Chapter 5: Hash Functions++

Hash Function Motivation

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

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 $\geq 1/2$ 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 $\mathrm{sqrt}(2^N) = 2^{N/2}$ 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-1** — A U.S. government standard, inner workings similar to MD5
  - 160 bit output
- Many other hashes, but MD5 and SHA-1 are the 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

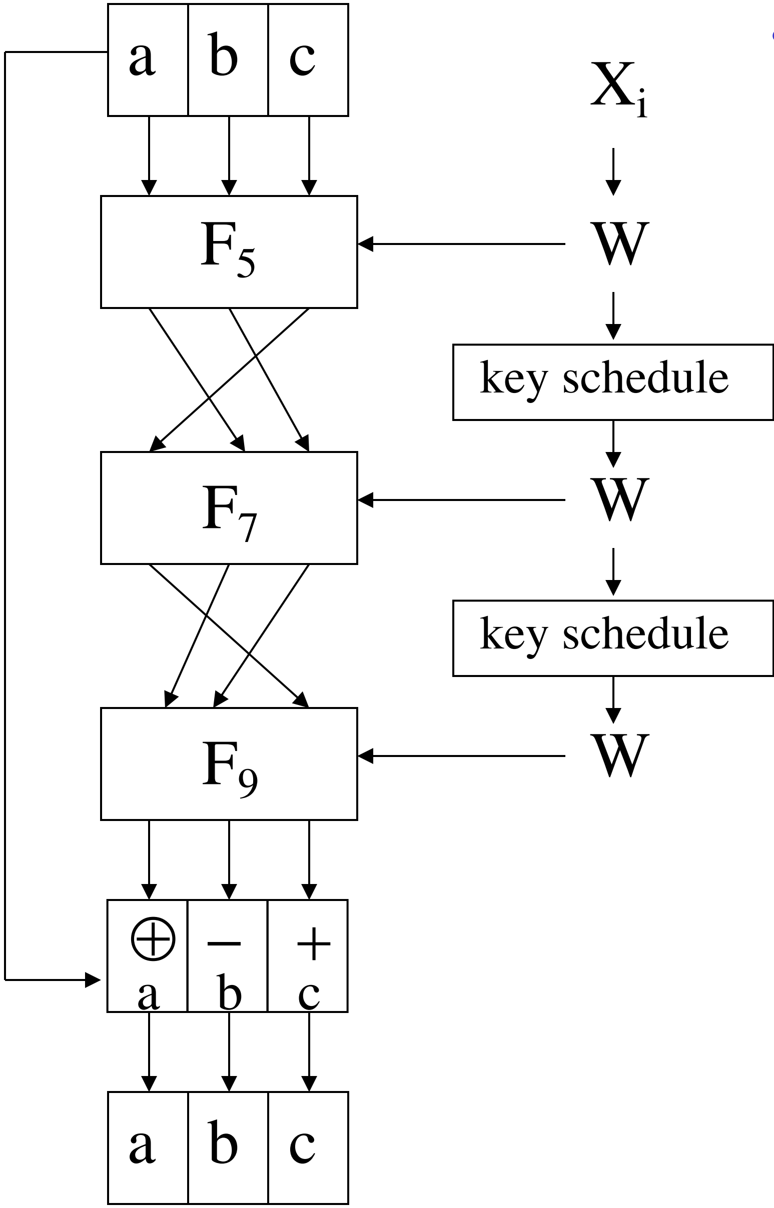
Tiger Hash

- "Fast and strong"
- Designed by Ross Anderson and Eli Biham — leading cryptographers
- Design criteria
  - Secure
  - Optimized for 64-bit processors
  - Easy replacement for MD5 or SHA-1

Tiger Hash

- Like MD5/SHA-1, input is divided into 512 bit blocks (padded)
- Unlike MD5/SHA-1, output is **192 bits** (three 64-bit words)
  - Truncate output if replacing MD5 or SHA-1
- Intermediate rounds are all 192 bits
- 4 S-boxes, each maps 8 bits to 64 bits
- A “key schedule” is used

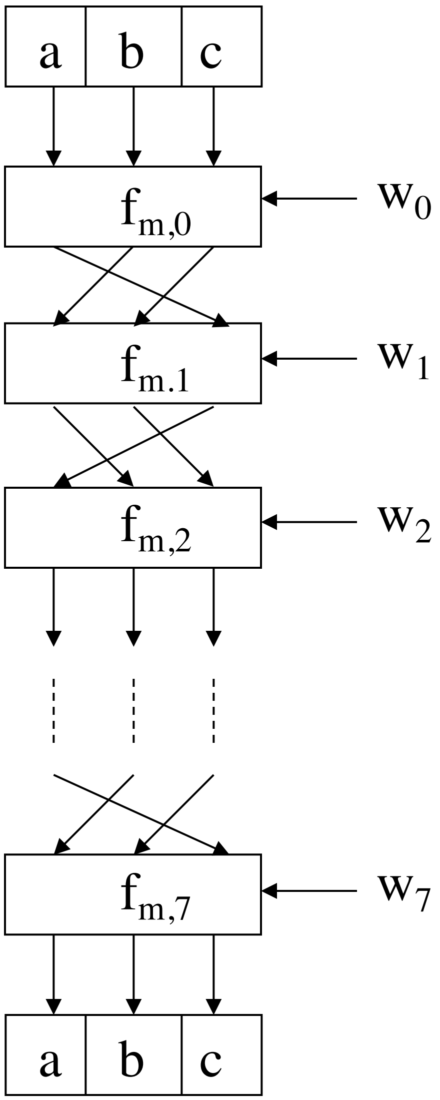
Tiger Outer Round



- Input is X
  - $X = (X_0, X_1, \ldots, X_{n-1})$
  - X is padded
  - Each X_i is 512 bits
- There are n iterations of diagram at left
  - One for each input block
- Initial (a,b,c) constants
- Final (a,b,c) is hash
- Looks like block cipher!

Tiger Inner Rounds

- Each F_m consists of precisely **8 rounds**
- 512 bit input W to F_m
  - $W=(w_0,w_1,\ldots,w_7)$
  - W is one of the input blocks X_i
- All lines are 64 bits
- The $f_{m,i}$ depend on the S-boxes (next slide)



Tiger Hash: One Round

- Each $f_{m,i}$ is a function of a,b,c,w_i and m
  - Input values of a,b,c from previous round
  - And w_i is 64-bit block of 512 bit W
  - Subscript m is multiplier
  - And $c = (c_0,c_1,\ldots,c_7)$
- Output of $f_{m,i}$ is
  - $c = c \oplus w_i$
  - $a = a - (S_0[c_0] \oplus S_1[c_2] \oplus S_2[c_4] \oplus S_3[c_6])$
  - $b = b + (S_3[c_1] \oplus S_2[c_3] \oplus S_1[c_5] \oplus S_0[c_7])$
  - $b = b * m$
- Each S_i is **S-box**: 8 bits mapped to 64 bits

Tiger Hash Key Schedule

- Input is X
  - $X=(x_0,x_1,\ldots,x_7)$
- Small change in X will produce large change in key schedule output

$$
\begin{aligned}
x_0 &= x_0 - (x_7 \oplus \text{0xA5A5A5A5A5A5A5A5})\\
x_1 &= x_1 \oplus x_0\\
x_2 &= x_2 + x_1\\
x_3 &= x_3 - (x_2 \oplus ((\sim x_1) << 19))\\
x_4 &= x_4 \oplus x_3\\
x_5 &= x_5 + x_4\\
x_6 &= x_6 - (x_5 \oplus ((\sim x_4) >> 23))\\
x_7 &= x_7 \oplus x_6\\
x_0 &= x_0 + x_7\\
x_1 &= x_1 - (x_0 \oplus ((\sim x_7) << 19))\\
x_2 &= x_2 \oplus x_1\\
x_3 &= x_3 + x_2\\
x_4 &= x_4 - (x_3 \oplus ((\sim x_2) >> 23))\\
x_5 &= x_5 \oplus x_4\\
x_6 &= x_6 + x_5\\
x_7 &= x_7 - (x_6 \oplus \text{0x0123456789ABCDEF})
\end{aligned}
$$

Tiger Hash Summary (1)

- Hash and intermediate values are 192 bits
- 24 (inner) rounds
  - **S-boxes:** Claimed that each input bit affects a, b and c after 3 rounds
  - **Key schedule:** Small change in message affects many bits of intermediate hash values
  - **Multiply:** Designed to ensure that input to S-box in one round mixed into many S-boxes in next
- S-boxes, key schedule and multiply together designed to ensure strong avalanche effect

Tiger Hash Summary (2)

- Uses lots of ideas from block ciphers
  - S-boxes
  - Multiple rounds
  - Mixed mode arithmetic