Crypto Hash

Hash Function Motivation

- Suppose Alice signs M
  - Alice sends M and $S = [M]_{Alice}$ to Bob
  - Bob verifies that $M = \{S\}_{Alice}$
  - Can Alice just send S?
- If M is big, $[M]_{Alice}$ costly to **compute** & **send**
- Suppose instead, Alice signs h(M), where h(M) is a much smaller "fingerprint" of M
  - Alice sends M and $S = [h(M)]_{Alice}$ to Bob
  - Bob verifies that $h(M) = \{S\}_{Alice}$

Hash Function Motivation

- So, Alice signs $h(M)$
  - That is, Alice computes $S = [h(M)]_{Alice}$
  - Alice then sends (M, S) to Bob
  - Bob verifies that $h(M) = \{S\}_{Alice}$
- What properties must $h(M)$ satisfy?
  - Suppose Trudy finds M' so that $h(M) = h(M')$
  - Then Trudy can replace (M, S) with (M', S)
- Does Bob detect this tampering?
  - No, since $h(M') = h(M) = \{S\}_{Alice}$

Crypto Hash Function

- Crypto hash function $h(x)$ must provide
  - **Compression** — output length is small
  - **Efficiency** — $h(x)$ easy to compute 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 **any**

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
  - We find $N = 253$

Birthday Problem

- How many people must be in a room before probability is ≥ 1/2 that any 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
  - And there are 365 possible birthdays

Of Hashes and Birthdays

- If $h(x)$ is N bits, then 2^N different hash values are possible
- So, if you hash about $\text{sqrt}(2^N) = 2^{N/2}$ values then you expect to find a collision
- **Implication?** "Exhaustive search" attack…
  - Secure N-bit hash requires $2^{N/2}$ work to "break"
  - Recall that secure N-bit symmetric cipher has work factor of 2^{N-1}
- Hash output length vs cipher key length?

Non-crypto Hash (1)

- Data $X = (X_1,X_2,X_3,\ldots,X_n)$, each X_i is a byte
- Define $h(X) = (X_1+X_2+X_3+\ldots+X_n) \bmod 256$
- Is this a secure cryptographic hash?
- Example: $X = (10101010, 00001111)$
- Hash is $h(X) = 10111001$
- If $Y = (00001111, 10101010)$ then $h(X) = h(Y)$
- 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 defined as

$$h(X) = (nX_1 + (n-1)X_2 + (n-2)X_3 + \ldots + 2\cdot X_{n-1} + X_n) \bmod 256$$

- Is this a secure cryptographic hash?
- Note that

$$h(10101010, 00001111) \neq h(00001111, 10101010)$$

- But hash of $(00000001, 00001111)$ is same as hash of $(00000000, 00010001)$
- Not "secure", but this hash is used in the (non-crypto) application rsync

Non-crypto Hash (3)

- Cyclic Redundancy Check (CRC)
- Essentially, CRC is the remainder in a long division calculation
- Good for detecting burst **errors**
  - Such random errors unlikely to yield a collision
- But easy to **construct** collisions
  - In crypto, Trudy is the enemy, not "random"
- CRC has been mistakenly used where crypto integrity check is required (*e.g.*, WEP)

Popular Crypto Hashes

- **MD5** — invented by Rivest (of course…)
  - 128 bit output
  - MD5 collisions easy to find, so it's broken
- **SHA-2** / **SHA-3** — A U.S. government standard
  - 224, 256, 384, 512 bit output
- 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 the message M with key K using a "hashed MAC" or **HMAC**
- HMAC is a **keyed** hash
  - Why would we need a key?
- How to compute HMAC?
- Two obvious choices: h(K,M) and h(M,K)
- Which is better?

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 $h(K,M') = h(K,M,X) = F(h(K,M),X)$
  - Attacker can compute HMAC of M' without K
- Is $h(M,K)$ better?
  - Yes, but… if $h(M') = h(M)$ then we might have $h(M,K)=F(h(M),K)=F(h(M'),K)=h(M',K)$

Still a collision

HMAC

- H(M, K)
- If we are concatenating M + K, then:

- What happens if M = "Hello" and K = "!!"?
- The result would look like: "Hello!!"

- What if M = "Hello!" and K = "!"
- The result would look like: "Hello!!"...again.

- That's something that we want to avoid...

Correct 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
- Also, many, many clever/surprising uses…

Online Bids

- 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
- A possible solution?
  - Alice, Bob, Charlie submit **hashes** h(A), h(B), h(C)
  - All hashes received and posted online
  - Then bids A, B, and C submitted and revealed
- Hashes don't reveal bids (one way)
- Can't change bid after hash sent (collision)

Random Numbers in Cryptography

Random Numbers

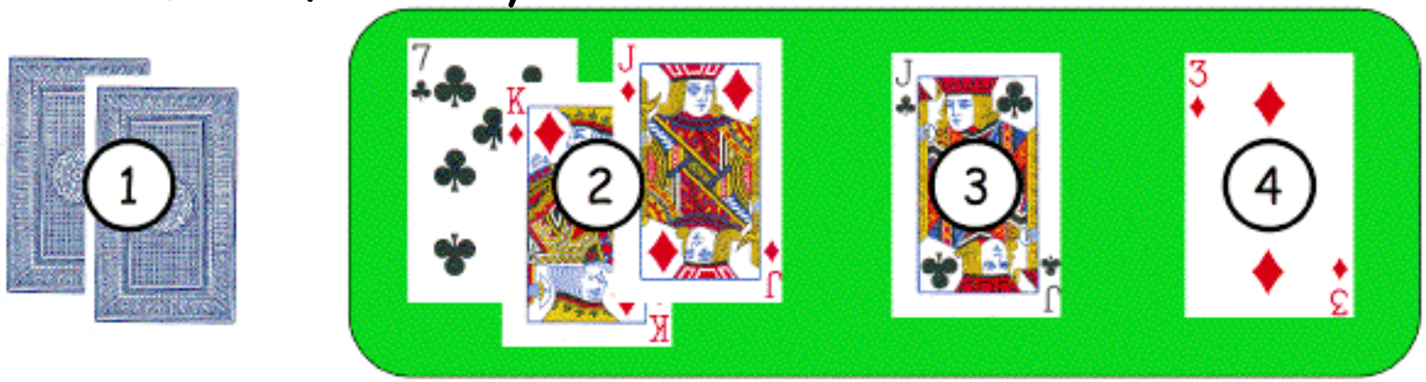
- Random numbers used to generate **keys**
  - Symmetric keys
  - RSA: Prime numbers
  - Diffie Hellman: secret values
- Random numbers used for nonces
  - Sometimes a sequence is OK
  - But sometimes nonces must be random
- Random numbers also used in simulations, statistics, etc.
  - In such apps,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

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
- Alice, Bob, and Charlie don't like Dave…
- Alice, Bob, and Charlie, working together, must not be able to determine K_D

Non-random Random Numbers

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



Player's hand

Community cards in center of the table

- Random numbers used to shuffle the deck
- Program did not produce a random shuffle
- A serious problem, or not?

Card Shuffle

- There are $52! > 2^{225}$ possible shuffles
- The poker program used "random" 32-bit integer to determine the shuffle
  - So, only 2^{32} distinct shuffles could occur
- Code 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
  - So, less than 2^{27} possible shuffles

Card Shuffle

- Seed based on milliseconds since midnight
- PRNG re-seeded with each shuffle
- By synchronizing clock with server, number of shuffles that need to be tested $< 2^{18}$
- Could then test all 2^{18} in real time
  - Test each possible shuffle against "up" cards
- Attacker knows **every card** after the first of five rounds of betting!

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 sequences not predictable
  - For example, keystream from RC4 cipher
  - But "seed" (or key) selection is still an issue!
- How to generate initial **random** values?
  - Keys (and, in some cases, seed values)

What is Random?

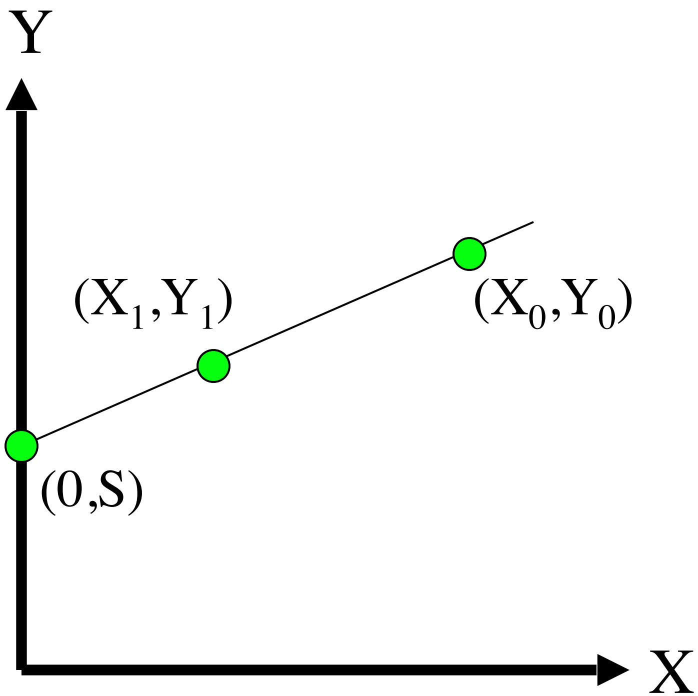
- True "random" hard to even define
- **Entropy** is a measure of randomness
- Good sources of "true" randomness
  - Radioactive decay — but, radioactive computers are not too popular
  - Hardware devices — many good ones on the market
  - Lava lamp — relies on chaotic behavior

Randomness

- Sources of randomness via software
  - Software is supposed to be deterministic
  - So, must rely on external "random" events
  - Mouse movements, keyboard dynamics, network activity, etc., etc.
- Can get **quality** random bits by such methods
- But **quantity** of bits is very limited
- Bottom line: "The use of pseudo-random processes to generate secret quantities can result in pseudo-security"

Secret Sharing

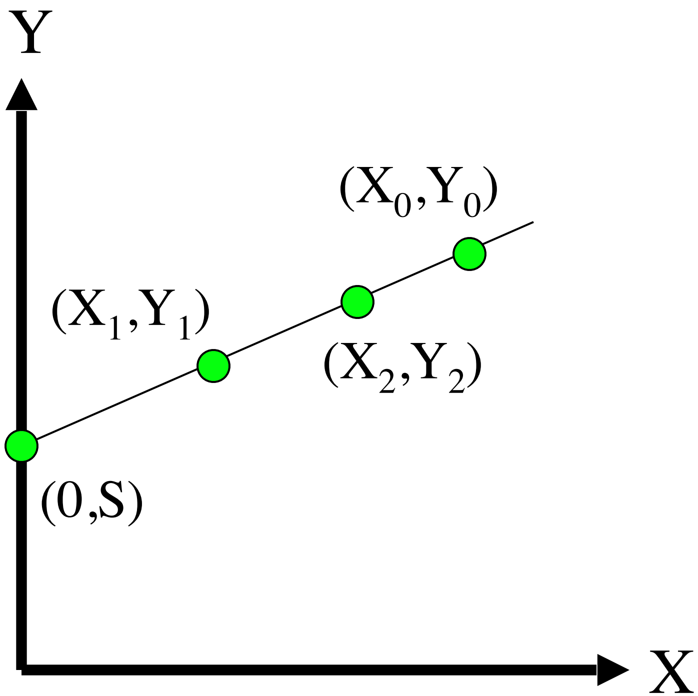
Shamir's Secret Sharing



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 $m \leq n$

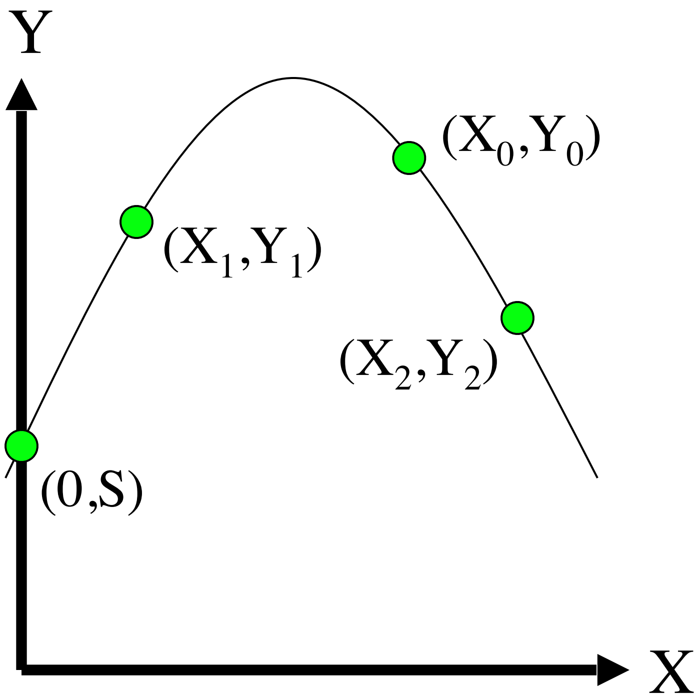
Shamir's Secret Sharing



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** can cooperate to find secret S
- No **one** can determine S
- A "2 out of 3" scheme

Shamir's Secret Sharing



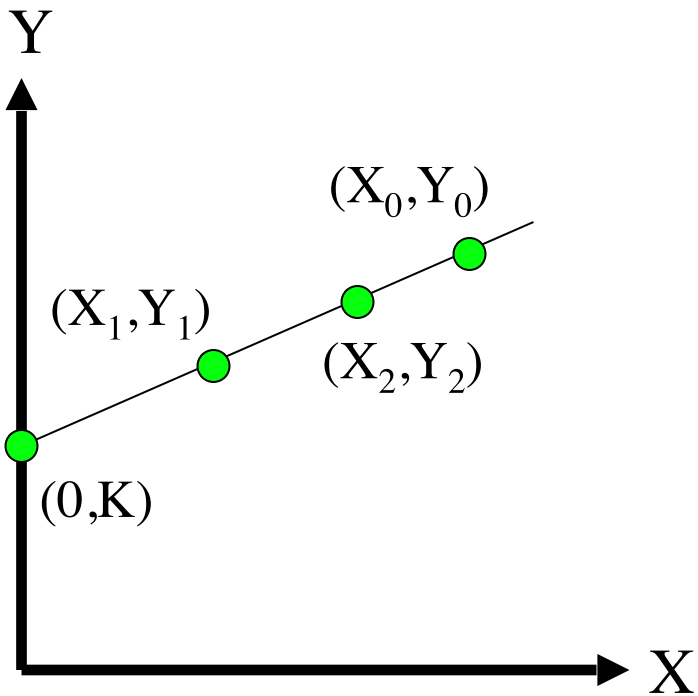
3 out of 3

- Give (X_0,Y_0) to Alice
- Give (X_1,Y_1) to Bob
- Give (X_2,Y_2) to Charlie
- 3 pts determine parabola
- Alice, Bob, **and** Charlie must cooperate to find S
- A "3 out of 3" scheme
- What about "3 out of 4"?

Secret Sharing Use?

- **Key escrow** — suppose it's required that your key be stored somewhere
- Key can be "recovered" with court order
- But you don't trust FBI to store your 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

Information Hiding

Information Hiding

- Digital Watermarks
  - Example: Add "invisible" info to data
  - Defense against music/software piracy
- Steganography
  - "Secret" communication channel
  - Similar to a **covert channel** (more later)
  - Example: Hide data in an image file

Watermark

- Add a "mark" to data
- Visibility (or not) of watermarks
  - Invisible — Watermark is not obvious
  - Visible — Such as **TOP SECRET**
- "Strength" of watermarks
  - Robust — Readable even if attacked
  - Fragile — Damaged if attacked

Watermark Examples

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

Watermark Example (1)

- Non-digital watermark: U.S. currency

  - Hold bill to light to see embedded info

Watermark Example (2)

- Add **invisible** watermark to photo
- Claim is that 1 inch^2 contains enough info to reconstruct entire photo
- If photo is damaged, watermark can be used to reconstruct it!

Steganography

- According to Herodotus (Greece 440 BC)
  - Shaved slave's head
  - Wrote message on head
  - Let hair grow back
  - Send slave to deliver message
  - Shave slave's head to expose a message warning of Persian invasion
- Historically, steganography used by military more often 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 don't matter…

Images and Stego

- Given an uncompressed image file…
  - For example, BMP format
- …we can insert information into low-order RGB bits
- Since low-order RGB bits don't matter, changes will be "invisible" to human eye
  - But, 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 the 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" reveals:

```
<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" reveals:

```
<font color=#000101>"The time has come," the Walrus said,</font><br>
<font color=#000100>"To talk of many things: </font><br>
<font color=#010000>Of shoes and ships and sealing wax </font><br>
<font color=#010000>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: **011 010 100 100 000 101**

Steganography

- Some formats (e.g., image files) are more difficult than html for **humans** to read
  - But easy for computer programs to read…
- Easy to hide info in **unimportant bits**
- Easy to damage info in unimportant bits
- To be **robust**, must use **important bits**
  - But stored info must not damage data
  - Collusion attacks are also a concern
- Robust steganography is tricky!