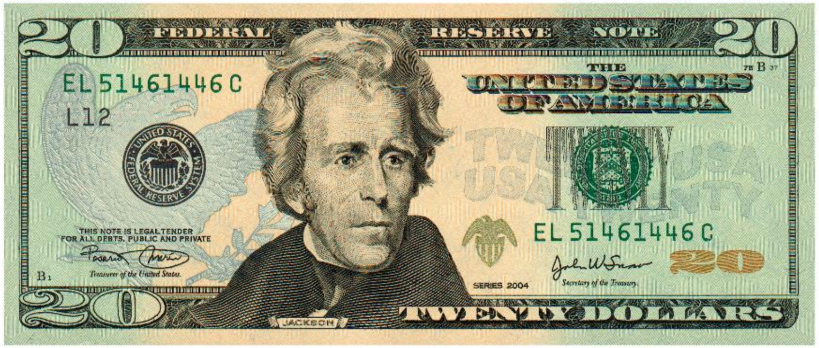
- Add a "mark" to data
- Several types of watermarks
  - Invisible — Not obvious the mark exists
  - Visible — Such as **TOP SECRET**
  - Robust — Readable even if attacked
  - Fragile — Mark destroyed if attacked

Watermark

- Add **robust invisible** mark to digital music
  - If pirated music appears on Internet, can trace it back to original source
- Add **fragile invisible** mark to audio file
  - If watermark is unreadable, recipient knows that audio has been tampered (integrity)
- Combinations of several types are sometimes used
  - E.g., visible plus robust invisible watermarks

Watermark Example (1)

- US currency includes watermark



- Image embedded in paper on rhs
  - Hold bill to light to see embedded info

Watermark Example (2)

- Add **invisible** watermark to photo print
- It is claimed that 1 square inch can contain enough info to reconstruct entire photo
- If photo is damaged, watermark can be read from an undamaged section and entire photo can be reconstructed!

Steganography

- According to Herodotus (Greece 440BC)
  - Shaved slave's head
  - Wrote message on head
  - Let hair grow back
  - Send slave to deliver message
  - Shave slave's head to expose message (warning of Persian invasion)
- Historically, steganography has been used more than cryptography!

Images and Steganography

- Images use 24 bits for color: RGB
  - 8 bits for red, 8 for green, 8 for blue
- For example
  - **0x7E 0x52 0x90** is this color
  - **0xFE 0x52 0x90** is this color
- While
  - **0xAB 0x33 0xF0** is this color
  - **0xAB 0x33 0xF1** is this color
- Low-order bits are unimportant!

Images and Stego

- Given an uncompressed image file
  - For example, BMP format
- Then we can insert any information into low-order RGB bits
- Since low-order RGB bits don't matter, result will be "invisible" to human eye
- But a computer program can "see" the bits

Stego Example 1

- Left side: plain Alice image
- Right side: *Alice* with entire *Alice in Wonderland* (pdf) "hidden" in image

Non-Stego Example

- Walrus.html in web browser

"The time has come," the Walrus said,
"To talk of many things:
Of shoes and ships and sealing wax
Of cabbages and kings
And why the sea is boiling hot
And whether pigs have wings."

- View source

```
<font color="#000000">"The time has come," the Walrus said,</font><br>
<font color="#000000">"To talk of many things:</font><br>
<font color="#000000">Of shoes and ships and sealing wax</font><br>
<font color="#000000">Of cabbages and kings</font><br>
<font color="#000000">And why the sea is boiling hot</font><br>
<font color="#000000">And whether pigs have wings."</font><br>
```

Stego Example 2

- stegoWalrus.html in web browser

"The time has come," the Walrus said,
"To talk of many things:
Of shoes and ships and sealing wax
Of cabbages and kings
And why the sea is boiling hot
And whether pigs have wings."

- View source

```
<font color="#010100">"The time has come," the Walrus said,</font><br>
<font color="#000100">"To talk of many things:</font><br>
<font color="#010100">Of shoes and ships and sealing wax</font><br>
<font color="#000101">Of cabbages and kings</font><br>
<font color="#000000">And why the sea is boiling hot</font><br>
<font color="#010001">And whether pigs have wings."</font><br>
```

- "Hidden" message: **110 010 110 011 000 101**

Steganography

- Some formats (jpg, gif, wav, etc.) are more difficult (than html) for humans to read
- Easy to hide information in **unimportant bits**
- Easy to **destroy** or remove info stored in unimportant bits!
- To be robust, information must be stored in **important bits**
- But stored information must not damage data!
- Collusion attacks also a major concern
- Robust steganography is trickier than it seems

Information Hiding The Bottom Line

- Surprisingly difficult to hide digital information: "obvious" approach **not** robust
  - **Stirmark** makes most watermarks in jpg images unreadable — **without** damaging the image
  - Watermarking is very active research area
- If information hiding is suspected
  - Attacker can probably make information/watermark unreadable
  - Attacker may be able to read the information, given the original document (image, audio, etc.)