Next Quarter

- 2-4 weeks to cover 16, 17, 18 and results on boolean functions.
- Rest on major reports:
  - Full Linear cryptanalysis of DES.
  - Full Differential cryptanalysis of DES.
  - Full Linear and differential cryptanalysis of FEAL.
  - Intro Algebraic cryptanalysis (including SFLASH) – John.
  - An algebraic cryptanalysis.
  - Dobbertin’s attack on MD4.
  - Chinese (Wang et. al) attack on SHA-1.
- Other topics (final quarter?)
  - Full factoring attack.
  - Full Elliptic Curve crypto selection, attacks, etc (3 weeks).
  - Full Discrete Log attack.
  - Full Re-estimation attack.
  - Random number analysis.
  - NIST Hash analysis.
  - Full Stream cipher analysis.

Cryptanalysis

Lecture Block 5: Cryptographic Hashes

John Manferdelli
jmanfer@microsoft.com
JohnManferdelli@hotmail.com

Cryptographic Hashes

- A cryptographic hash (“CH”) is a “one way function,” h, from all binary strings (of arbitrary length) into a fixed block of size n (called the size of the hash) with the following properties:

1. Computing h is relatively cheap.
2. Given y=h(x) it is infeasible to calculate a x’ $\neq$x such that y=h(x’). (“One way,” “non-invertibility” or “pre-image” resistance). Functions satisfying this condition are called One Way Hash Functions (OWHF)
3. Given u, it is infeasible to find w such that h(u)=h(w). (weak collision resistance, 2nd pre-image resistance).
4. It is infeasible to find u, w such that h(u)=h(w). (strong collision resistance). Note 3$\rightarrow$2. Functions satisfying this condition are called Collision Resistant Functions (CRFs).

Cryptographic Hashes

- h must be compressive (otherwise copy of original binary string satisfies requirement)
- Just like symmetric ciphers ratio of work factor for computation of hash vs work factor to break hash should be very high.
- Adversary has complete information on computing hash and (obviously) can compute as many hashes from the target as she wants.

Observations on Cryptographic Hashes

- Hashes are a strong "checksum"
- OWHF and CRF conditions make CHs satisfy many of the properties of "random functions"
  - Small changes should create large changes (otherwise the pre-image of near neighbors are near neighbors making collisions easy to find)
  - Small input changes should be statistically unrelated (uncorrelated) to changes in a subset of the hash bits
  - Analysis of CHs very similar to Symmetric Cipher techniques

Popular practical cryptographic hashes

- MD4, MD5 (now "broken")
- SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 (last 4 are "SHA-2")
- RIPEMD

Observations

- Collision Resistance → 2nd pre-image resistance
- Let $f(x) = x^2 - 1 \pmod{p}$.
  - $f(x)$ acts like a random function but is not a OWHF since square roots are easy to calculate mod p.
- Let $f(x) = x^2 \pmod{pq}$.
  - $f(x)$ is a OWHF but is neither collision nor 2nd pre-image resistant
- If either $h_1(x)$ or $h_2(x)$ is a CRHF so is $h(x) = h_1(x) \| h_2(x)$
- MDC+signature & MAC+unknown Key require all three properties
- Ideal Work Factors:

Type	**Work**	**Property**
OWHF	2^n	Pre-image 2nd Pre-image
CRHF	$2^{n/2}$	Collision
MAC	2^t	Key recovery, computational resistance

What are Hash Functions Good for?

- Modification Detection Codes (MDCs): This is a strong checksum (integrity check). Sometimes called “unkeyed” hashes.
- Message Authentication Code (MACs): If shared secret is part of the hash, two parties can determine authenticated integrity with CHs. Called “keyed hashes”.
- Message Digests (MDs): Encrypting (with private key) the CH of a message (its MD) acts as a certification that the message was “approved” by possessor of private key. This is called a Digital Signature. [Note: you could “sign’ the whole message rather than the hash but this would take oodles of time by comparison.]

What are Hash Functions Good for?

- Uniquely and securely identifies bit streams like programs. Hash is strong name for program.
- Entropy mixing: Since CHs are random functions into fixed size blocks with the properties of random functions, they are often used to “mix” biased input to produce a “seed” for a psuedo-random number generator.
- Password Protection: Store salted hash of password instead of password (Needham).
- Bit Committment

MACs using Hashes

- Prefix and suffix attacks
- $\mathrm{Hash}(k_1, \mathrm{Hash}(k_2, m))$
- $\mathrm{Hash}(k|p|m|k)$

- $\mathrm{HMAC}_K(x) = \text{SHA-1}(K \oplus \mathrm{opad} \,||\, \text{SHA-1}(K \oplus \mathrm{ipad}) || x)$

One-Way Functions

Hashes come from two basic classes of one-way functions

- Mathematical
  - Multiplication: $Z=X•Y$
  - Modular Exponentiation: $Z = Y^X$ (mod n) (Chaum vP Hash)
- Ad-hoc (Symmetric cipher-like constructions)
  - Custom Hash functions (MD4, SHA, MD5, RIPEMD)

Attacks on Cryptographic Hashes

- Birthday (Yuval) attacks
  - Probability of collision determined by “Birthday Paradox” calculation:
    - $(1- 1/n)\ (1- 2/n)\ \ldots\ (1-(k-1)/n)= (n!/k!)/n^k$
    - Probability of collision is >.5 when $k^2 > n$.
    - Need 2^{80} blocks for SHA.
    - $1+x \square e^x$, $\square_{i=1}^{i=k} (1-i/n) \square e^{-k(k-1)/(2n)}$
- Dobbertin Attacks on MD4
- Attacks on 2nd preimage
  - (Old) If you hash 2^t messages, the average work to find a 2nd primage is 2^{n-t}
  - (New) If you hash 2^t *blocks*, the average work to find a 2nd primage is $t2^{n/2+1}+ 2^{n-t+1}$ [Kelsey Schneier]
  - Appending length doesn’t help against 2nd pre-image attacks

Attacks on Hashes

- Selective and Existential Forgery
- Key Recovery for MAC
- Chaining
  - Meet in Middle
  - Fixed Point
- Padding
- Differential

Attacks on Cryptographic Hashes

- Berson (1992) using differential cryptanalysis on 1 round MD-5.
- Boer and Bosselaers (1993), Pseudo collision in MD5.
- Dobbertin (1996), Collisions in compression function. Attacks inspired RIPEMD proposal.
- Biham and Chen (2004), Collisions in SHA-0.
- Chabaud and Joux (2004), Collisions in SHA-0 .
- Wang, Feng, Lai, Yu, (2004), MD4, MD5, RIPEMD
- Wang et al, (2004, 2005), SHA-1
- SHA-1 has stood up best: best known theoretical attack (11/05) requires 2^{63} operations.

Birthday Attacks

- Probability of collision determined by “Birthday Paradox” calculation:
  - $(1-1/n)\ (1-2/n)\ \ldots\ (1-(k-1)/n) = (n!/k!)/n^k$
  - Probability of collision is $>.5$ when $k^2 > n$.
  - Need 2^{80} blocks for SHA.
  - $1+x$ □ e^x, □ $_{i=1}{}^{i=k}\ (1-i/n)$ □ $e^{-k(k-1)/(2n)}$

Chaum-vanHeijst-Pfitzmann Compression Function

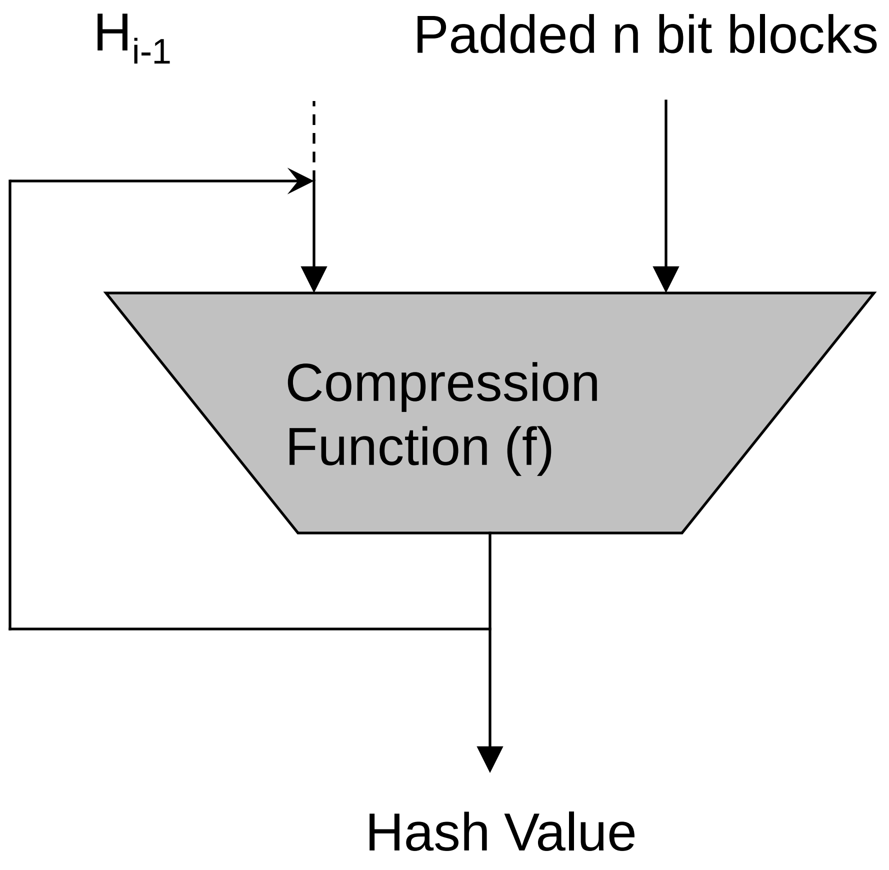
- Suppose p is prime, q=(p-1)/2 is prime, a is a primitive root in F_p, b is another primitive root so a^x=b (mod p) for some unknown x).
- g: $\{1,2,\ldots,q-1\}^2 \rightarrow \{1,2,\ldots,p-1\}$, q=(p-1)/2 by:
  - g(s, t) = $a^s b^t$ (mod p)
- Reduction to discrete log:

  Suppose g(s, t)= g(u, v) can be found. Then $a^s b^t$ (mod p)= $a^u b^v$ (mod p).

  So a^{s-u} (mod p)= b^{v-t} (mod p). Let b= a^x (mod p). Then (s-u)=x(y-t) (mod p-1).

  But p-1= 2q so we can solve for x, thus determining the discrete log of b.

Merkle/Damgard Construction



Input: $x=x_1||\ldots||x_t$
Input is usually padded

H_0= IV
H_i= $f(H_{i-1}, x_i)$
$h(x)$= $g(h_t)$

Graphic by Josh Benaloh

Proofs about compression function

- Theorem: If g: $\{0,1\}^m \rightarrow \{0,1\}^n$, for a sequence of n bit blocks, **x**= $x_1, x_2, \ldots, x_t$, we can define a hash function h: $\{0,1\}^* \rightarrow \{0,1\}^n$ by H_0= c, H_{i+1}= $g(H_i||x_i)$ with h(x)=H_t. h is collision resistant if g is.

  – Proof: Let **x**= $x_1, x_2, \ldots, x_t$ and **x'**= $x_1', x_2', \ldots, x_{t'}'$ be two strings with h(**x**)=h(**x'**) and let H_i, $H_{i'}$ be the intermediate values. Suppose there is an i<t: H_{t-i}=$H_{t'-i}'$ and H_{t-i-1}!=$H_{t'-i-1}'$. Then $g(H_{t-i-1}||x_i) = g(H_{t-i-1}'||x_i')$ so g is not collision resistant. Otherwise H_i=H_i' and either x_i=x_i', i<=t, in which case there is nothing to prove or some x_i!=x_i'(but then $g(H_i||x_i) = g(H_i'||x_i')$ and again g is not collision resistant) or $g(H_{t-1})$= $g(H_j'||x_j')$, j>t and again g is not collision resistant.

Technique for CHs from Block Ciphers

Let input be x= $x_1 \| x_2 \| \ldots \| x_t$ where each x_i is n bits long. Let g be a function taking an n bit input to an m bit input. Let E(k, x) be a block cipher with m bit keyspace and n bit block. Let H_0= IV.

Construction 1: $H_i = E(g(H_{i-1}), x_i) \oplus H_{i-1}$

Construction 2: $H_i = E(x_i, H_{i-1}) \oplus H_{i-1}$

Construction 3: $H_i = E(g(H_{i-1}), x_i) \oplus x_i \oplus H_{i-1}$

Note: Because of collisions n should be >64. Ideally, m=n and g= id. DES with n= 64 is too small. AES with n=m=128 is better.

Nostradamus (“herding") attack

- Let h be a Merkle-Damgard hash with compression function f and initial value IV . Goal is to hash a prefix value (P) quickly by appending random suffixes (S).
- Procedure
  - Phase 1: Pick k, generate $K=2^k$ random d_{0i} from each pair of the values f(IV $||d_{i,i+1}$) and two messages $M_{0,j}$;$M_{1,j}$ which collide under f . Call this value $d_{1,j}$ this takes effort $2^{n/2}$ for each pair. Do this (colliding $d_{i,j}$; $d_{i+1,j}$ under $M_{i,j}$;$M_{i+1,j}$ to produce $d_{i,j+1}$ until you reach $d_{K,0}$). This is the diamond.
  - Publish y = w($d_{K,0}$) where w is the final transformation in the hash as the hash (i.e. - claim y = h(P||S).

Nostradamus (“herding") attack

- The cost of phase 1 is $(2^k - 1)2^{n/2}$.
- In phase 2, guess S’ and compute T = f(IV||P||S’).
- Keep guessing until T is one of the d_{ij}. Once you get a collision, follow a path through the M_{ij} to $d_{K,0}$, append these M_{ij} to P||S’ and apply w to get right hash.
- Total cost: $W= 2^{n-k-1} + 2^{n/2+k/2} + k2^{n/2+1}$. k=(n-5)/3 is a good choice. For 160 bit hash, k=52.

Multicollision (Joux)

- Iterative construction is vulnerable to multi-collision
- Suppose $M_1;M_1';M_2;M_2'; \ldots;M_t;M_t'$ all collide.

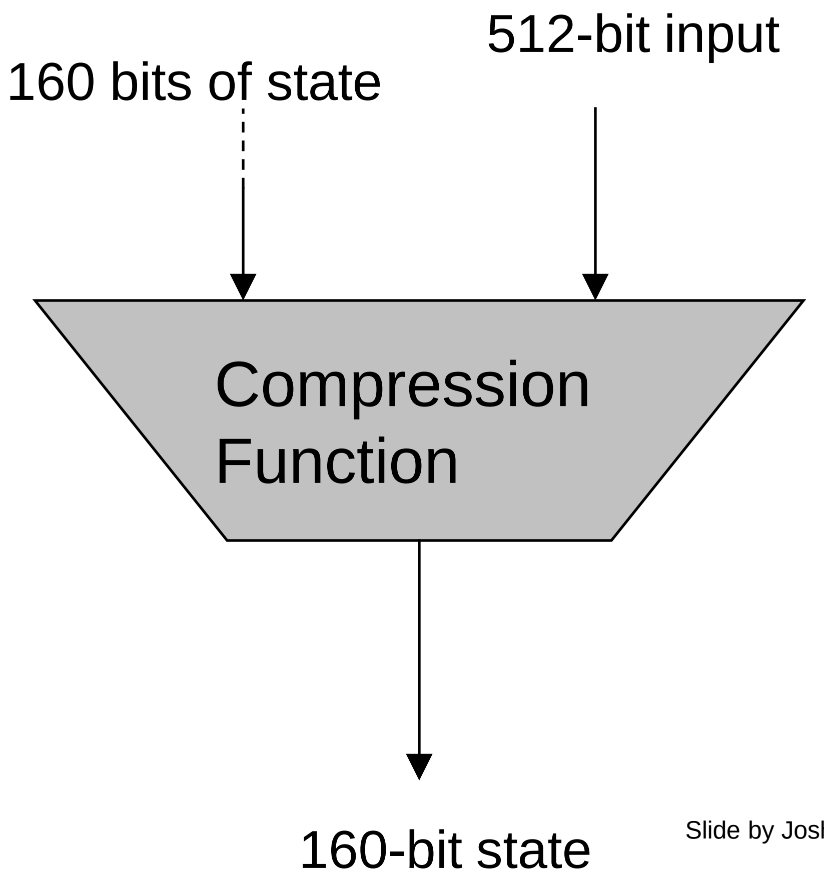
- From these we get 2^t collisions.

- If r people each have one of N possible birthdays, there is a greater than 50% chance of k collisions if $r>N^{\square}$, $\square$= k-1/k.

Random Oracle Model

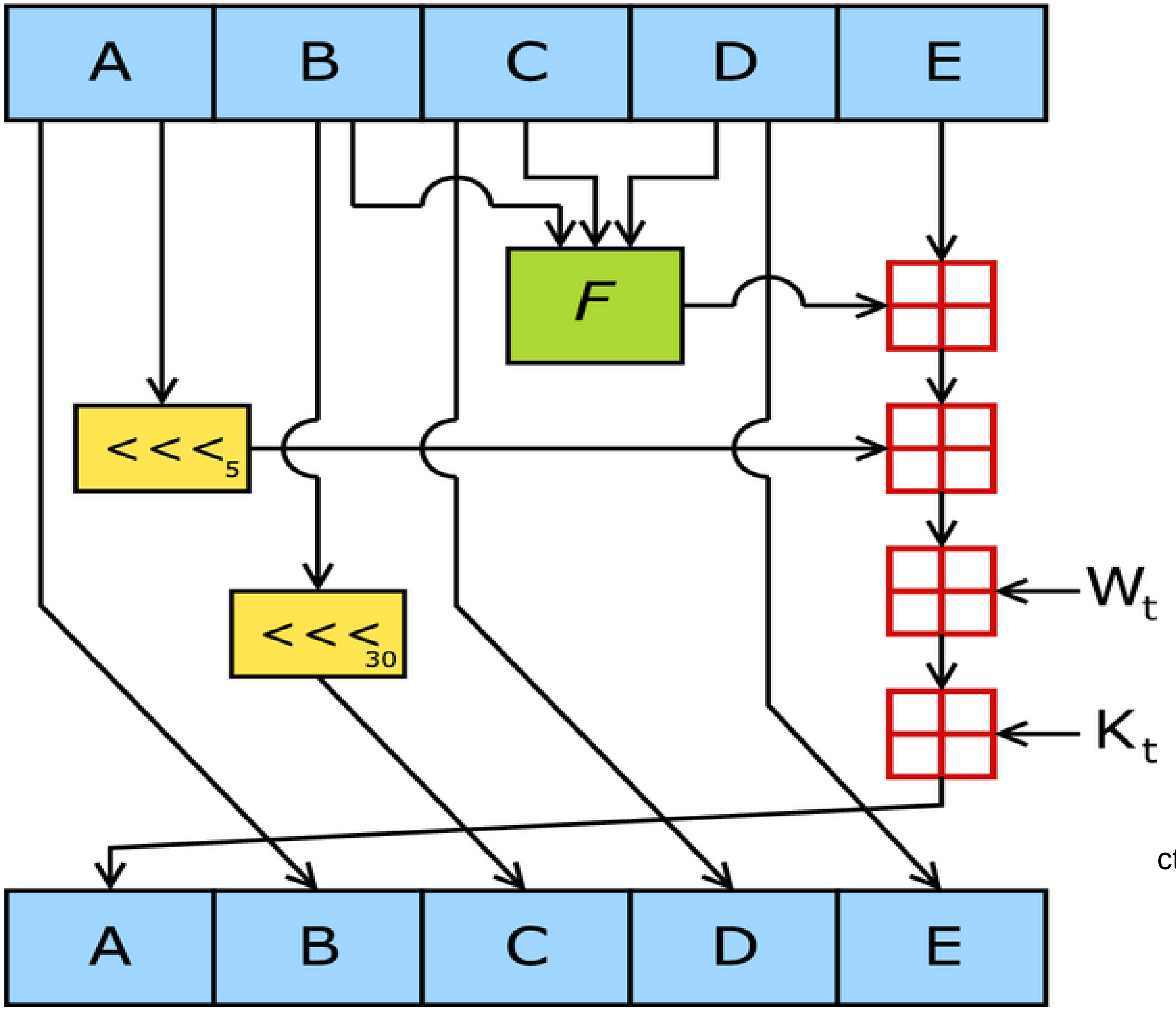
- Let *f be a OWF with trapdoor, (y_1, y_2) = (f(r); h(r)+m) is used as encryption.*
- An oracle with *l requests L.*
- *Pr(guess right) = P(r in L) + ½ P(r not in L).*
- *Set p = 1/*2 + *□*
- *□<= Pr(r in L).*
- Canetti, Goldreich, Halevi constructed cryptosystem that is secure in the Random Oracle model but any secure for any concrete hash.

A Cryptographic Hash: SHA-1



Slide by Josh Benaloh

A Cryptographic Hash: SHA-1



cture from Wikipedia

Padding

- Standard technique
  - Let last message block have k bits. If k=n, make a new block and set k= 0.
  - Append a 1 to last block leaving r=n-k-1 remaining bits in block.
  - If r>=64, append r-64 0s then append bit length of input expressed as 64 bit unsigned integer
  - If r<64, append n-r 0’s (to fill out block), append n-64 0’s at beginning of next block then append bit length of input expressed as 64 bit unsigned integer

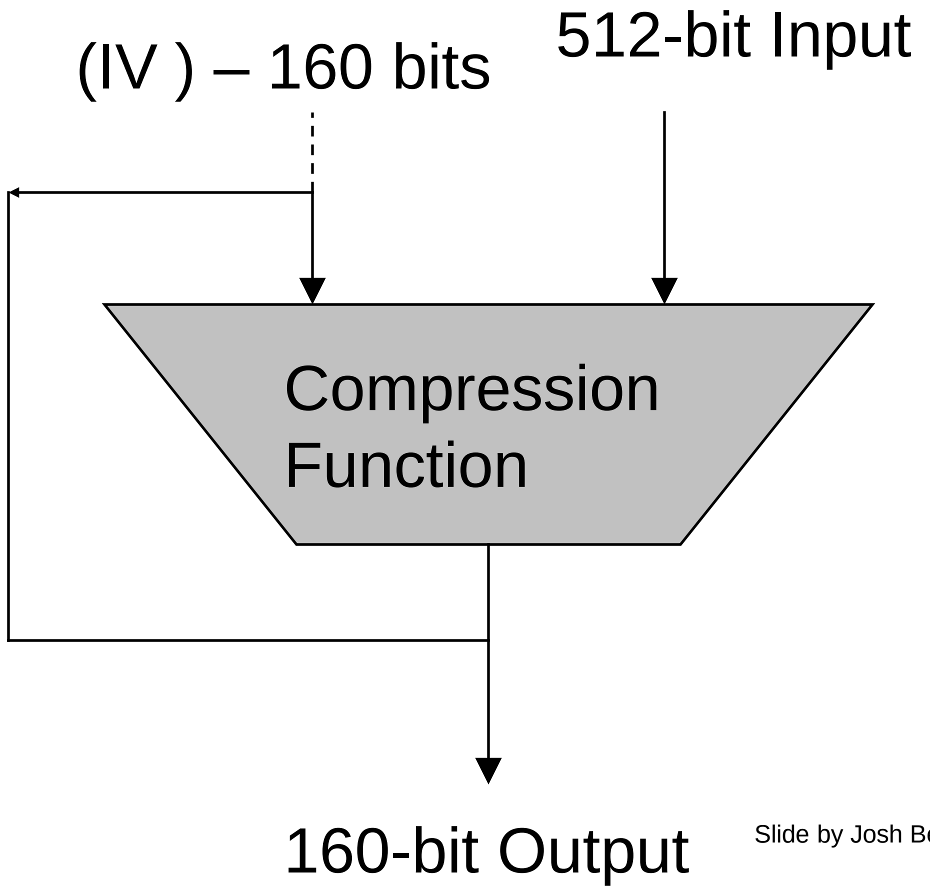
Winnowing and Chaffing (Rivest)

- Want to send 1001. Pick random stream (m_i) and embed message at positions (say) 3, 7, 8 14 MAC each packet (mm_i).
- Make sure MAC is correct only in message positions

Lai-Massey

- Assume the padding contains the length of the input string and that the input to the CH function, h, is at least two blocks long. Finding a 2nd pre-image for h with fixed IV requires 2^n operations iff finding a 2nd pre-image for the compression function, f, with arbitrarily chosen H_{i-1} requires 2^n operations where n is the number of bits of h’s output.

A Cryptographic Hash: SHA-1



Slide by Josh Benaloh

A Cryptographic Hash: SHA-1

Depending on the round, the “non-linear” function f is one of the following.

$$f(X,Y,Z) = (X \wedge Y) \vee ((\neg X) \wedge Z)$$

$$f(X,Y,Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$$

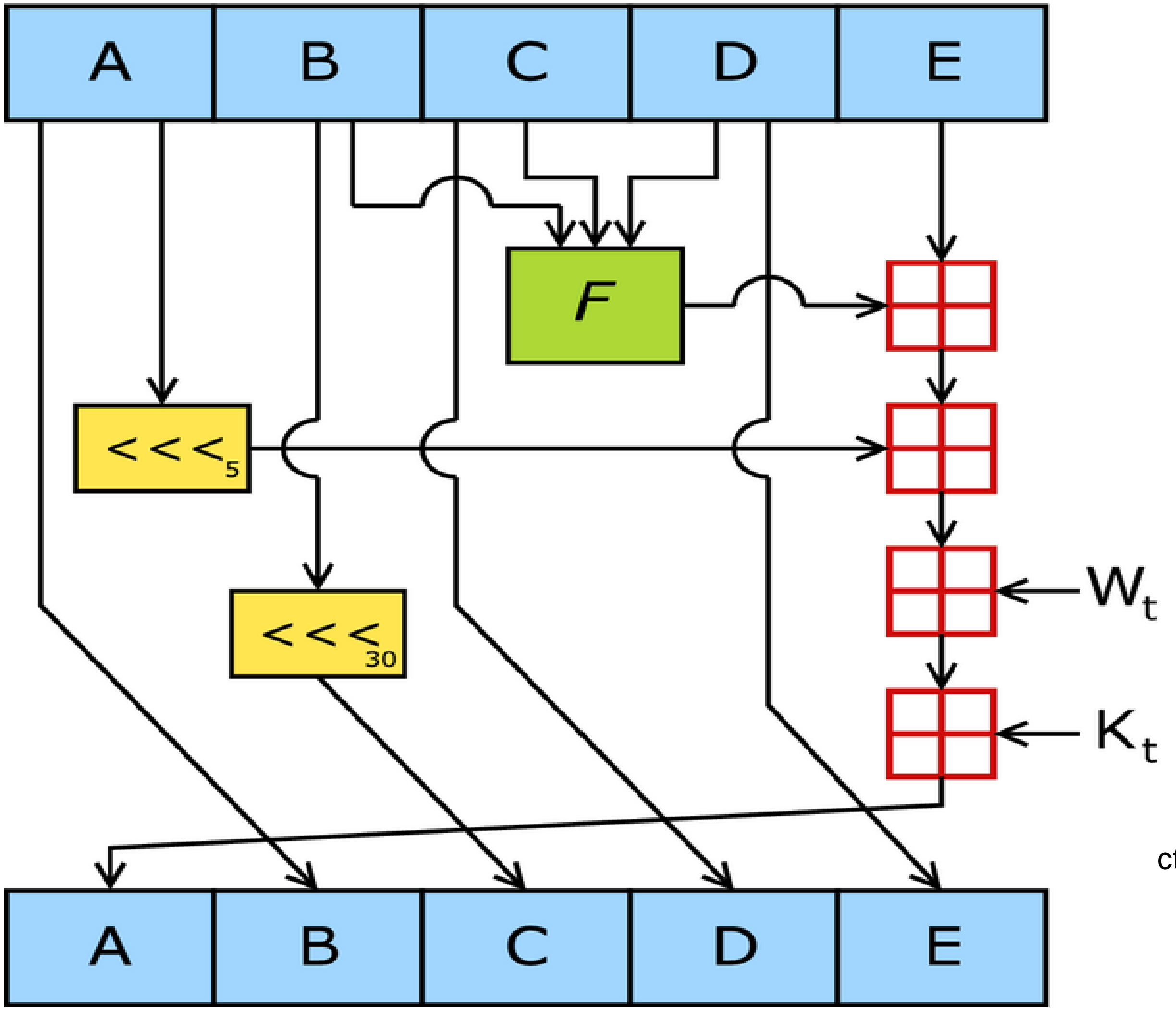
$$f(X,Y,Z) = X \oplus Y \oplus Z$$

A Cryptographic Hash: SHA-1

What's in the final 32-bit transform?

- Take the rightmost word.
- Add in the leftmost word rotated 5 bits.
- Add in a round-dependent function f of the middle three words.
- Add in a round-dependent constant.
- Add in a portion of the 512-bit message.

A Cryptographic Hash: SHA-1



cture from Wikipedia

SHA-1

```
A= 0x67452301, B= 0xefcdab89,
C= 0x98badcfe, D= 0x10325476
E= 0xc3d2e1f0
```

$F_t(X,Y,Z)= (X \wedge Y) \vee ((\neg X) \wedge Z)$, $t= 0,\ldots,19$

$F_t(X,Y,Z)= X \oplus Y \oplus Z$, $t= 20,\ldots,39$

$F_t(X,Y,Z)= (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$, $t= 40,\ldots,59$

$F_t(X,Y,Z)= X \oplus Y \oplus Z$, $t= 60,\ldots,79$

K_t= 0x5a827999, $t= 0,\ldots,19$

K_t= 0x6ed9eba1, $t=20,\ldots,39$

K_t= 0x8f1bbcdc, $t= 40,\ldots,59$

K_t= 0xca62c1d6, $t=60,\ldots,79$

```
Do until no more input blocks {
    If last input block
        Pad to 512 bits by adding 1
        then 0s then 64 bits of
            length.
    $M_i$= input block(32 bits) i=
        0,…,15
    $W_t$= $M_t$ , t= 0,…,15;
    $W_t$= ($W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}$)<<<1,
                t= 16,…,79
    a= A; b= B; c= C; d= D; e= E;
    for(t=0 to 79) {
        x= (a<<<5)+$f_t$(b,c,d)+e+$W_t$+$K_t$
        e= d; d=c; c= b<<<30;
        b=a; a= x;
        }
    A+= a; B+=b; C+= c; D+= d; E+= e;
    }
```

SHA-0

Absence of this term is only difference between SHA-0 and SHA-1

```
A= 0x67452301, B= 0xefcdab89,
C= 0x98badcfe, D= 0x10325476
E= 0xc3d2e1f0
```

$F_t(X,Y,Z)= (X \wedge Y) \vee ((\neg X) \wedge Z)$,
    $t= 0,…,19$
$F_t(X,Y,Z)= X \oplus Y \oplus Z$,
    $t= 20,…,39$
$F_t(X,Y,Z)= (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$,
    $t= 40,…,59$
$F_t(X,Y,Z)= X \oplus Y \oplus Z$, $t= 60,…,79$

K_t= 0x5a827999, $t= 0,…,19$
K_t= 0x6ed9eba1, $t=20,…,39$
K_t= 0x8f1bbcdc, $t= 40,…,59$
K_t= 0xca62c1d6, $t=60,…,79$

```
Do until no more input blocks {
    If last input block
        Pad to 512 bits by adding 1
        then 0s then 64 bits of
            length.
    $M_i$= input block(32 bits)
        $i= 0,…,15$
    $W_t= M_t$, $t= 0,…,15$;
    $W_t= (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})$ $\boxed{<<<1,}$
                $t= 16,…,79$
    a= A; b= B; c= C; d= D; e= E;
    for(t=0 to 79) {
        x= $(a<<<5)+f_t(b,c,d)+e+W_t+K_t$
        e= d; d=c; c= b<<<30;
        b=a; a= x;
        }
    A+= a; B+=b; C+= c; D+= d; E+= e;
    }
```

SHA-0 Strategy (Chabaud and Joux)

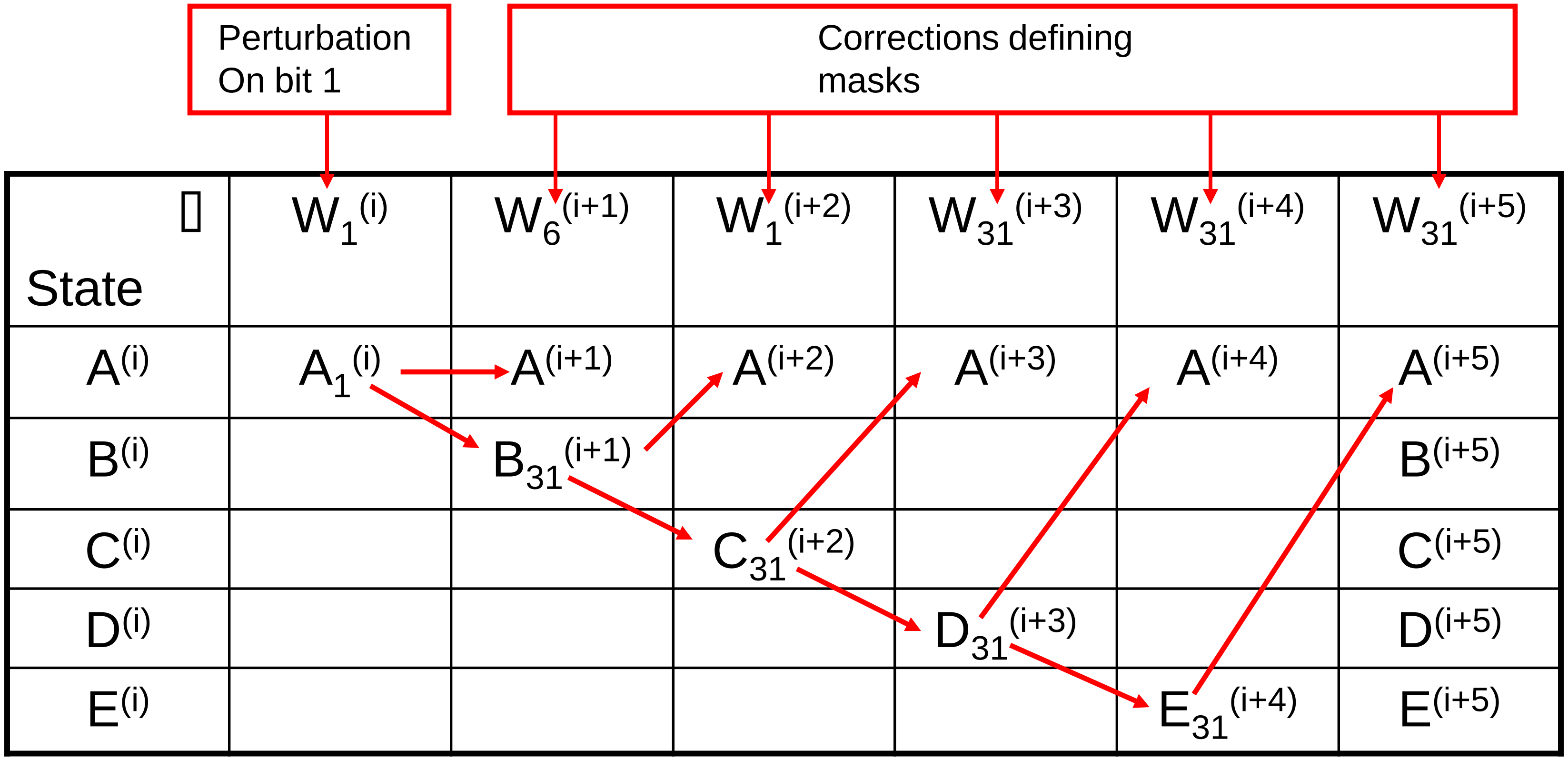
- Basic idea is to look for small differences that can be tracked through rounds like differential cryptanalysis.
- Consider three approximations to the SHA-0 compression function.
  - SHI-1
    - Use Xor instead of Add
    - Make $f^{(i)}$ linear
  - SHI-2
    - Use Xor instead of Add
    - Restore $f^{(i)}$ to original values
  - SHI-3
    - Restore Add
    - Make $f^{(i)}$ linear

SHI-1 Finding Collisions

- Assume the $W^{(i)}$ are unrelated and follow progress of a change to $W^{(1)}$.

	A	B	C	D	E
1	W^1+ROL_5(A)+f(B,C,D)+ E+K	A	ROL_{30}(B)	C	D
2	W^2+ …				
3			ROL_{30}(-)		
4					
5					ROL_{30}(W^1+ROL5(A)+ f(B,C,D)+E+K)
6	W^6+ … - fixes W^1 perturbation				

SHI-1 Error Propagation in Hash



▯ State	$W_1^{(i)}$	$W_6^{(i+1)}$	$W_1^{(i+2)}$	$W_{31}^{(i+3)}$	$W_{31}^{(i+4)}$	$W_{31}^{(i+5)}$
$A^{(i)}$	$A_1^{(i)}$	$A^{(i+1)}$	$A^{(i+2)}$	$A^{(i+3)}$	$A^{(i+4)}$	$A^{(i+5)}$
$B^{(i)}$		$B_{31}^{(i+1)}$				$B^{(i+5)}$
$C^{(i)}$			$C_{31}^{(i+2)}$			$C^{(i+5)}$
$D^{(i)}$				$D_{31}^{(i+3)}$		$D^{(i+5)}$
$E^{(i)}$					$E_{31}^{(i+4)}$	$E^{(i+5)}$

SHI-1 Restoring Expansion

- Flip bit 1 of W^1. This modified A in round “0” resulting, potentially to different (A, B, C, D, E) in round 6. By following linear process we can determine bits in W^1, , W^6 which, when flipped, produce the same (A,B,C,D,E) in round 6.
- Let $M^{(i)}$ be 0 in all positions that are unchanged in round i and 1 where bits are flipped to restore the result in round 6. This is called a local collision.
- This is easy to do, as we’ve seen if there is no expansion.
- Question: If there is expansion, what successful masks are preserved by expansion if bits are flipped in $W^{(1)}$?
- Answer: $M^{(i)} = M^{(i-3)} \oplus M^{(i-8)} \oplus M^{(i-14)} \oplus M^{(i-16)}$, $10<i<80$
- Because SHA-0 expansion doesn’t interleave bits, we can consider each of the 32 bits independently and exhaustively search for a successful pattern

SHI-2

- Restore $f^{(i)}$ and note that rounds 0-19, 40-59 are no longer Xors

Round	Name	$f^{(i)}(X,Y,Z)$	$K^{(i)}$
0-19	IF	$(X \wedge Y) \vee (X \wedge Z)$	0x5a827999
20-39	XOR	$X \oplus Y \oplus Z$	0x6ed9eba1
40-59	MAJ	$(X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$	0x8f1bbcdc
60-79	XOR	$X \oplus Y \oplus Z$	0xca62c1d6

- When does $f^{(i)}$ behave like an Xor for IF and MAJ?
- Again the action on each of the 32 bits is independent

SHI-2 Finding Collisions

- What inputs make IF and MAJ act like and XOR
  - $B^{(i)\prime} = B^{(i)}$, $C^{(i)\prime} = C^{(i)}$, $D^{(i)\prime} = D^{(i)}$
  - Single bit change in $B^{(i)}$, e.g. $B^{(i)} \oplus 2^1$.
  - Single bit change in $C^{(i)}_{31}$ or $D^{(i)}_{31}$, e.g. $C^{(i)} \oplus 2^{31}$ or $D^{(i)} \oplus 2^{31}$ or both.
- The mask becomes probabilistic: We can find a pattern that has the probability with $p=2^{-24}$ from these. Must check every perturbation has foregoing effect: 2,6,14,16,17,18,19,21,22,26,27,28,35,37,41,45,48, 51,54,55,56,58,59,62,63,68,69,70,71,72.
- Perturbations in positions 2, 6 occurs with $p=2^{-6}$. Try many $W^{(15)}$.
- Collision!

```
1a6191b0 3c4a331c 1f228ea2 403b7609  04
062ec496 48611ca8 583401bc 399879d0  4d9
2270fdbd 2a8090f0 4b12fd98 473cc7a1  acc
002831a9 50fe1535 61ac0d3d f26700ec  fa
```

SHI-3

- Change the Xor back to add and f back to linear
- Perturbation in bit 1 of $W^{(i)}$ leading to corrections in $W_{31}^{(i+3)}$, $W_{31}^{(i+4)}$, $W_{31}^{(i+5)}$.
- To prevent carries, non linear constraints must hold on $W_1^{(i)}$, $W_6^{(i+1)}$, $W_1^{(i+2)}$.
- So we fix these.
- Collision

```
53c29e14 44fe051b 4a8ce882 576e1943    91
0c0abc30 3806260d 76cbeb2f 1b8379a8    8bfe
0da433ac 6337b011 1041e2a9 20b44364    e596
1a3f8b70 0e7a4620 25e81245 289acb2b    9382aa9
```

SHA-0

- Perturbations must be inserted without carry.
- Case in SHI-2 where bit 31 in both C and D flip doesn’t work
- Yields two good patterns with probability 2^{-69}.
- Trick to suppress perturbations in rounds 16 and 17 reduces probability to 2^{-61}.
- Probability of finding on is 2^{-22} using basic 2^{-14} collision
- Partial Collision (35 rounds)

```
78fb1285 77a2dc84 4035a90b b61f0b39  97
4a4d1c83 186e8429 74326988 7f220f79  19fa7
a08e7920 16a3e469 2ed4213d 4a75b904  29ac
38bef788 2274a40c 4c14e934 cee12cec  6a
```

- None of this works in SHA-1 because of interleaved bits.

SHA-0 Finding Collisions

- Change the Xor back to add
- Prob of finding on is 2^{-22} using basic 2^{-14} collision
- Partial Collision (35 rounds)

```
78fb1285 77a2dc84 4035a90b b61f0b39  97
4a4d1c83 186e8429 74326988 7f220f79  19fa7
a08e7920 16a3e469 2ed4213d 4a75b904  29ac
38bef788 2274a40c 4c14e934 cee12cec  6a
```

SHA-0 Collisions --- Comments

- Message Expansion: $W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}$ means any round can be determined from any consecutive 16 rounds of message expansion.
- The expanded rounds (all 80) can be represented using a linear transformation, A on 512 bits: $(w, Aw, A^2w, A^3w, A^4w)^T$.
- When the round functions are linearized, a change in bit j of word W_i can be "corrected" by changes in bits j+6, j, j+30, j+30, j+30 in rounds i+1, i+2, i+3, i+4 and i+5.
- When the round functions are replaced by their non-linear versions a change in bit 1 can be corrected by the same pattern with probability between 2^{-2} and 2^{-5}. If change is made to position $j \neq 1$, the probability of correction is reduced by 2^{-3}.
- For SHA-1, because of rotation, one bit change propagates to 107 bits in expansion.

SHA-0 Biham and Chen

- Introduces “disturbance vectors”
  - Collision when last 5 vectors is 0
- Full collision on 65 rounds
- 82 round SHA-0 is weaker than 80 round
- Neutral Bits: Bit i is neutral if disturbance pattern unchanged with complemented i.
- 2-neutral set. Size k(r) of maximal 2-neutral set.

Other Cryptographic Hashes and Performance

Hash Name	Block Size	Relative Speed
MD4	128	1
MD5	128	.68
RIPEMD-128	128	.39
SHA-1	160	.28
RIPEMD-160	160	.24

Breaking news on “Chinese” Attacks on Hashes

- Don’t use MD4 or you’ll look really really silly.
- Don’t use MD5.
- Don’t use RIPEMD-128
- SHA-1 appears to have collision attacks of the order 2^{64}
- Use SHA-2 functions
  - Truncate to provide legacy compatibility if you have to (i.e. – gun to head)
  - Required by “Suite B” Standards

Message Expansion

- Process of expanding from 16 32 bit words to 80 32 bit words in the compression function is called message expansion
  - MD5
    - Permutations
  - SHA-0
    - Linear code (LFSR)
  - SHA-1
    - Linear code with rotation
- Has profound effect on possible disturbance vectors in Differential attacks
- Being studied to provide greater protection
- Replace xor with modular addition to prevent codeword difference propagation
- Conditions on chaining variables for local collision (Prob between 2^{-39} and 2^{-42})

SHA-2

- FIPS 180-2, 8/02.
  - Defines SHA-256, SHA-384, SHA-512.
  - SHA-224 (truncated) added 2/04
- Great increase in mixing between bits of the words compared to SHA-1.
- US Patent *6,829,355*
- Inventor: Glenn Lilly
- Assignee: NSA
- Can obtain source from
  - http://en.wikipedia.org/wiki/SHA-2

SHA-256

```
//Initialize variables:
h0 := 0x6a09e667   //232 times the square root of the first 8 primes 2..19
h1 := 0xbb67ae85, h2 := 0x3c6ef372, h3 := 0xa54ff53a, h4 := 0x510e527f
h5 := 0x9b05688c, h6 := 0x1f83d9ab,h7 := 0x5be0cd19
//Initialize table of round constants:
k(0..63) :=        //232 times the cube root of the first 64 primes 2..311
 0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5, 0x3956c25b, 0x59f111f1,
      0x923f82a4, 0xab1c5ed5,
   0xd807aa98, 0x12835b01, 0x243185be, 0x550c7dc3, 0x72be5d74, 0x80deb1fe,
      0x9bdc06a7, 0xc19bf174,
   0xe49b69c1, 0xefbe4786, 0x0fc19dc6, 0x240ca1cc, 0x2de92c6f, 0x4a7484aa,
      0x5cb0a9dc, 0x76f988da,
   0x983e5152, 0xa831c66d, 0xb00327c8, 0xbf597fc7, 0xc6e00bf3, 0xd5a79147,
      0x06ca6351, 0x14292967,
   0x27b70a85, 0x2e1b2138, 0x4d2c6dfc, 0x53380d13, 0x650a7354, 0x766a0abb,
      0x81c2c92e, 0x92722c85,
   0xa2bfe8a1, 0xa81a664b, 0xc24b8b70, 0xc76c51a3, 0xd192e819, 0xd6990624,
      0xf40e3585, 0x106aa070,
   0x19a4c116, 0x1e376c08, 0x2748774c, 0x34b0bcb5, 0x391c0cb3, 0x4ed8aa4a,
      0x5b9cca4f, 0x682e6ff3,
   0x748f82ee, 0x78a5636f, 0x84c87814, 0x8cc70208, 0x90befffa, 0xa4506ceb,
      0xbef9a3f7, 0xc67178f2
```

SHA-256

```
//Pre-processing:
append a single "1" bit to  message
append "0" bits until message length ≡ 448 ≡ -64 (mod 512)
append length of message (before pre-processing), in bits as 64-bit big-endian
      integer to message

//Process the message in successive 512-bit chunks:
break message into 512-bit chunks
for each chunk
    break chunk into sixteen 32-bit big-endian words w(i), 0 ≤ i ≤ 15

    //Extend the sixteen 32-bit words into sixty-four 32-bit words:
    for i from 16 to 63
        s0 := (w(i-15) rightrotate 7) xor (w(i-15) rightrotate 18) xor (w(i-
      15) rightshift 3)
        s1 := (w(i-2) rightrotate 17) xor (w(i-2) rightrotate 19) xor (w(i-2)
      rightshift 10)
        w(i) := w(i-16) + s0 + w(i-7) + s1

    //Initialize hash value for this chunk:
    a := h0, b := h1, c := h2, d := h3, e := h4, f := h5,  g := h6,  h := h7
```

SHA-256

```
//Main loop:
    for i from 0 to 63
        s0 := (a rightrotate 2) xor (a rightrotate 13) xor (a rightrotate 22)
        maj := (a and b) or (b and c) or (c and a)
        t0 := s0 + maj
        s1 := (e rightrotate 6) xor (e rightrotate 11) xor (e rightrotate 25)
        ch := (e and f) or ((not e) and g)
        t1 := h + s1 + ch + k(i) + w(i)

        h := g, g := f, f := e, e := d + t1,
        d := c, c := b, b := a, a := t0 + t1

    //Add this chunk's hash to result so far:
    h0 := h0 + a, h1 := h1 + b , h2 := h2 + c, h3 := h3 + d
    h4 := h4 + e, h5 := h5 + f,  h6 := h6 + g, h7 := h7 + h

//Output the final hash value (big-endian):
digest = hash = h0 append h1 append h2 append h3 append h4
                   append h5 append h6 append h7
```

Chinese Attack-1

Chinese attack on MD5:
Find M, M': H(M)=H(M'). Select message difference M'=M \oplus \Delta
and differential path b_i'=b_i \oplus \Delta b_i together with sufficient conditions.
For MD5: H_i= f(H_{i-1}, M_i), 0 \le i < 16,
f: (a,b,c,d, w_i, w_{i+1}, w_{i+2}, w_{i+3}) is computed as follows,
a= b+((a+ \phi_i(b,c,d)+ w_i + t_i) <<< s_i),
d= a+((d+ \phi_{i+1}(a,b,c)+ w_{i+1} + t_{i+1}) <<< s_{i+1}),
c= d+((c+ \phi_{i+2}(d,a,b)+ w_{i+2} + t_{i+2}) <<< s_{i+2}),
b= c+((b+ \phi_{i+3}(c,d,a)+ w_{i+3} + t_{i+3}) <<< s_{i+3}) with
\phi_i(X,Y,Z)= (X \vee Y) \wedge (\neg X \vee Z), 0 \le i \le 15,
\phi_i(X,Y,Z)= (X \vee Z) \wedge (Y \vee \neg Z), 16 \le i \le 31,
\phi_i(X,Y,Z)= (X \oplus Y \oplus Z), 32 \le i \le 47,
\phi_i(X,Y,Z)= Y \oplus (X \vee \neg Z), 48 \le i \le 63 and
w_i is the expanded message, w_i, t_i are round dependant constants.
Define \Delta X = X' - X.
\Delta H_0 \rightarrow_{(M_0, M_0')} \Delta H_1 \rightarrow_{(M_1, M_1')} \Delta H_2 \ldots
\rightarrow_{(M_{i-1}, M_{i-1}')} \Delta H_i = H with each composed of
\Delta H_i \rightarrow_{P_2} \Delta R_{i+1,1}
\rightarrow_{P_2} \Delta R_{i+1,2} \rightarrow_{P_3} \Delta R_{i+1,3}
\rightarrow_{P_4} \Delta R_{i+1,4} = \Delta H_{i+1}.

Chinese Attack-2

Let \Delta{i,j}= x_{i,j}' - x_{i,j} = \pm 1 and \Delta x_{i}[j_1 , j_2 , …, j_l]= x_{i}[j_1 , j_2 , … , j_l] - x_i.

Collision is caused by 1024 bit input: (M_0, M_1) with \Delta M_0= (0,0,0,0,2^{31}, 0,0,0,0,0,0,2^{15},0,0,2^{31},0) and \Delta M_1= (0,0,0,0,2^{31}, 0,0,0,0,0,0,-2^{15},0,0,2^{31},0).

Sufficient conditionsinsure that differential holds with high probability. At 8th iteration, b_2= c_2+(b_1+F(c_2,d_2,a_2)+m_7+t_7)<<<22,

we try to control (\Delta c_2 , \Delta d_2, \Delta a_2, \Delta b_1) \rightarrow \Delta b_2 with the following

(A) non-zero bits of \Delta b_2:

- d_{2,11}=1, b_{2,1}=0,
- d_{2,26}= {\overline {a_{2,26}}}=1, b_{2,16}=0,
- d_{2,28}= {\overline {a_{2,28}}}=0, b_{2,i}=0,
- d_{2,11}=1, b_{2,24}=0;

(B) zero bits of \Delta b_2:

- c_{2,i}=0,
- d_{2,i}= a_{2,i},
- c_{2,1}=1,
- d_{2,6}={\overline {a_{2,6}}}=0,
- d_{2,i}= 0,
- d_{2,12}= 1,
- a_{2,24}= 0,

7th bit of c_2, d_2, a_2 result in no change in b_2.

Chinese Attack-3

Algorithm

1: Repeat until first block is found

- (a) Select random M_0,
- (b) Modify M_0,
- (c) M_0, M_0'= M_0 + \Delta M_0 produce \Delta M_0 \rightarrow (\Delta H_1, \Delta M_1) with probability 2^{-37},
- (d) Test characteristics.

2: Repeat until first block is found

- (a) Select random M_1,
- (b) Modify M_1,
- (c) M_1, M_1'= M_1 + \Delta M_1 produce \Delta M_1 \rightarrow 0 with probability 2^{-30}
- (d) Test characteristics.

Breaking news on “Chinese” Attacks on Hashes

- Don’t use MD4 or you’ll look really really silly.
- Don’t use MD5.
- Don’t use RIPEMD-128
- SHA-1 appears to have collision attacks of the order 2^{64}
- Use SHA-2 functions
  - Truncate to provide legacy compatibility if you have to (i.e. – gun to head)
  - Required by “Suite B” Standards

SHA-2

- FIPS 180-2, 8/02.
  - Defines SHA-256, SHA-384, SHA-512.
  - SHA-224 (truncated) added 2/04
- Great increase in mixing between bits of the words compared to SHA-1.
- US Patent *6,829,355*
- Inventor: Glenn Lilly
- Assignee: NSA
- Can obtain source from
  - http://en.wikipedia.org/wiki/SHA-2

SHA-2 Definitions

SHA-256 definitions:

Ch(x,y,z)= (x \wedge y) \oplus (\neg x \wedge z),

Maj(x,y,z)= (x \wedge y) \vee (x \wedge z) \vee (y \wedge z).

\psi$_{256}^{\{i, j, k\}}$(x)= ROTR^{i} (x) \oplus ROTR^{j}(x) \oplus ROTR^{k}(x),

\phi$_{256}^{\{i, j, k\}}$ (x)= ROTR ^{i}(x) \oplus ROTR ^{j} (x) \oplus SHR ^{k} (x).

\Sigma$_{0}^{256}$(x)= \psi$_{256}^{\{2, 13, 22\}}$(x),

\Sigma$_{1}^{256}$ (x)= \psi$_{256}^{\{6, 11, 25\}}$(x).

\sigma$_{0}^{256}$ (x)= \phi$_{256}^{\{7, 18, 3\}}$(x),

\sigma$_{1}^{256}$ (x)= \phi$_{256}^{\{17, 19, 10\}}$ (x).

SHA-512 definitions:

Ch(x,y,z)= (x \wedge y) \oplus (\neg x \wedge z),

Maj(x,y,z)= (x \wedge y) \vee (x \wedge z) \vee (y \wedge z).

\psi$_{512}^{\{i, j, k\}}$(x)= ROTR^{i} (x) \oplus ROTR$^{\wedge ij}$(x) \oplus ROTR^{k} (x),

\phi$_{512}^{\{i, j, k\}}$ (x)= ROTR^{i} (x) \oplus ROTR^{j} (x) \oplus SHR^{k} (x).

\Sigma$_{0}^{512}$ (x)= \psi$_{512}^{\{28,34,39\}}$(x),

\Sigma$_{1}^{512}$(x)= \psi$_{512}^{\{14, 18, 41\}}$(x).

\sigma$_{0}^{512}$(x)= \phi$_{512}^{\{1,8,7\}}$(x),

\sigma$_{1}^{512}$ (x)= \phi$_{512}^{\{19, 61, 6\}}$(x).

SHA-256

```
SHA-256($M_1$ || $M_2$ ||…|| $M_N$):
     for(i=1; i <= N; i++)  {
          $W_t$= $M_t^{(i)}$ , 0 <= t <=15,
          $W_t$ = \sigma$_1^{256}$($W^{t-2}$) \oplus $W^{t-7}$ \oplus \sigma$_0^{256}$($W^{t-15}$ ) \oplus $W^{t-1}$ , 16 <= t <= 63;
          a= $H_0^{(i-1)}$; b= $H_1^{(i-1)}$; c= $H_2^{(i-1)}$; d= $H_2^{(i-1)}$;
          e= $H_4^{(i-1)}$; f= $H_5^{(i-1)}$; g= $H_6^{(i-1)}$; e= $H_7^{(i-1)}$;
          for(t=0; t<64;t++) {
               $T_1$=h + \sigma$_1^{256}$(e)+Ch(e,f,g)+$K_t^{256}$+ $W_t$; $T_2$= \sigma$_0^{256}$ (a)+Maj(e,f,g);
               h= g; g= f; f=e; e= d+$T_1$; d=c;
               c=b; b=a; a= $T_1$+$T_2$;
               }
          $H_0^{(i)}$= a+ $H_0^{(i-1)}$; $H_1^{(i)}$= b+ $H_1^{(i-1)}$; $H_2^{(i)}$= c+ $H_2^{(i-1)}$; $H_3^{(i)}$= d+ $H_3^{(i-1)}$;
          $H_4^{(i)}$= e+ $H_4^{(i-1)}$; $H_5^{(i)}$= f+ $H_5^{(i-1)}$; $H_6^{(i)}$= g+ $H_6^{(i-1)}$; $H_7^{(i)}$= h+ $H_7^{(i-1)}$;
     }
```

SHA-512 is the same except there are 79 rounds and the words are 64 bits long.

Other Cryptographic Hashes and Performance

Hash Name	**Block Size**	**Relative Speed**
MD4	128	1
MD5	128	.68
RIPEMD-128	128	.39
SHA-1	160	.28
RIPEMD-160	160	.24

What to take home

- Symmetric ciphers and hashes provide key ingredients for “distributed security”
  - Fast data transformation to provide confidentiality
  - Integrity
  - Public key crypto provides critical third component (trust negotiation, key distribution)
- It’s important to know properties of cryptographic primitives and how likely possible attacks are, etc.
  - Most modern ciphers are designed so that knowing output of n-1 messages provides no useful information about n^{th} message.
  - This has an effect on some modes of operation.

End