Chapter 5
Hash Functions++

Non-crypto Hash (1)

- Data $X = (X_0,X_1,X_2,\ldots,X_{n-1})$, each X_i is a byte
- Spse $\text{hash}(X) = X_0+X_1+X_2+\ldots+X_{n-1}$
- 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 $(00000000,00010001)$
- Not one-way, 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 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)
  - 160 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

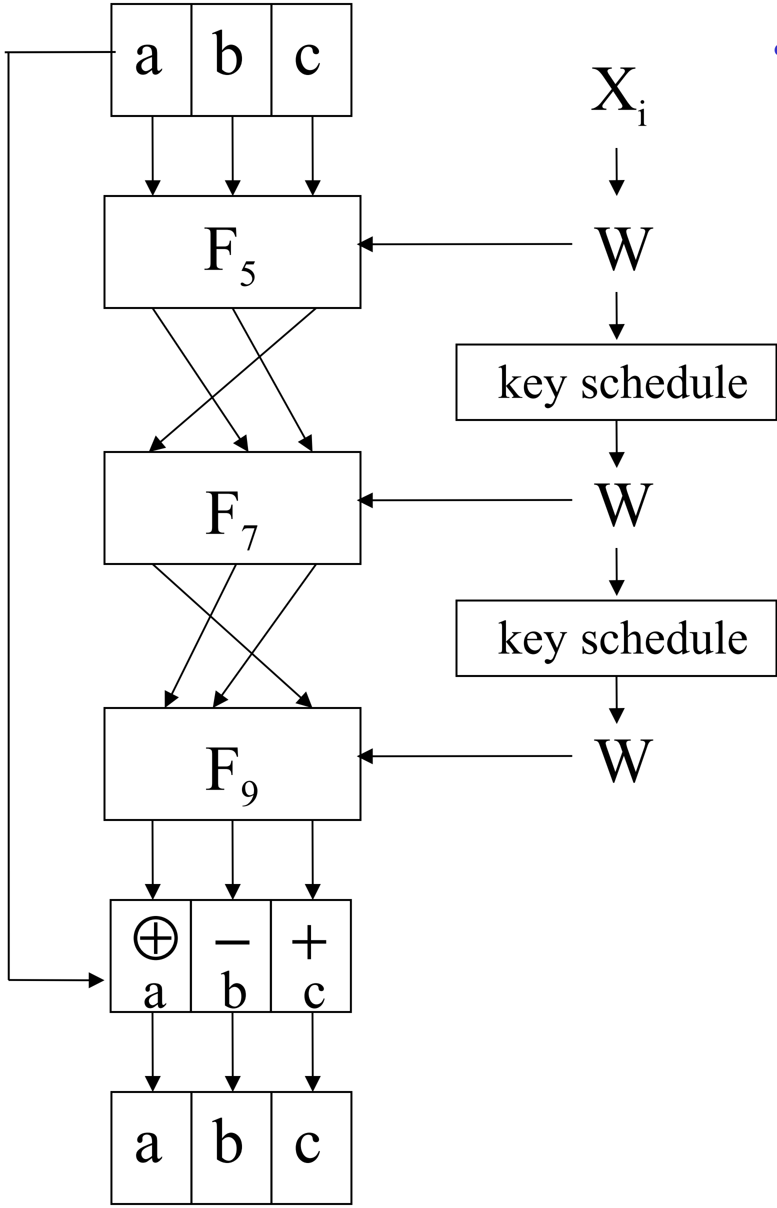
Tiger Hash

- "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 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

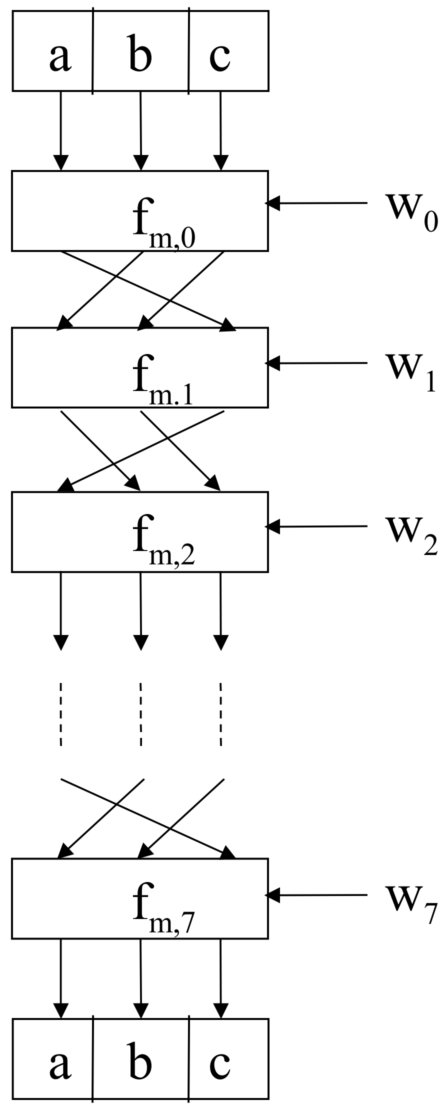
Tiger Outer Round



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

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

Tiger Hash Summary (1)

- Hash and intermediate values are 192 bits
- 24 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 insure that input to S-box in one round mixed into many S-boxes in next
- S-boxes, key schedule and multiply together designed to insure strong **avalanche** effect

Tiger Hash Summary (2)

- Uses lots of ideas from block ciphers
  - S-boxes
  - Multiple rounds
  - Mixed mode arithmetic
- At a higher level, Tiger employs
  - Confusion
  - Diffusion

HMAC

- Can compute a MAC of the message 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