Cryptanalysis

Cryptographic Hashes

John Manferdelli
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 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 4→3. Functions satisfying this condition are called Collision Resistant Functions (CRFs).

Observations

- Collision Resistance $\rightarrow$ 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

One-Way Functions

- Hashes come from two basic classes of one-way functions
  - Mathematical
    - Multiplication: $Z = X \cdot Y$
    - Modular Exponentiation: $Z = Y^X \pmod{n}$ (Chaum vP Hash)
  - Ad-hoc (Symmetric cipher-like constructions)
    - Custom Hash functions (MD4, SHA, MD5, RIPEMD)

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 $a^x=b \pmod p$ for some unknown x).
- g: $\{1,2,\dots,q-1\}^2 \to \{1,2,\dots,p-1\}$, $q=(p-1)/2$ by:
  - $g(s, t) = a^s b^t \pmod p$
- Reduction to discrete log:

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

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

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

A Cryptographic Hash: SHA-1

(IV) – 160 bits

512-bit Input

Compression
Function

160-bit Output

Slide by Josh Benaloh
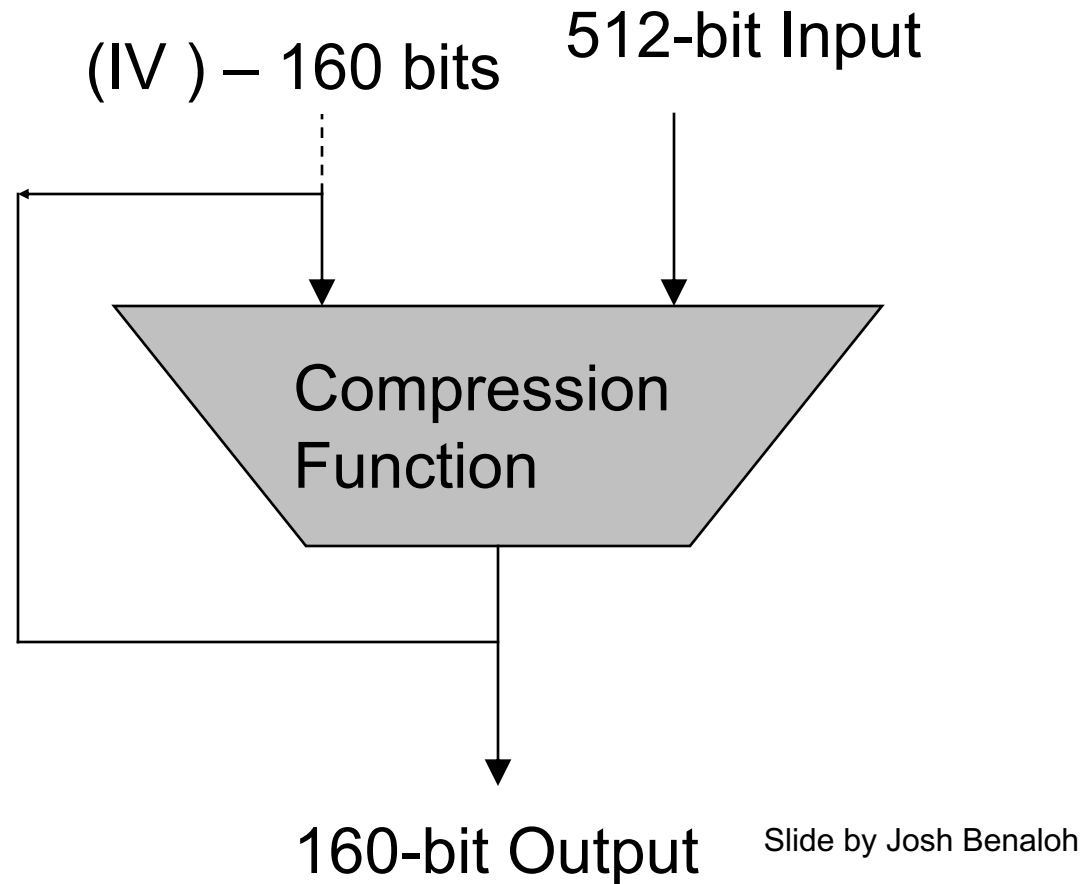
SHA-1: State and message schedule

- Compression function takes 160-bit state and 512 bit input and produces new 160 bit state (one Merkle Damgard round)
- 512-bit message input block: 16 32-bit words (M_0, …, M_{15})
- Compression consists of 80 rounds
  - Each round uses one 32 bit word derived from input block
  - Message expansion algorithm produces subsequent rounds
    - $W_t = M_t$, $0 \leq t < 16$
    - $W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})$ <<<1, $16 \leq t < 80$
    - Structure of round is same for all 80 rounds:

      X= (a<<<5)+f_t(b,c,d)+e+W_t+K_t

      E= d; d=c; c= b<<<30; b=a; a= x;

      Three f_t functions. First used in rounds 0 through 19, Second used in rounds 20 through 39.   Third used in rounds 40-59. First reused in rounds 60-79

SHA-1round
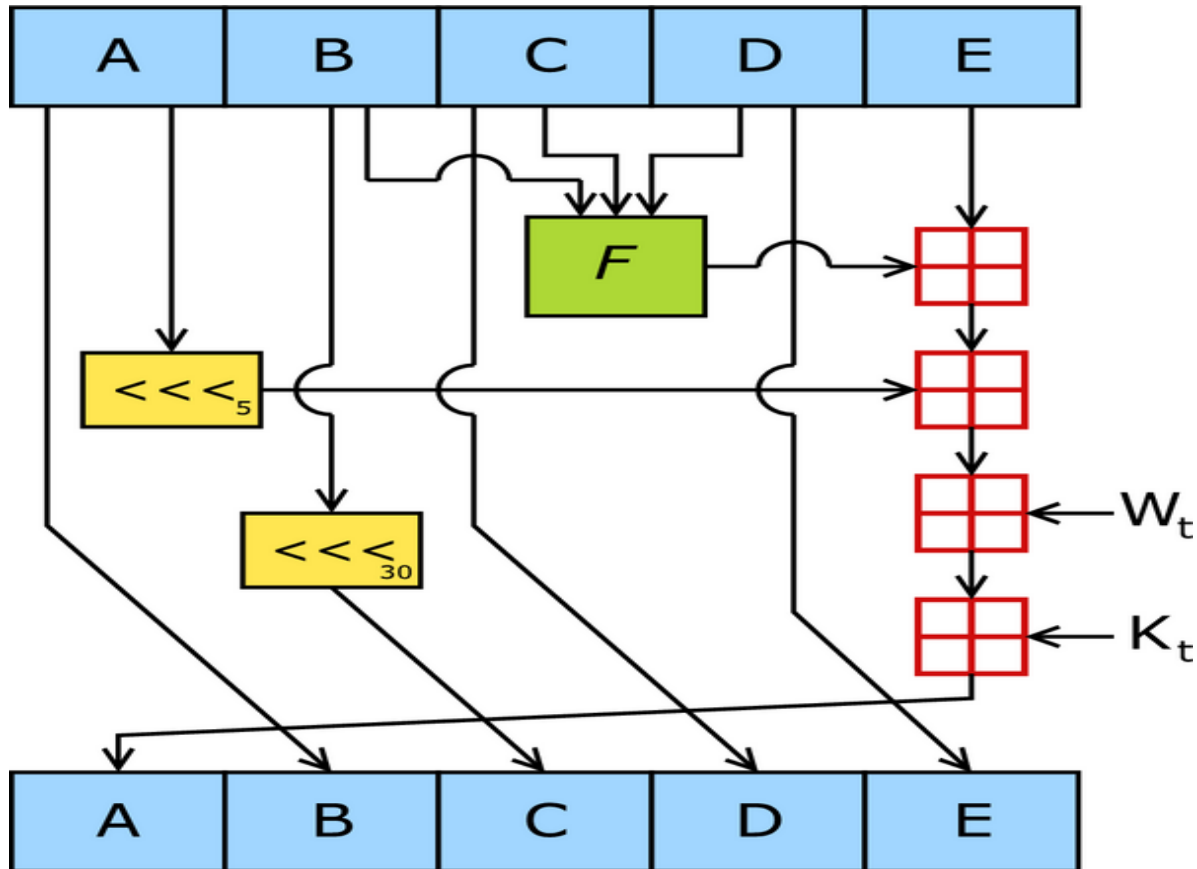
Picture from Wikipedia

A Cryptographic Hash: SHA -1

- Depending on the round, the "function f is one of the following.

$$f(X,Y,Z) = (X \land Y) \lor ((\neg X \land Z)$$

$$f(X,Y,Z) = (X \land Y) \lor (X \land Z) \lor (Y \land Z)$$

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

- Note first two are non-linear. Third is linear and provides diffusion.

SHA-0/1

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

$F_t(X,Y,Z) = (X \land Y) \lor ((\lnot X) \land Z)$,
    t= 0,…,19
$F_t(X,Y,Z) = X \oplus Y \oplus Z$,
    t= 20,…,39
$F_t(X,Y,Z) = (X \land Y) \lor (X \land Z) \lor (Y \land Z)$,
    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.
    Mi= input block(32 bits)
        i= 0,…,15
    Wt= Mt, t= 0,…,15;
    Wt= (Wt-3⊕Wt-8⊕Wt-14⊕Wt-16) <<<1,
                t= 16,…,79
    a= A; b= B; c= C; d= D; e= E;
    for(t=0 to 79) {
        x= (a<<<5)+ft(b,c,d)+e+Wt+Kt
        e= d; d=c; c= b<<<30;
        b=a; a= x;
        }
    A+= a; B+=b; C+= c; D+= d; E+= e;
}
```
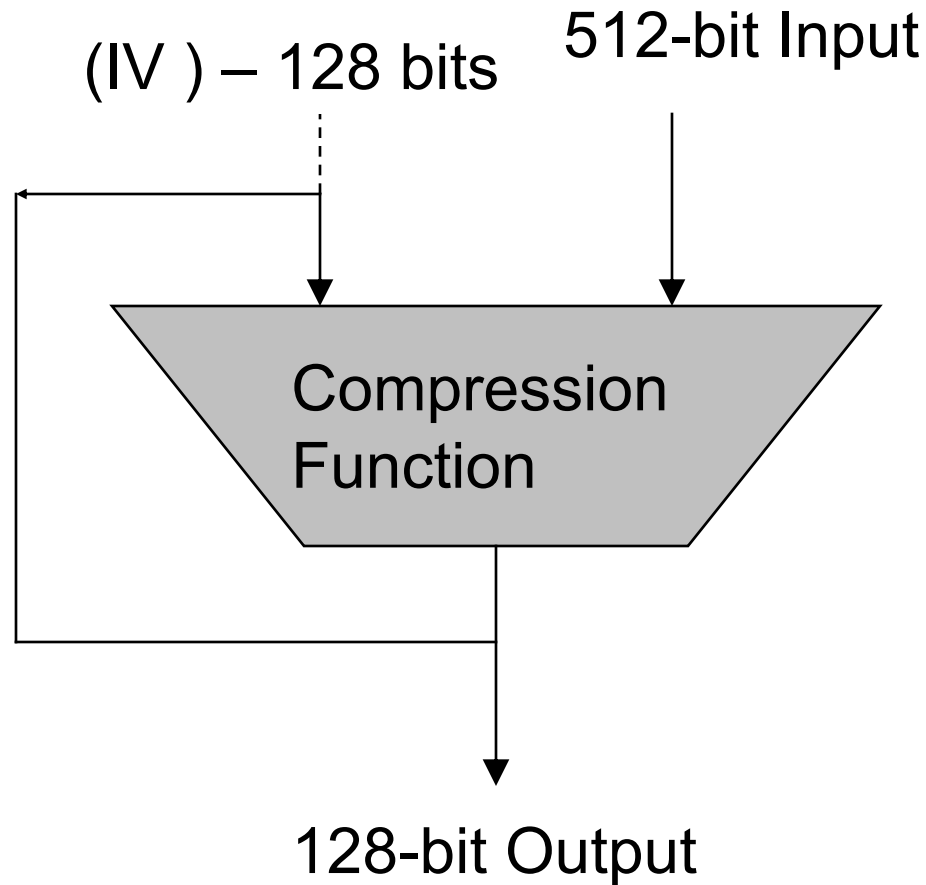
MD4

- Invented by Rivest, ca 1990
- Weaknesses found by 1992
  - Rivest proposed improved version (MD5), 1992
  - SHA-0/1, 1993/1995
  - SHA-2, 2001
  - SHA-3, 2012
- Dobbertin found MD4 collision in 1998

A Cryptographic Hash: MD-4

(IV) – 128 bits   512-bit Input

Compression Function

128-bit Output   Slide by Josh Benaloh

MD4: State and message schedule

- Compression function takes 128-bit state and 512 bit input and produces new 128 bit state (one Merkle Damgard round)
- 512-bit message input block: 16 32-bit words (M_0, …, M_{15})
- Compression consists of 48 rounds
  - Each round uses one 32 bit word derived from input block
  - Message expansion algorithm produces subsequent rounds
    - $W_t = M_{s(t)}$, $0 \le t < 47$
    - Structure of round is same for all 48 rounds, 3 round functions

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s(t)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
t	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
s(t)	0	4	8	12	1	5	9	13	2	6	10	14	3	7	11	15
t	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
s(t)	0	8	4	12	2	10	6	14	1	9	5	13	3	11	7	15

MD4 round
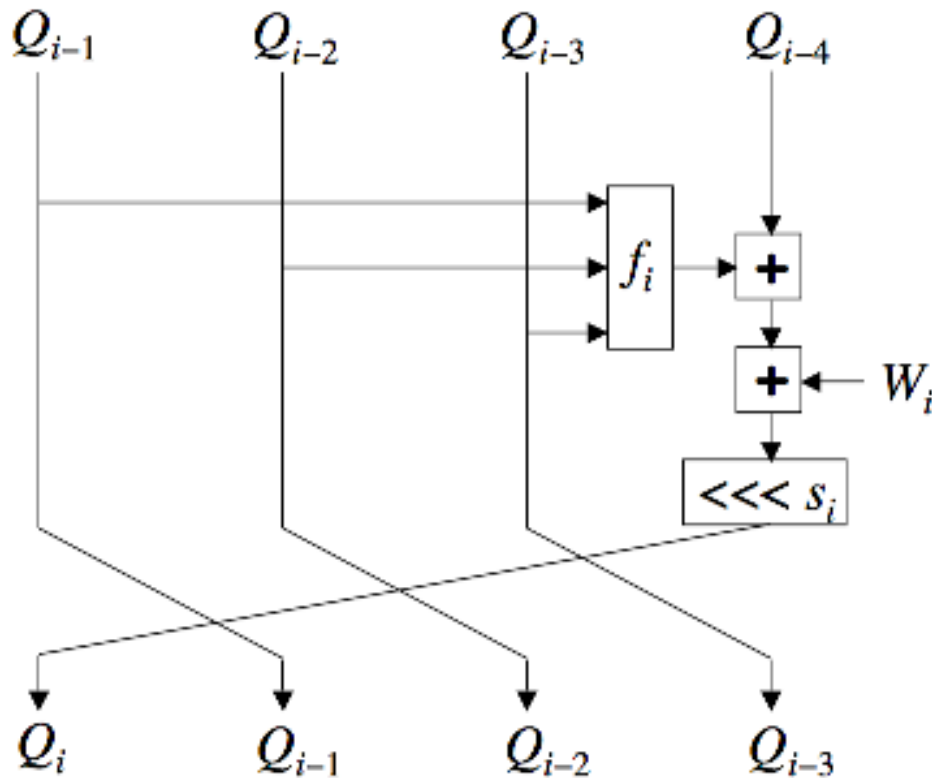

```
fi(A,B,C)
   (A∧B)∨(¬A∧C), 0≤i<16
   (A∧B)∨(A∧C)∨(B∧C), 16≤i<32
    A⊕B⊕C, 32≤i<48

K0 = 0x00000000,
K1 = 0x5a827999,
K2 = 0x6ed9eba1.
```

- Where $f_i(A,B,C) = \begin{cases} F(A,B,C) + K_0 & \text{if } 0 \le i \le 15 \\ G(A,B,C) + K_1 & \text{if } 16 \le i \le 31 \\ H(A,B,C) + K_2 & \text{if } 32 \le i \le 47 \end{cases}$

Slide by Mark Stamp

MD4 Algorithm

// $M = (Y_0, Y_1, \ldots, Y_{N-1})$, message to hash, after padding
// Each Y_i is a 32-bit word and N is a multiple of 16
MD4(M)
    // initialize $(A, B, C, D) = \mathrm{IV}$
    $(A, B, C, D) = (\mathtt{0x67452301}, \mathtt{0xefcdab89}, \mathtt{0x98badcfe}, \mathtt{0x10325476})$
    for $i = 0$ to $N/16 - 1$
        // Copy block i into X
        $X_j = Y_{16i+j}$, for $j = 0$ to 15
        // Copy X to W
        $W_j = X_{\sigma(j)}$, for $j = 0$ to 47
        // initialize Q
        $(Q_{-4}, Q_{-3}, Q_{-2}, Q_{-1}) = (A, D, C, B)$
        // Rounds 0, 1 and 2
        Round0(Q, X)
        Round1(Q, X)
        Round2(Q, X)
        // Each addition is modulo 2^{32}
        $(A, B, C, D) = (Q_{44} + Q_{-4}, Q_{47} + Q_{-1}, Q_{46} + Q_{-2}, Q_{45} + Q_{-3})$
    next i
    return A, B, C, D
end MD4

Overview of attack

- Try to find one block collision
- Denote $M = (X_0, X_1, \ldots, X_{15})$
- Define M' by $X'_i = X_i$ for $i \neq 12$ and $X'_{12} = X_{12} + 1$
- Word X_{12} only appears in steps 12, 19, 35
  - This provides a "natural" round division of the attack
- We have the freedom to choose $X_0, X_1, \ldots, X_{11}$ at our convenience
- Goal is to find pair M and M' with $\Delta_{35} = (0,0,0,0)$

JLM 20101208

Dobbertin's attack strategy

- Specify a differential condition
- If condition holds, there's a probability of collision---try enough times for overall probability to be high.
- Derive system of nonlinear equations: solution satisfies differential condition
- Find efficient method to solve equations
- Find enough solutions to yield a collision
- Find one-block collision, where M= $(X_0, X_1, \ldots, X_{15})$, M′= $(X'_0, X'_1, \ldots, X'_{15})$
- Difference is subtraction mod 2^{32}
- Blocks differ in only 1 word
  - Difference in that word is exactly 1
- Limits avalanche effect to steps 12 thru19
  - Only 8 of the 48 steps are critical to attack!
  - System of equations applies to these 8 steps

Slide by Mark Stamp

Notation

- Suppose $(Q_j, Q_{j-1}, Q_{j-2}, Q_{j-3}) = MD4_{0\ldots j}(IV, M)$
  and $(Q'_j, Q'_{j-1}, Q'_{j-2}, Q'_{j-3}) = MD4_{0\ldots j}(IV, M')$
- Define $\triangle_j = (Q_j - Q'_j,\ Q_{j-1} - Q'_{j-1},\ Q_{j-2} - Q'_{j-2},\ Q_{j-3} - Q'_{j-3})$
  where subtraction is modulo 2^{32}
- Let $\pm 2^n$ denote $\pm 2^n \bmod 2^{32}$.
  - $2^{25} = $ 0x02000000 and $-2^5 = $ 0xfffffffe0
- All arithmetic is modulo 2^{32}

Three phases of MD4 attack

1. Show: $\Delta_{19} = (2^{25}, -2^5, 0, 0)$ implies probability at least $1/2^{30}$ that the Δ_{35} condition holds

   – Uses differential cryptanalysis

2. "Backup" to step 12: We can start at step 12 and have Δ_{19} condition hold

   – By solving system of nonlinear equations

3. "Backup" to step 0: And find collision

   – In each phase of attack, some words of M are determined

   – When completed, have M and M'

     • Where $M \neq M'$ but $h(M) = h(M')$

   – Equation solving step is tricky part

     • Nonlinear system of equations

     • Must be able to solve efficiently

Steps 19 to 35

- Differential phase of the attack
- M and M′ as given above
  - Only differ in word 12
- Assume that $\Delta_{19} = (2^{25}, -2^5, 0, 0)$
  - $G(Q_{19}, Q_{18}, Q_{17}) =$ $G(Q'_{19}, Q'_{18}, Q'_{17})$
- Then we compute probabilities of "Δ" conditions at steps 19 thru 35
- Total probability: 2^{-30}, actually 2^{-22}

j	ΔQ_j	ΔQ_{j-1}	ΔQ_{j-2}	ΔQ_{j-3}	i	s_j	p	Input
			Δ_j					
19	2^{25}	-2^5	0	0	$*$	$*$	$*$	$*$
20	0	2^{25}	-2^5	0	1	3	1	X_1
21	0	0	2^{25}	-2^5	1	5	$1/9$	X_5
22	-2^{14}	0	0	2^{25}	1	9	$1/3$	X_9
23	2^6	-2^{14}	0	0	1	13	$1/3$	X_{13}
24	0	2^6	-2^{14}	0	1	3	$1/9$	X_2
25	0	0	2^6	-2^{14}	1	5	$1/9$	X_6
26	-2^{23}	0	0	2^6	1	9	$1/3$	X_{10}
27	2^{19}	-2^{23}	0	0	1	13	$1/3$	X_{14}
28	0	2^{19}	-2^{23}	0	1	3	$1/9$	X_3
29	0	0	2^{19}	-2^{23}	1	5	$1/9$	X_7
30	-1	0	0	2^{19}	1	9	$1/3$	X_{11}
31	1	-1	0	0	1	13	$1/3$	X_{15}
32	0	1	-1	0	2	3	$1/3$	X_0
33	0	0	1	-1	2	9	$1/3$	X_8
34	0	0	0	1	2	11	$1/3$	X_4
35	0	0	0	0	2	15	1	$X_{12}, X_{12} + 1$

Slide by Mark Stamp

JLM 20101208

Computing p

- Consider Δ_{35}
- Suppose j = 34 holds: Then $\Delta_{34} = (0,0,0,1)$ and

$$
\begin{aligned}
Q_{35} &= (Q_{31} + H(Q_{34}, Q_{33}, Q_{32}) + X_{12} + K_2) \lll 15 \\
&= ((Q'_{31} + 1) + H(Q'_{34}, Q'_{33}, Q'_{32}) + X_{12} + K_2) \lll 15 \\
&= (Q'_{31} + H(Q'_{34}, Q'_{33}, Q'_{32}) + (X_{12} + 1) + K_2) \lll 15 \\
&= Q'_{35}
\end{aligned}
$$

- Implies $\Delta_{35} = (0,0,0,0)$ with probability 1
  – As summarized in j = 35 row of table

Slide by Mark Stamp

Steps 12 to 19

- Analyze steps 12 to 19, find conditions that ensure $\Delta_{19} = (2^{25},-2^5,0,0)$
  - $G(Q_{19},Q_{18},Q_{17}) = G(Q'_{19},Q'_{18},Q'_{17})$, as required in differential phase
- Step 12 to 19—equation solving phase
- This is most complex part of attack
  - Last phase, steps 0 to 11, is easy

j	i	s_j	M Input	M' Input
12	0	3	X_{12}	$X_{12}+1$
13	0	7	X_{13}	X_{13}
14	0	11	X_{14}	X_{14}
15	0	19	X_{15}	X_{15}
16	1	3	X_0	X_0
17	1	5	X_4	X_4
18	1	9	X_8	X_8
19	1	13	X_{12}	$X_{12}+1$

JLM 20101208

Steps 12 to 19

- To apply differential phase, must have $\Delta_{19} = (2^{25}, -2^5, 0, 0)$

  $Q_{19} = Q'_{19} + 2^{25}$

  $Q_{18} + 2^5 = Q'_{18}$

  $Q_{17} = Q'_{17}$

  $Q_{16} = Q'_{16}$

- At step 12 we have

  $Q_{12} = (Q_8 + F(Q_{11}, Q_{10}, Q_9) + X_{12}) <<< 3$

  $Q'_{12} = (Q'_8 + F(Q'_{11}, Q'_{10}, Q'_9) + X'_{12}) <<< 3$

- Since $X'_{12} = X_{12} + 1$ and $(Q_8, Q_9, Q_{10}, Q_{11}) = (Q'_8, Q'_9, Q'_{10}, Q'_{11})$, $(Q'_{12} <<< 29) - (Q_{12} <<< 29) = 1$

Slide by Mark Stamp

Equations for 12 to 19

- Similar analysis for remaining steps yields system of equations:

$$1 = (Q'_{12} \lll 29) - (Q_{12} \lll 29)$$

$$F(Q'_{12}, Q_{11}, Q_{10}) - F(Q_{12}, Q_{11}, Q_{10}) = (Q'_{13} \lll 25) - (Q_{13} \lll 25)$$

$$F(Q'_{13}, Q'_{12}, Q_{11}) - F(Q_{13}, Q_{12}, Q_{11}) = (Q'_{14} \lll 21) - (Q_{14} \lll 21)$$

$$F(Q'_{14}, Q'_{13}, Q'_{12}) - F(Q_{14}, Q_{13}, Q_{12}) = (Q'_{15} \lll 13) - (Q_{15} \lll 13)$$

$$G(Q'_{15}, Q'_{14}, Q'_{13}) - G(Q_{15}, Q_{14}, Q_{13}) = Q_{12} - Q'_{12}$$

$$G(Q_{16}, Q'_{15}, Q'_{14}) - G(Q_{16}, Q_{15}, Q_{14}) = Q_{13} - Q'_{13}$$

$$G(Q_{17}, Q_{16}, Q'_{15}) - G(Q_{17}, Q_{16}, Q_{15}) = Q_{14} - Q'_{14} + (Q'_{18} \lll 23)$$
$$- (Q_{18} \lll 23)$$

$$G(Q'_{18}, Q_{17}, Q_{16}) - G(Q_{18}, Q_{17}, Q_{16}) = Q_{15} - Q'_{15} + (Q'_{19} \lll 19)$$
$$- (Q_{19} \lll 19) - 1$$

Slide by Mark Stamp

Solving the equations

- To solve this system must find
  $$(Q_{10}, Q_{11}, Q_{12}, Q_{13}, Q_{14}, Q_{15}, Q_{16}, Q_{17}, Q_{18}, Q_{19}, Q'_{12}, Q'_{13}, Q'_{14}, Q'_{15})$$
  so that all equations hold.
- Since there are 14 variables and 8 equations, we have wiggle room
- Given such a solution, we determine X_j for $j = 13, 14, 15, 0, 4, 8, 12$
  so that we begin at step 12 and arrive at step 19 with Δ_{19} condition satisfied
- This phase reduces to solving (nonlinear) system of equations
- Can manipulate the equations so that
  - Choose $(Q_{14}, Q_{15}, Q_{16}, Q_{17}, Q_{18}, Q_{19})$ arbitrary
  - Which determines $(Q_{10}, Q_{13}, Q'_{13}, Q'_{14}, Q'_{15})$

Slide by Mark Stamp

Conditions for solution

- Three conditions must be satisfied:

$$G(Q_{15}, Q_{14}, Q_{13}) - G(Q'_{15}, Q'_{14}, Q'_{13}) = 1$$
$$F(Q'_{14}, Q'_{13}, 0) - F(Q_{14}, Q_{13}, -1) - (Q'_{15} \lll 13) + (Q_{15} \lll 13) = 0.$$
$$G(Q_{19}, Q_{18}, Q_{17}) = G(Q'_{19}, Q'_{18}, Q_{17})$$

- First 2 are "check" equations
  - Third is "admissible" condition
- Naïve algorithm: choose six Q_j, yields five Q_j, Q'_j until 3 equations satisfied
- How much work is this?

Message conditions for equations

- Using this we can solve for seven message words:
    - X_{13}= anything
    - X_{14}= $(Q_{14}<<<21)-Q_{10}-F(Q_{13}, Q_{12}, Q_{11})$
    - X_{15}= $(Q_{15}<<<21)-Q_{11}-F(Q_{14}, Q_{13}, Q_{12})$
    - X_0= $(Q_{16}<<<21)-Q_{12}-G(Q_{15}, Q_{14}, Q_{13}) -K_1$
    - X_4= $(Q_{17}<<<21)-Q_{13}-G(Q_{16}, Q_{15}, Q_{14}) -K_1$
    - X_8= $(Q_{18}<<<21)-Q_{14}-G(Q_{17}, Q_{16}, Q_{15}) -K_1$
    - X_{12}= $(Q_{19}<<<21)-Q_{15}-G(Q_{18}, Q_{17}, Q_{16}) -K_1$

Solution

- Choose $Q_{12}= -1$, $Q_{12}'=0$, $Q_{11}=0$. Then
  - $Q_{15}'= Q_{15}-G(Q_{18}',Q_{17},Q_{16})+G(Q_{18},Q_{17},Q_{16})+(Q_{19}'<<<19)-(Q_{19}<<<19)-1$
  - $Q_{14}'= Q_{14}-G(Q_{18}',Q_{17},Q_{16})+G(Q_{18},Q_{17},Q_{16})+(Q_{18}'<<<23)-(Q_{19}<<<23)$
  - $Q_{13}= (Q_{14}<<<21)-(Q_{14}'<<<21)$
  - $Q_{13}'= Q_{13}-G(Q_{16},Q_{15}',Q_{14}')+G(Q_{16},Q_{15},Q_{14})$
  - $Q_{10}= (Q_{13}'<<<25)-(Q_{13}<<<25)$
  - $F(Q_{18}',Q_{17},Q_{16})-F(Q_{18},Q_{17},Q_{16})= (Q_{15}'<<<13)-(Q_{15}<<<13)$
  - $G(Q_{18}',Q_{17},Q_{16})-G(Q_{18},Q_{17},Q_{16})= Q_{12}-Q_{11}'$
- Choose $Q_{14}, \ldots, Q_{19}$ arbitrarily and solve for $Q_{10}, Q_{13}, Q_{13}', Q_{14}', Q_{15}'$
  - $G(Q_{15},Q_{14},Q_{13})-G(Q_{15}', Q_{14}', Q_{13}')= 1$
  - $F(Q_{14}', Q_{13}', 0)-F(Q_{14}, Q_{13,}, -1)= 0$
  - $G(Q_{19}', Q_{18}', Q_{17})= G(Q_{19}, Q_{18,}, Q_{17})$

Continuous Approximation

- Each equation holds with probability $1/2^{32}$
- Appears that 2^{96} iterations required
  - Since three 32-bit check equations
  - Birthday attack on MD4 is only 2^{64} work!
- Solution
  - A "continuous approximation"
  - Small changes, converge to a solution

Slide by Mark Stamp

Approximation technique

- Generate random Q_i values until first check equation is satisfied
  - Random one-bit modifications to Q_i
  - Save if 1st check equation still holds **and** 2nd check equation is "closer" to holding
  - Else try different random modifications
- Modifications converge to solution
  - Then 2 check equations satisfied
  - Repeat until admissible condition holds

Slide by Mark Stamp

Steps 0 to 11

- At this point, we have (Q_8,Q_9,Q_{10},Q_{11}) and $MD4_{12...47}(Q_8,Q_9,Q_{10},Q_{11},X) = MD4_{12...47}(Q_8,Q_9,Q_{10},Q_{11},X')$
- To finish, we must have $MD4_{0...11}(IV,X) = MD4_{0...11}(IV,X') = (Q_8,Q_9,Q_{10},Q_{11})$
- Recall, X_{12} is only difference between M, M′
- Also, X_{12} first appears in step 12
- Have already found X_j for j= 0,4,8,12,13,14,15
- Free to choose X_j for j= 1,2,3,5,6,7,9,10,11 so that $MD4_{0...11}$ equation holds easily!

Slide by Mark Stamp

Recap

- Attack proceeds as follows…
    1. Steps 12 to 19: Find $(Q_8, Q_9, Q_{10}, Q_{11})$ and X_j for j= 0,4,8,12,13,14,15
    2. Steps 0 to 11: Find X_j for remaining j
    3. Steps 19 to 35: Check Δ_{35}= (0,0,0,0)
        - If so, have found a collision!
        - If not, go to 2.

JLM 20101208

Meaningful Collision

- Different contracts, same hash value

```
********************          ********************
CONTRACT                      CONTRACT

At the price of $176,495 Alf Blowfish    At the price of $276,495 Alf Blowfish
sells his house to Ann Bonidea ...       sells his house to Ann Bonidea ...
```

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)].

Diamond structure
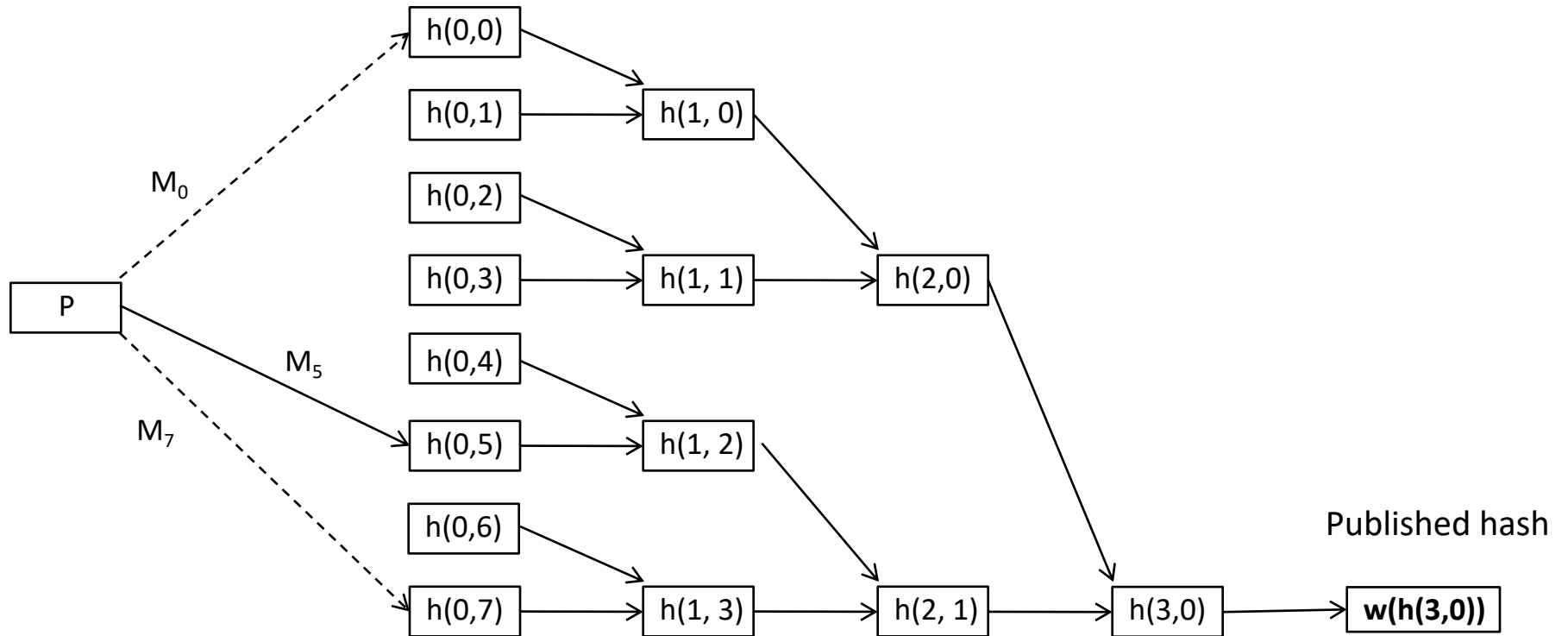
Nostradamus ("herding") attack

- 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.

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.

Jesse Walker, Ph.D.

Intel Corporation

Intel Labs – Circuits and Systems Research – Security Research Lab

jesse.walker@intel.com

The Early Years

- Historical Context
- Rabin's Hash Function
- Davies-Meyer
- MDC2

Historical Context

- Computer scientists introduced **hash function**s to create a compact table index optimizing search

- Requirement: a hash function $H : Objects \rightarrow Indices$ acts like a **random mapping**
  - Minimize probability that $H(m) = H(m')$ when $m \neq m'$

    $m_1 \ldots m_k \leftarrow m; \ h_0 \leftarrow 0$

    $\mathbf{do} \ j = 1 \ \mathbf{to} \ k \Rightarrow h_j \leftarrow f(h_{j-1}, m_j) \ \mathbf{od}$

    $\mathbf{return} \ h_k$
  - f usually chosen to be a number theoretic mixer, e.g., $f(h,m) = (h + am + b) \ \mathbf{mod} \ c$ for primes a, b, c

Digital Signatures

- In 1978 M. Rabin wanted to create a digital signature scheme

- Rabin needed something like a hash function to "compress" the message into a fixed sized "index"

- Requirements:
  - Act like a random mapping
  - **Collision resistance**: it is hard find two documents with same hash or **digest**
  - **2nd pre-image resistance**: given a hash of a document, it is hard to find a second document with same hash
  - **Pre-image resistance**: given a hash value, it is to find a document that produces that hash

Rabin's Hash Function

- Rabin realized DES, being a strong pseudo-random mixer, can replace the non-cryptographic f in conventional hash function designs

  $RabinHash\ (h_0, m)$

  $\quad m_1 \ldots m_k \leftarrow m$

  $\quad \textbf{do}\ j = 1\ \textbf{to}\ k \Rightarrow h_j \leftarrow DES(m_j, h_{j-1})\ \textbf{od}$

  $\quad \textbf{return}\ (h_0, h_m)$

- Must return (h_0, h_m) instead of h_m to obtain collision resistance

  $RabinHash\ (h_0, m_1\ m_2) = DES\ (m_2, DES\ (m_1, h_0)) = DES\ (m_2, h_1) = RabinHash\ (h_1, m_2)$

- Lesson 1: The initial value h_0 must be fixed to obtain collision resistance

Birthday Problems

- The standard **Birthday Problem**:
  - Given q people who live on a planet with an n day year, what is the probability two share a birthday?
  - Answer: Assuming birthdays are uniformly distributed, approximately $q^2/2n$ if $q \le n^{1/2}$

- The Birthday Problem for two sets:
  - Given a population of q_1 boys and q_2 girls who live on a planet with an n day year, what is the probability a boy and girl share a birthday?
  - Answer: When $q = q_1 = q_2$, assuming birthdays are uniformly distributed, approximately q^2/n if $q \le n^{1/2}$

Attacking Rabin Hash

Coppersmith: To find a 2nd pre-image for $RabinHash\ (h_0, m_1\ m_2)$:

- Let $h_2 = RabinHash\ (h_0, m_1\ m_2)$ Then compute

  $\textbf{do}\ j = 1\ \textbf{to}\ 2^{32} \Rightarrow s_j \leftarrow_\$ \{0,1\}^{56};\ u_j \leftarrow DES\ (s_j, h_0)\ \textbf{od}$

  $\textbf{do}\ j = 1\ \textbf{to}\ 2^{32} \Rightarrow t_j \leftarrow_\$ \{0,1\}^{56};\ v_j \leftarrow DES^{-1}\ (t_j, h_2)\ \textbf{od}$

- By the Birthday problem for two lists the probability that j, j' exists with $u_j = v_{j'}$ is approximately $(2^{32})(2^{32})/2^{64} = 1$

- Then $RabinHash\ (h_0, s_j\ t_{j'}) = DES\ (t_{j'}, DES\ (s_j, h_0)) = DES\ (t_{j'}, u_j) = DES\ (t_{j'}, v_{j'}) = h_2 = RabinHash\ (h_0, m_1\ m_2)$

Discussion

- Collision resistance implies 2nd pre-image resistance, because if we produce a 2nd pre-image then we also produce a collision

- Exercise: modify the attack to produce pre-images

- Lesson 2: We must somehow neutralize the decryption function to build successful hash functions from block ciphers

- Lesson 3: Hash functions are attacked by multi-block messages, which enables various forms of the Birthday problem to govern their security

Neutralizing Decryption

- In the early 1980s Davies and Meyer observed that $(h, m) \rightarrow DES\,(m, h) \oplus h$ is one-way

  – Given h' it is hard to find m and h such that
  $$h' = DES\,(m, h) \oplus h$$

- The **Davies-Meyer construction** replaces DES in the Rabin hash function:

  $DaviesMeyerHash\,(m)$

  $m_1 \ldots m_k \leftarrow m;\ h_0 \leftarrow iv$

  do $j = 1$ **to** $k \Rightarrow h_j \leftarrow DES(m_j, h_{j-1}) \oplus h_{j-1}$ **od**

  return h_m

- Does this work?

The Ideal Cipher Model

- Davies and Meyer reasoned as if DES were an **ideal cipher** E.
  - For each "key" m, $DES(m, \cdot)$ acts like a random permutation of 64 bits strings $\{0,1\}^{64}$

- It is easy to reason about an ideal cipher E:
  - $\Pr[E(m, h) \oplus h = h'] = \Pr[E(m, h) = h' \oplus h] = \Pr[E(m, h) = h''] = 1/2^n$ (pre-image resistance)
  - Also easy to show $\Pr[E(m, h) \oplus h = E(m', h') \oplus h'] = 2^{-n/2}$ (collision resistance) in the ideal cipher model

- Lesson 4. Nearly all hash function rationales or "security proofs" rely on the ideal cipher model
- Lesson 5. The digest size must be at least twice the block size of the underlying block cipher

2nd-Preimages with Davies-Meyer Compression Functions

- It is easy to find **fixed points** for the Davies-Meyer construction

$$E\,(m, h) \oplus h = h \Leftrightarrow E\,(m, h) = 0 \Leftrightarrow h = E^{-1}\,(m, 0)$$

- The Attack: Given a message $m = m_1 \ldots m_k$ compute $h = DaviesMeyerHash\,(m)$ (with E replacing DES) and

  do $j = 1$ **to** $2^{n/2} \Rightarrow s_j \leftarrow_\$ \{0,1\}^n; u_j = E\,(s_j, iv) \oplus iv$ **od**

  do $j = 1$ **to** $2^{n/2} \Rightarrow t_{j'} \leftarrow_\$ \{0,1\}^n; v_{j'} = E^{-1}\,(t_{j'}, 0) \oplus iv$ **od**

- By the Birthday problem for two lists with high probability there are j, j' with $u_j = v_{j'}$

- Then $DaviesMeyerHash\,(iv, m) = DaviesMeyerHash\,(iv, s_j\,t_{j'}) = DaviesMeyerHash\,(iv, s_j\,t_{j'}\,t_{j'}) = DaviesMeyerHash\,(iv, s_j\,t_{j'}\,t_{j'}\,t_{j'}) = \ldots$

- Conclusion: With Davies-Meyer 2nd pre-image resistance is no more expensive than collision resistance

MDC2: Widening the Block Size

- The Davies-Meyer enhancement can only provide collision resistance to $O(2^{64/2}) = O(2^{32})$ DES operations
- In 1987 IBM proposed MDC2 to obtain $O(2^{64})$ collision resistance

$MDC2\ (m)$

$\qquad m_1 \ldots m_k \leftarrow m;\ h_0 \leftarrow iv$

$\qquad \textbf{do}\ j = 1\ \textbf{to}\ k \Rightarrow$

$\qquad\qquad h_{left}\ h_{right} \leftarrow h_{j-1}$

$\qquad\qquad d \leftarrow DES(h_{left}, m_j) \oplus m_j$

$\qquad\qquad e \leftarrow DES(h_{right}, m_j) \oplus m_j$

$\qquad\qquad h_j \leftarrow d_{left}\ e_{right}\ e_{left}\ d_{right}$

$\qquad \textbf{od}$

$\qquad \textbf{return}\ h_m$

Discussion

- The construction $(h,m) \to DES(h, m) \oplus m$ offers the same collision and pre-image bounds as Davies-Meyer
  - Nearly an identical argument in the ideal cipher model
  - This is the **Matyas-Meyer-Oseas construction**
- Swapping the left and right digest halves is essential for security
  - Collisions could be found in $2^{32} + 2^{32} = 2^{33}$ instead of 2^{64} DES operations, because without the swap the digest is just the concatenation of digests from two independent hashes
- Steinberger proved MDC2 is collision resistant in 2007

Length Problems 1

- Let $m \in (\{0,1\}^{56})^+$, i.e., m is a string whose bit length is a multiple of 56

- For any string n it is easy to verify $hash\,(m\,n) = hash\,(hash\,(m)\,n)$ for each of the hash constructions we have considered

  - This is called a **length extension attack**

  - Length extension attacks succeed even if the attacker never sees m

- Length extension attacks indicate something is still missing from our construction

Length Problems 2

- Suppose the message digest of a hash function is n bits wide

- Consider the message $m = m_1 \ldots m_k$ for $k \geq 2^{n/2}$

- By the standard birthday problem there is at probability of at least 0.5 that at least two messages in $\{m_1, m_1\, m_2, m_1\, m_2\, m_3, \ldots, m_1 \ldots m_k\}$ collide.

- Lesson 6. To achieve collision resistance the length of all the combined inputs to a hash function must be less than $2^{n/2}$ bits

Early Years Summary

- The Davies-Meyer hash is too weak for practical applications

  – Collisions found in 2^{32} DES operations

- The MDC2 hash is too expensive for practical use

  – 1 DES operation $\approx$ 500 cycles; 1 MDC2 operation $\approx$ 1000 cycles = 125 cycles *per byte*

- There is something wrong in the way early hash functions deal with the length of their inputs

- Question: Even though the inner loop is collision/pre-image/2nd pre-image resistant, why do we believe the hash function is?

Revolution

- At Crypto 1989 Merkle and Damgård published papers revolutionizing hash function design

- Replace the *DES* construction by a clean **compression function** abstraction $compress : \{0,1\}^s \times \{0,1\}^n \rightarrow \{0,1\}^n$ operating on s bit message blocks and an n bit **chaining variable**

- Define a padding scheme to block length extension attacks

- Because it blocks length extension attacks, the padding scheme extends compression function's collision resistance to the entire hash function

$MD\text{-}Hash\ (m)$

$m' \leftarrow pad\ (m);\ m_1 \ldots m_k \leftarrow m';\ h_0 \leftarrow iv$

$\textbf{do}\ j = 1\ \textbf{to}\ k \Rightarrow h_j \leftarrow compress(h_{j-1}, m_j)\ \textbf{od}$

$\textbf{return}\ h_m$

Merkle-Damgård Padding

- If the compression function *compress* operates on s bit message blocks and n bit chaining variables then

  $pad\ (m)$

  $\quad l \leftarrow |m|$ -- find m's length in bits

  $\quad t \leftarrow s - (l\ \textbf{mod}\ s) - n/2 - 1$ -- compute number of 0

  $\quad \textbf{if}\ t < 0 \Rightarrow t \leftarrow s + t\ \textbf{fi}$ --   bits needed

  $\quad m' \leftarrow m\ 1\ 0^t <l>_{n/2}$ -- append a 1 bit, t 0 bits, and l

  $\quad \textbf{return}\ m'$ --   encoded as an $n/2$ bit integer

- Key property: $pad\ (m)$ gives the number of bits l of m

- This scheme makes it unambiguous where the message m ends and where the padding ends

Collision Resistance

- Why does collision resistance of *compress* imply collision resistance of *md-hash*?

- Suppose we can easily find $m \neq m'$ with *md-hash* $(m) =$ *md-hash* (m')

- Two cases: *md-hash* $(m) =$ *md-hash* (m') with $|m| = |m'|$, $m = m_1 m_2 \ldots m_k$, $m' = m_1' m_2' \ldots m_i'$

- Case 1: Since $|m| = |m'|$, we know $k = i$ and the last block (of padding) is the same ($m_k = m_i$). There must be some $1 \leq j < k$ such that *compress* $(h_{j-1}, m_j) =$ *compress* (h_{j-1}', m_j') but $m_j \neq m_j'$. This contradicts the assumption it is hard to find collisions for *compress*

- Case 2: Since $|m| \neq |m'|$ we know that the final (padding) blocks $m_k \neq m_i'$ and *compress* $(h_{k-1}, m_k) =$ *compress* (h_{i-1}', m_i'), a contradiction since it is hard to find collisions for *compress*

Example: SHA-1

algorithm SHA-1 (M)

    return sha-md (5a827999 || 6ed9eba1 || 8f1bbcdc || ca62c1dc, M)

algorithm sha-md (K, M)

    M := pad (M)

    parse M into 512-bit blocks $M_1 \ldots M_k$

    IV := 67452301 || efcdab89 || 98badcfe || 10325476 || c3d2e1f0

    do $i = 1$ **to** $k \Rightarrow IV$:= sha-compress (K, IV, M_i) **od**

    return IV

Merkle-Damgård construction

algorithm sha-compress (K, IV, M)

    parse K into 32-bit blocks $K_1 \ldots K_4$ and IV into $IV_1 \ldots IV_5$

    parse M into 32-bit blocks $W_1 \ldots W_{16}$

    do $i = 17$ **to** $80 \Rightarrow W_i$:= LROT ($1, W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}$) **od**

    $A := IV_1, B := IV_2, C := IV_3, D := IV_4, E := IV_5$

    do $i = 1$ **to** $20 \Rightarrow L_i := K_1, L_{i+20} := K_2, L_{i+40} := K_3, L_{i+60} := K_4$ **od**

    do $i = 1$ **to** $80 \Rightarrow$

        if $1 \le i \le 20 \Rightarrow f := (B \wedge C) \vee ((\neg B) \wedge D)$

        else if $41 \le i \le 80 \Rightarrow f := (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$

        else $f := B \oplus C \oplus D$ **fi**

        $t := $ LROT ($5, A$) $+ f + E + W_i + L_i$

        $E := D, D := C, C := $ LROT ($30, B$), $B := A, A := t$

    od

    $IV_1 := IV_1 + A, IV_2 := IV_2 + B, IV_3 := IV_3 + C, IV_4 := IV_4 + D, IV_5 := IV_5 + E$

    return $IV_1 || IV_2 || IV_3 || IV_4 || IV_5$

Block cipher key schedule

Block cipher

Davies-Meyer feed-forward

Structural Problems

- Second pre-image attacks
- Random Mapping properties
- Multi-block Differential Attacks

Joux's Multi-collision Attack

- Let $compress : \{0,1\}^s \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a collision resistant compression function and $m_1\, m_2$ be a $2s$ bit message

- By assumption we can find $m_1' \neq m_1$ such that $compress\,(iv, m_1) = compress\,(iv, m_1')$ in $2^{s/2}$ operations
- Similarly we can find $m_2' \neq m_2$ such that $compress\,(iv, m_2) = compress\,(iv, m_2')$ in $2^{s/2}$ operations
- Therefore $m_1\, m_2'$, $m_1'\, m_2$, and $m_1'\, m_2'$ are **three** 2nd preimages of $m_1\, m_2$ under md-hash that we have found in $2^{s/2} + 2^{s/2} = 2^{s/2+1}$ operations instead of 2^s

- Clearly the attack can be extended to k block messages to find 2^{k-1} 2nd pre-images in time $k2^{s/2}$ instead of 2^s

- Conclusion: 2nd pre-image resistance from the Merkle-Damgård construction is no stronger than collision resistance

The Random Mapping Property

- A **random oracle** is a public random mapping
  - A random oracle returns a fixed length random string in response to any input
- It is widely assumed in practice that hash functions behave like random oracles
- Let $m = m_1 \, m_2$. Then it is easy to see that

$$md\text{-}hash(m_1 \, pad(m_1) \, m_2) = md\text{-}hash(md\text{-}hash(m_1) \, m_2)$$

- If *hash* acted like a random oracle, then *hash* (*m pad (m) n*) and *hash* (*hash* (*m*) *n*) should assume independent values
- This makes Merkle-Damgård hash functions hard to use in practice
  - We don't know that constructions using Merkle-Damgård hash functions deliver the security claimed
- Merkle-Damgård hash functions leak that they are iterative constructions

Random Oracles

- D. Simon showed that random oracles cannot be instantiated
  - Random oracles assume an infinite world, so can always be distinguished from real-word constructions

- Maurer introduced the notion of **indifferentiability** to replace the notion of distinguishability when reasoning about hash functions

- Collision resistance is not enough; hash functions should be indifferentiable from random oracles

Indifferentiability

- Question: When can an iterated construction replace a monolithic construction?
- Answer: When for every adversary a simulation environment exists wherein the adversary cannot distinguish the real construction from the monolithic construction operating in the simulation

Relationships

Random oracle

Collision resistance

2nd pre-image resistance

+

Non-trivial compression

Pre-image resistance
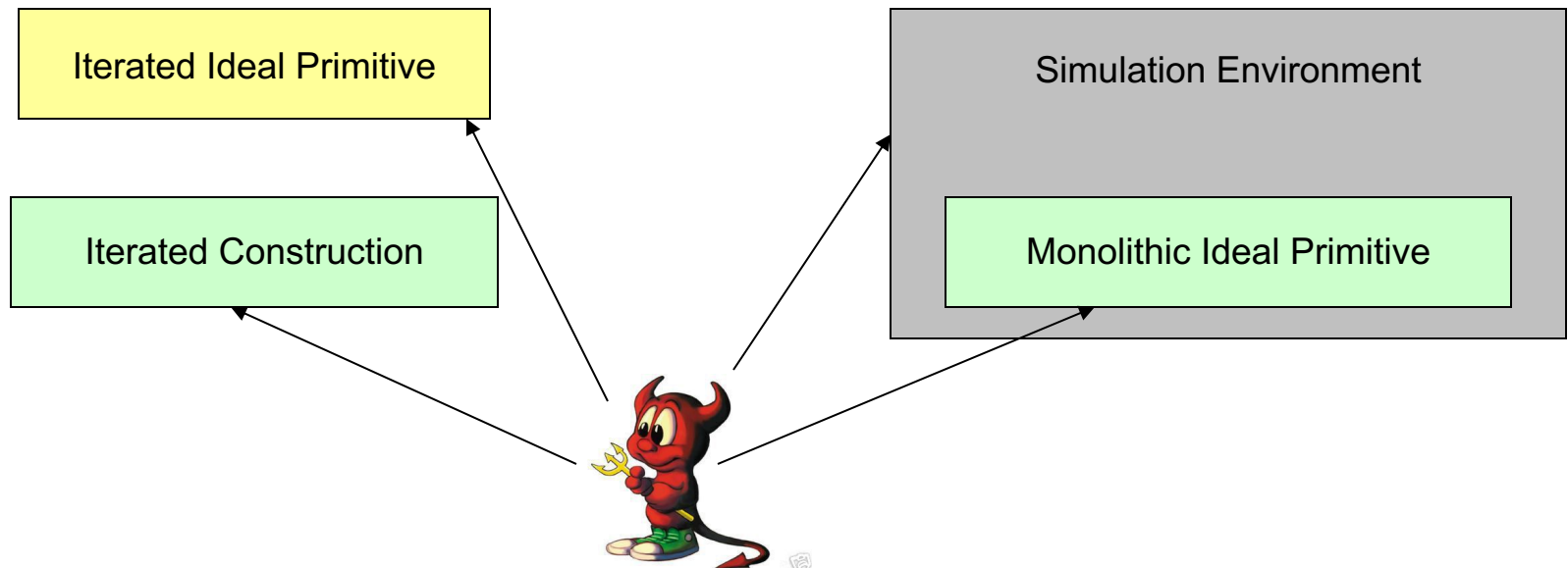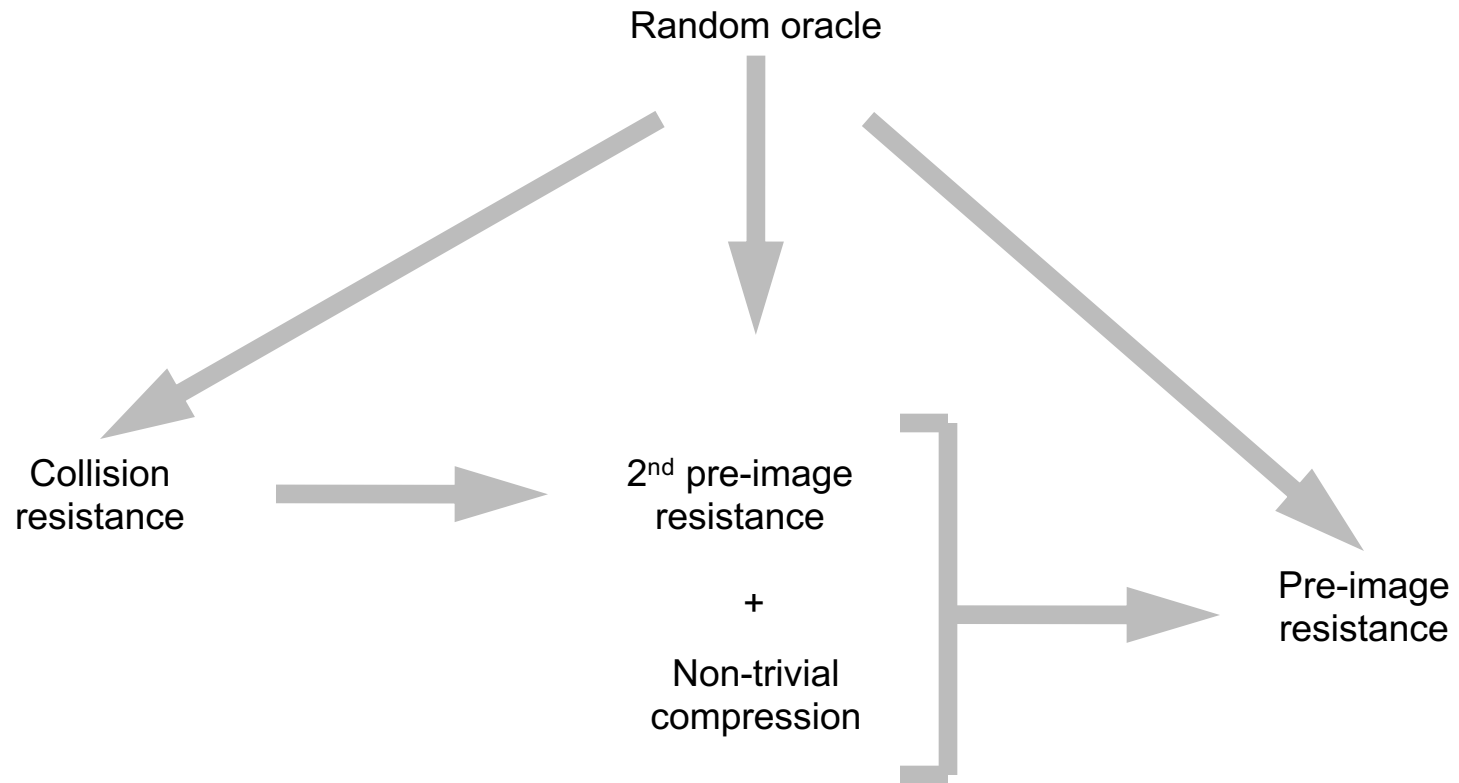
Multi-block Differential Attacks

- Differential cryptanalysis was introduced to study block ciphers

- Given a key K and X, X' with difference $X \oplus X'$, what is the difference $E(K,X) \oplus E(K,X')$?

- This often yields useful information about K and deep insight into E's structure

- Since compression functions for Merkle-Damgård hashing are based on block ciphers, there should be some way to extend differential cryptanalysis to hashing

  - Since hashing is multi-block, we need some way to extend differential cryptanalysis to multi-block attacks

The Multi-Block Technique

Step 1. Find a **near collision** $m \neq m'$ producing a designated output difference $h \approx h'$
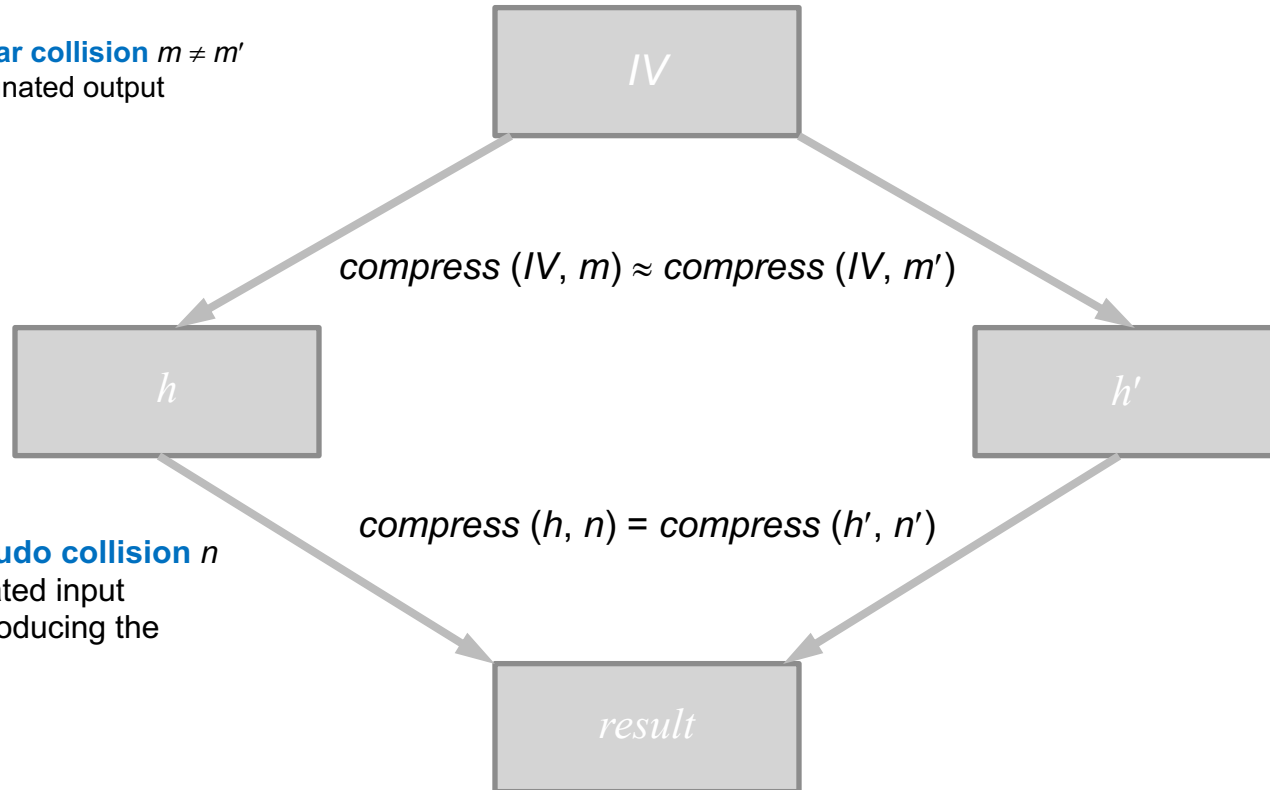
Step 2. Find a **pseudo collision** $n \neq n'$ from a designated input difference $h \approx h'$ producing the same result

IV

$compress\ (IV,\ m) \approx compress\ (IV,\ m')$

h

h'

$compress\ (h,\ n) = compress\ (h',\ n')$

$result$

$m\ n$ and $m'\ n'$ are colliding messages when successful

Wang's Attack

- In 2004 Xiayuan Wang applied the multi-block technique to break the collision resistance of MD4, MD5, and Ripe-MD
  - In 2009 their attack was extended to forge the certificate of real CA that supported MD5
- In 2005 Wang and colleagues used the technique to defeat the collision resistance of SHA-1
  - They showed a collision could be found at cost 2^{62} instead of 2^{80} operations
- These attacks caused deep trauma and introspection in the crypto community
  - "Do we know what a hash function is?"

What Went Wrong?

algorithm SHA-1 (M)

       return sha-md (5a827999 || 6ed9eba1 || 8f1bbcdc || ca62c1dc, M)

algorithm sha-md (K, M)

       M := pad (M)

       parse M into 512-bit blocks $M_1 \ldots M_k$

       IV := 67452301 || efcdab89 || 98badcfe || 10325476 || c3d2e1f0

       do $i = 1$ **to** $k \Rightarrow IV$:= sha-comp ($K. IV, M$) **od**

       return IV

algorithm sha-comp (K, IV, M)

       parse K into 32-bit blocks $K_1 \ldots K_4$ **and** IV **into** $IV_1 \ldots IV_5$

       parse M into 32-bit blocks $W_1 \ldots W_{16}$

       do $i = 17$ **to** $80 \Rightarrow W_i$:= LROT ($1, W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}$) **od**

       $A := IV_1, B := IV_2, C := IV_3, D := IV_4, E := IV_5$

       do $i = 1$ **to** $20 \Rightarrow L_i := K_1, L_{i+20} := K_2, L_{i+40} := K_3, L_{i+60} := K_4$ **od**

       do $i = 1$ **to** $80 \Rightarrow$

              if $1 \leq i \leq 20 \Rightarrow f := (B \wedge C) \vee ((\neg B) \wedge D)$

              else if $41 \leq i \leq 80 \Rightarrow f := (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$

              else $f := B \oplus C \oplus D$ **fi**

              t := LROT ($5, A$) $+ f + E + W_i + L_i$ Poor diffusion

              $E := D, D := C, C :=$ LROT ($30, B$), $B := A, A := t$

       od

       $IV_1 := IV_1 + A, IV_2 := IV_2 + B, IV_3 := IV_3 + C, IV_4 := IV_4 + D, IV_5 := IV_5 + E$

       return $IV_1 || IV_2 || IV_3 || IV_4 || IV_5$

Key schedule doesn't resist related key attacks or compensate for cipher's poor diffusion

68

Discussion

- Davies-Meyer elevates the importance of related key attacks in block cipher designs, because the attacker has control over differences between encryption key
  - The block being hashed is the encryption key
  - The attacks exploit the fact that making small changes in one block can be canceled by a later block
- We have learned that hash functions and block ciphers are attacked in similar ways
  - No longer surprising, given how hash function have been built
- All of the state-of-the-art design techniques for design and validation of block ciphers should be applied to hash function designs
  - e.g., show that every input bit flows to every output bit after a few rounds

The Merkle-Damgård Years

- Merkle-Damgård theory finally puts collision resistance, 2nd pre-image resistance, and pre-image resistance on a firm foundation

- Merkle-Damgård 2nd pre-image is much weaker than anticipated

- Merkle-Damgård hash functions do not act like random oracles
  - So we don't know many of our constructions are safe

- The Multi-block technique appears to threaten Merkle-Damgård designs

SHA-3 and Modern Hash Function Construction

- The SHA-3 competition

- HAIFA

- Domain Switching

- The Sponge Construction

- And the winner is . . .

The SHA-3 Competition

- NIST adopted the SHA-2 family in 2003
  - Block sizes of 224, 256, 384, and 512 bits to address Moore's Law

- Design of SHA-2 family very similar to that for SHA-1
  - Is SHA-2 vulnerable to Wang's attack? No, but this was not established until after SHA-3 competition was under way

- Due to similarity of SHA-2 family to SHA-1, consensus was we need a new hash algorithm design

- Crypto community's BKM for designing new algorithms: hold a contest

- NIST published RFP January 7, 2007 announcing competition

- Submissions due October 31, 2007, with 64 designs received

The SHA-3 Competition

- NIST accepted 51 of the 64 submissions into Round 1
- Extensive cryptanalysis of all designs by the international community
  - All designs independently analyzed by multiple parties
  - Majority of designs broken
- Extensive performance data collected at the e-BACS site
- NIST selected 14 designs for Round 2 in July 2009
- NIST selected 5 finalist algorithms in December 2010

Round 2 Candidates and Finalists

| Candidate	Designer Origins	Design Type
BLAKE	🇨🇭 🇬🇧	**ARX, HAIFA**
Blue Midnight Wish	🇳🇴	ARX, MD + FT
CubeHash	🇺🇸	ARX, MD + FT
ECHO	🇫🇷	AES, HAIFA
Fugue	🇺🇸	AES, MD + FT
Grøstl	🇦🇹 🇧🇪 🇩🇰 🇵🇱	**AES, MD + FT**
Hamsi	🇧🇪	s-box, MD + FT
JH	🇸🇬	**s-box, Sponge**
Keccak	🇧🇪 🇮🇹	**s-box, Sponge**
Luffa	🇧🇪 🇯🇵	s-box, MD + FT
Shabal	🇫🇷	Mix, MD
SHAvite-3	🇮🇱	AES, HAIFA
SIMD	🇫🇷	Mix, MD + FT
Skein	🇺🇸 🇩🇪	**ARX, MD(+ FT)**

Addressing Merkle-Damgård Weaknesses

- 3 Approaches proposed
  - The HAIFA construction
  - Domain switching (aka "Final Transform")
  - The Sponge construction
- HAIFA and domain switching patch Merkle-Damgård, while a sponge is something entirely new
- All five finalists employ one or more of these approaches
- All five finalists appear to have comparable security levels
  - Significantly better safety margins than SHA-2
  - All are indifferentiable from random oracles

HAIFA Construction

- Developed by Biham and Dunkleman

- Idea: hash each message block through the compression function with the number of bits hashed so far and an optional salt

- Intuition: This makes each compression function invocation independent

- Theoretical foundation:

  - The mapping $m \rightarrow (0, m_1) \, (s, m_2) \, (2s, m_3) \ldots \, ((k{-}1)s, m_k)$ is a **prefix-free encoding** of m

  - Coron et al proved that the Merkle-Damgård hash of a prefix-free encoded message is indifferentiable from a random oracle

HAIFA Example: Skein's UBI Construction
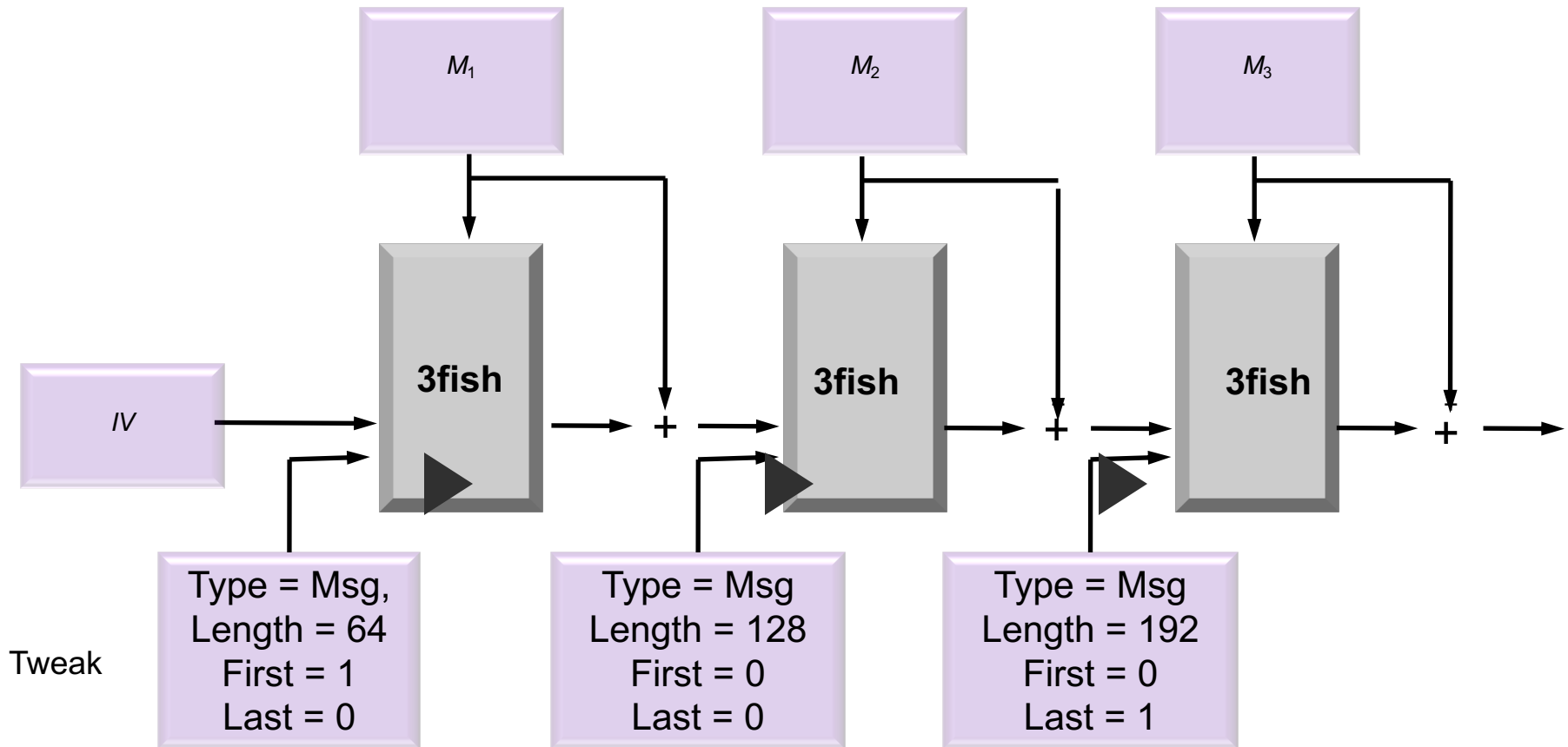
Domain Switching

- Developed by Bellare and Ristenpart
- Idea: Rehash the output from Merkle-Damgård under an independent compression function
- Intuition: Hide the iterative structure with an independent hash ("domain switch")
- Theoretical foundation:
  - If the compression function acts like a random oracle, then so is a Merkle-Damgård digest after being post-processed in this way
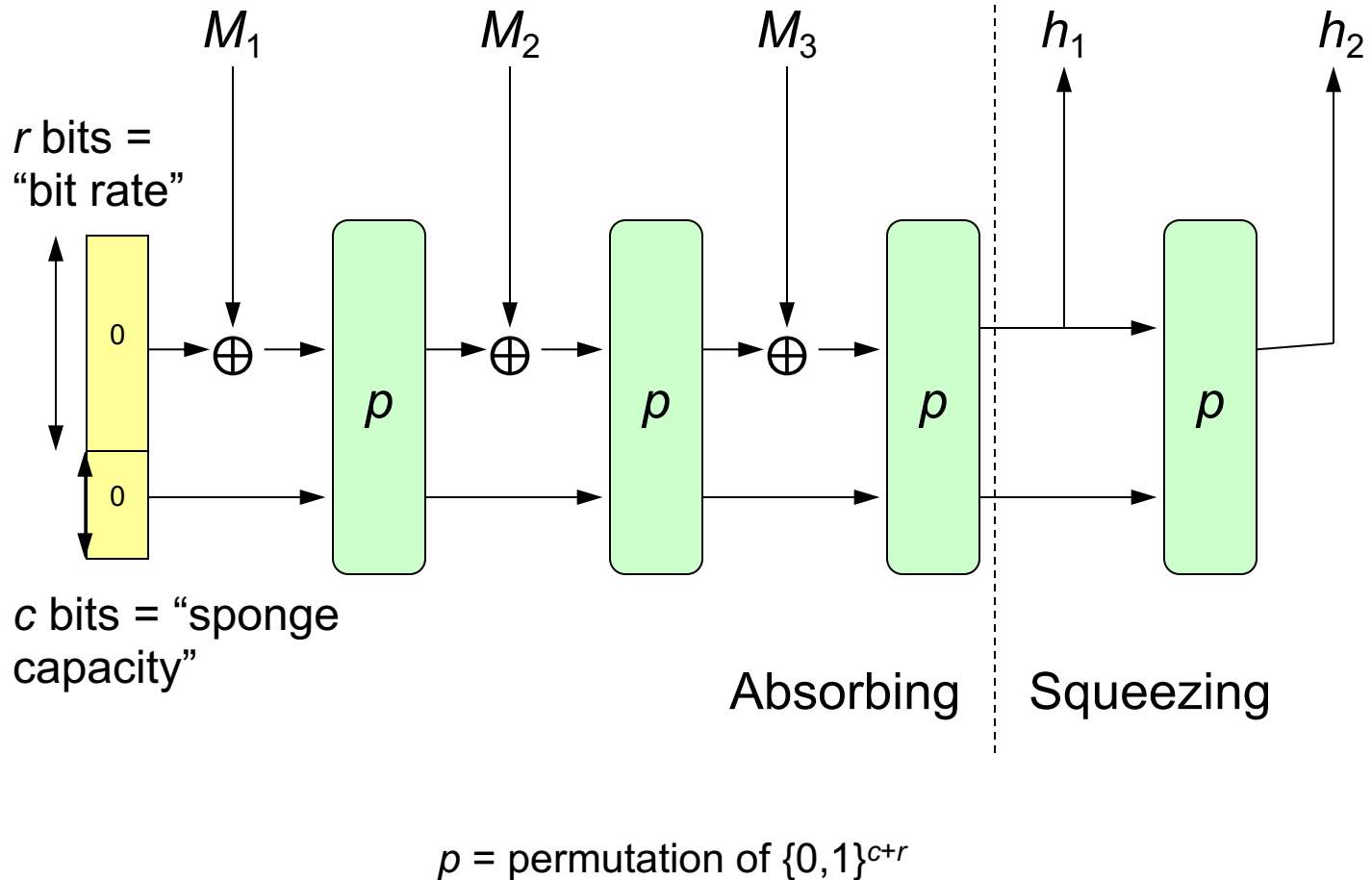
Domain Switching Example: Grøstl

Message hashed in 1st domain

Digest rehashed in a 2nd domain
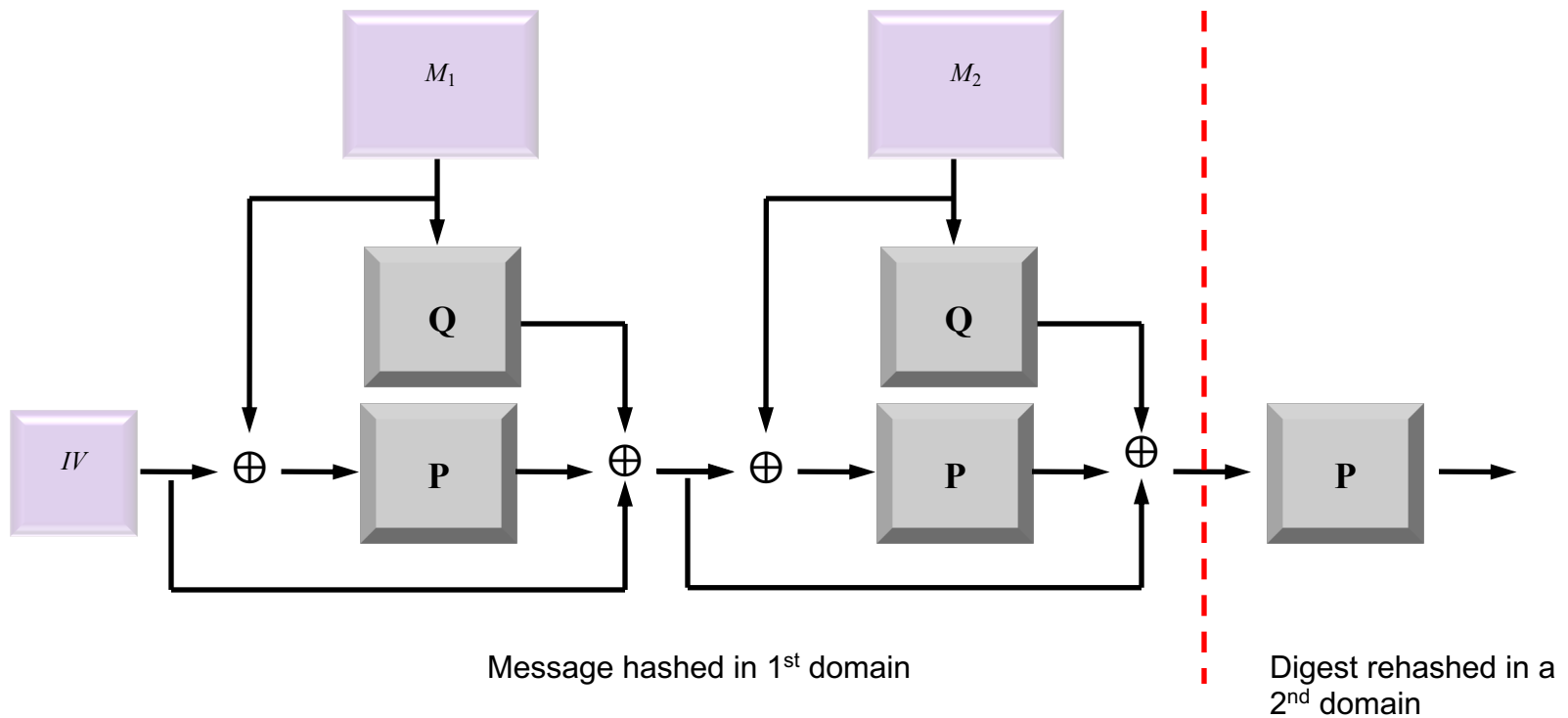
The Sponge Construction

- Developed by Bertoni, Daemen, Peeters, and Van Assche
- Idea: We don't know the right design criteria except that a hash function act like a random oracle, so make the design act as much like a random oracle as possible
- Intuition: A permutation with a large state space, only some of which can be updated by the environment, acts like a random oracle
- Theoretical foundation:
  - Can prove a sponge is indifferentiable from a random oracle

The Sponge Construction

M_1 $\qquad$ M_2 $\qquad$ M_3 $\qquad$ h_1 $\qquad$ h_2

r bits = "bit rate"

c bits = "sponge capacity"

Absorbing | Squeezing

p = permutation of $\{0,1\}^{c+r}$

And the Winner is . . .

- **Keccak**

- Keccak was designed by Guido Bertoni, Joan Daemen, Michael Peeters, Gilles Van Assche
  - Joan Daemen and Vincent Rijman designed AES

- NIST announced the SHA-3 winner on October 2, 2012
  - AES winner announced on October 2, 2000

- NIST indicated design diversity drove their choice
 - SHA-2, BLAKE, Grøstl, Skein are Merkle-Damgård based

High Level Design

- Keccak uses a 24 round permutation in the sponge construction
- Keccak's permuation is called $\text{Keccak-}f$ and parameterized by rate r and capacity c
  - $r+c = 1600 = 25 \times 64$
    - Keccak-512: $r = 512$, $c = 1088 \Rightarrow$ faster with 2^{544} security bound
    - Keccak-256: $r = 256$, $c = 1344 \Rightarrow$ slower with 2^{672} security bound
- Design goal: $\text{Keccak-}f$ has no exploitable properties
- $\text{Keccak-}f$'s design based on the **wide-trail** design strategy
  - Spread a round's non-linear across across the entire round using well-chosen linear transformations to get provable resistance to linear and differential cryptanalysis
- $\text{Keccak-}f$ round: $\iota \circ \chi \circ \pi \circ \rho \circ \theta(state) = \iota(\chi(\pi(\rho(\theta(state)))))$
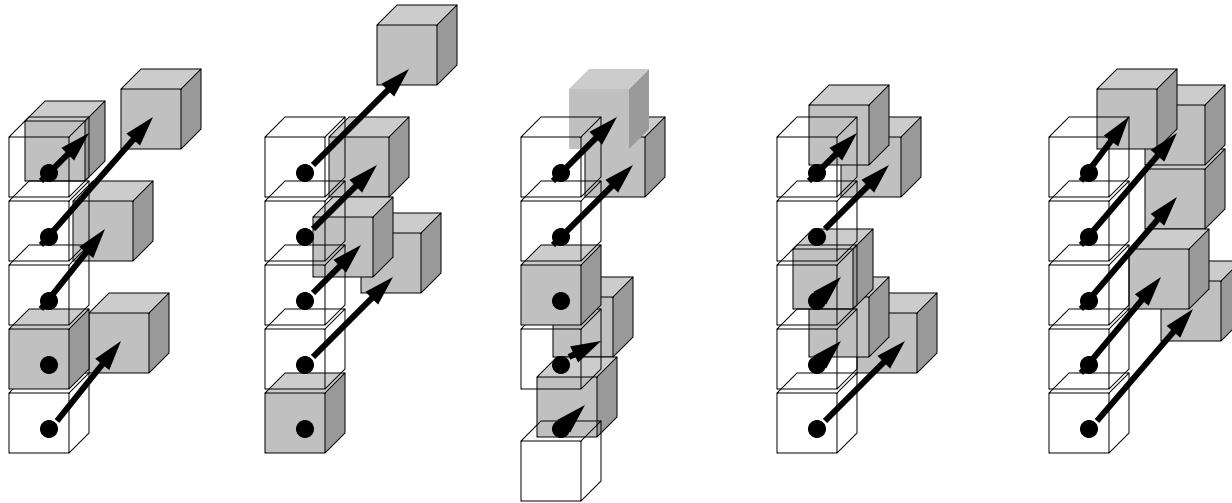
Keccak State
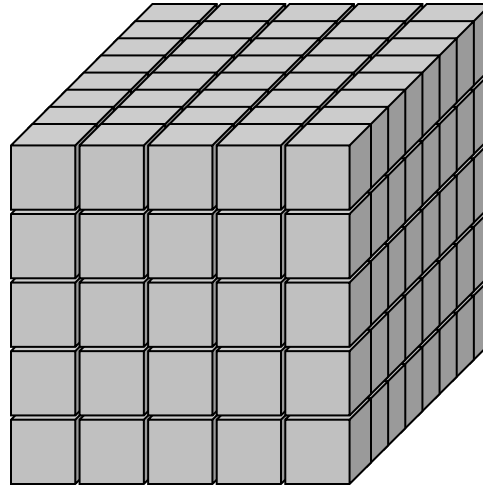
- Keccak represents its 1600 bit state as a $5 \times 5 \times 64$ bit cube
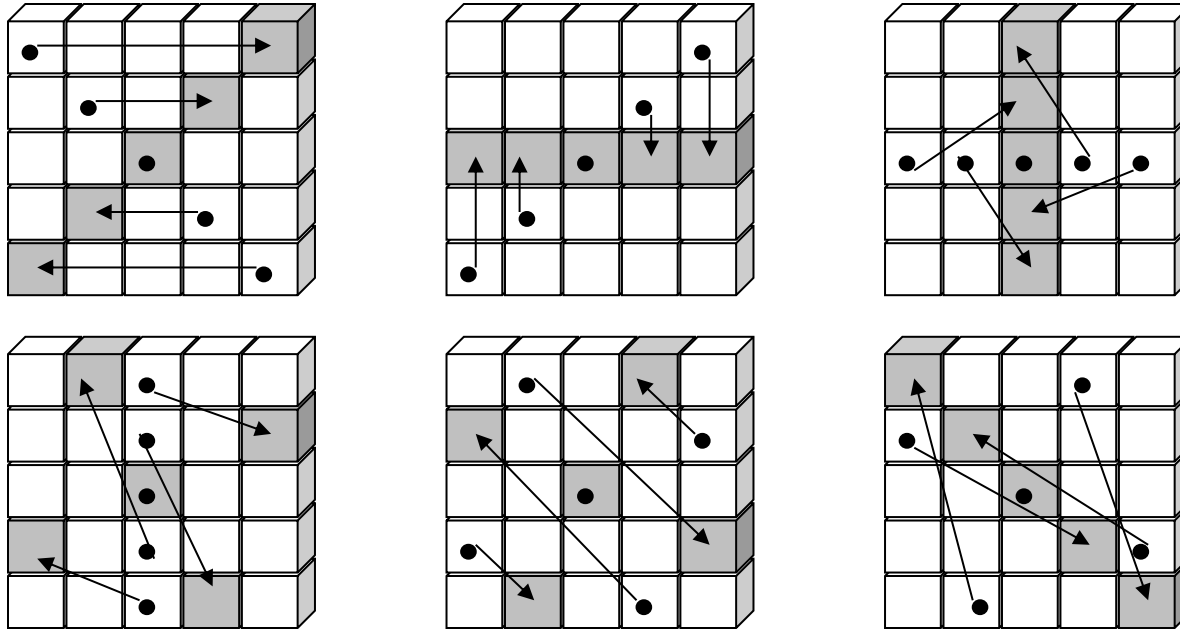
The Keecak θ Function
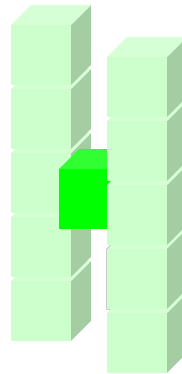
- ι°χ°π°ρ°θ(*state*) = 1 of 24 Keccak-*f* rounds
- θ provides diffusion – each bit affects 11 adjacent bits
- θ implemented by 50 XORs and 5 rotations
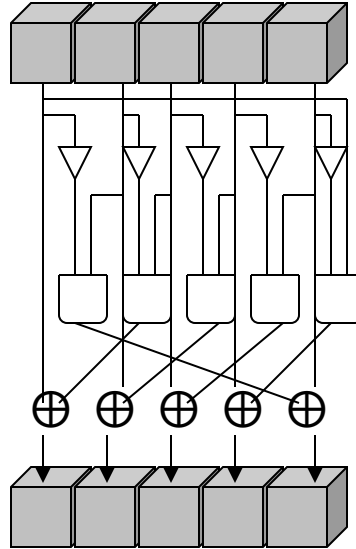
The Keccak ρ Function

- $\iota \circ \chi \circ \pi \circ \rho \circ \theta(state) = 1$ of 24 Keccak-f rounds
- ρ provides inter-slice dispersion by moving 25 bits of a slice to 25 different slices
- Implemented by 24 rotations

The Keccak π Function

- $\iota \circ \chi \circ \pi \circ \rho \circ \theta (state) = $ 1 of 24 Keccak-f rounds
- π distributes horizontal/vertical alignment using a period 24 cycle about a fixed origin
- Implemented as a linear mapping of $GF(5) \times GF(5)$

The Keccak χ Function

- $\iota \circ \chi \circ \pi \circ \rho \circ \theta(\mathit{state}) =$ 1 of 24 Keccak-f rounds
- χ provides non-linearity
- Note it is a Feistel construction

The Keccak ι Function

- $ι°χ°π°ρ°θ(state) =$ 1 of 24 Keccak-f rounds

- ι breaks symmetry, to
  - Defend against slide attacks
  - Reduce the effectiveness of cross-round attacks

- Implemented by adding a round constant to state

SHA-3 Summary

- All of the SHA-3 finalists offer excellent security

- Design diversity drove NIST's selection of Keccak as the SHA-3 winner

- Keccak is indifferentiable from a random oracle, and so meets any conceivable hash function requirement

Key Takeways

- Cryptographic hash function design has deep roots in conventional computer science, but only received a firm foundation with Merkle-Damgård

- Identifying the right problems to solve has been a treacherous adventure

- New hash function designs should strive to construct random oracles

- Keccak is a worthy winner of the SHA-3 competition

Suggested Reading

- B. Preneel, *Analysis and Design of Cryptographic Hash functions*, Ph.D. thesis
- J. Black, P. Rogaway, and T. Shrimpton, *Black-Box Analysis of Block-Cipher-Based Hash Function Constructions from PGV*, Crypto 2002, pp 320-355
- P. Rogaway and T. Shrimpton, *Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance*, FSE 2004, pp 371-388
- R. Merkle, *One way hash functions and DES*, Crypto 1989, pp 228-246
- I. Damgård, *A Design Principle for Hash Functions*, Crypto 1989, pp 416-427
- J.S. Coron, Y. Dodis, C. Malinaud, and P. Puniya. *Merkle-Damgard Revisited: How to Construct a Hash Function*, Crypto 2005, pp 21-39
- X. Wang and H. Yu. *How to Break MD5 and Other Hash Functions*, EuroCrypt 2005, pp 19-35

Suggested Reading

- M. Bellare, and T. Ristenpart, *Multi-Property-Preserving Hash Domain Extension and the EMD Transform*, AsiaCrypt, 2006

- S. Lucks, *A Failure-Friendly Design Principle for Hash Functions*, AsiaCrypt 2005

- J. Black, M. Cochran, and T. Shrimpton, *On the Impossibility of Highly Efficient Blockcipher-Based Hash Functions*, Eurocrypt 2005, pp 526-541

- A. Joux, *Multicollisions in Iterated Hash Functions: Application to Cascaded Constructions*, Crypto 2004

- E. Biham, and O. Dunklemann, *A Framework for Iterative Hash Functions – HAIFA*, eprints 2007/278

- G. Berton, J. Daemen, M. Peeters, and G. Van Gilles, *On the Indifferentiability of the Sponge Construction*, EuroCrypt 2008

- U. Maurer, R. Reener, and C. Holenstein, *Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology*, TCC 2004, pp 21-39

Suggested Reading

- J.P. Aumasson, L. Henzen, W. Meier, and R. Phan, *SHA-3 Proposal BLAKE*, https://131002.net/blake/blake.pdf
- L. Gauravaram, Knudsen, K. Matusiewicz, C. Rechberger, M. Shläffer, and S. Thomsen, *Grøstl – a SHA-3 Candidate*, http://www.groestl.info/Groestl.pdf
- H. Wu, *The Hash Function JH*, http://www3.ntu.edu.sg/home/wuhj/research/jh/jh_round3.pdf
- G. Bertoni, J. Daemen, M. Peeters, and G. Van Gilles, *The Keccak SHA-3 submission*, http://keccak.noekeon.org/Keccak-submission-3.pdf
- N. Ferguson, S. Lucks, B. Schneier, D. Whiting, M. Bellare, T. Kohno, J. Callas, J. Walker, *The Skein Hash Function Family*, http://www.skein-hash.info/sites/default/files/skein1.1.pdf

End