Selected Areas in Cryptology
Cryptanalysis
Week 4

Marc Stevens

stevens@cwi.nl

https://homepages.cwi.nl/~stevens/mastermath/

Cryptographic Hash Functions

Theoretical Cryptology:

Hash function family with the same range $\mathcal{H}$ (e.g. $\{0,1\}^{256}$)

$$\mathcal{F} = \{f: \{0,1\}^* \to \mathcal{H}\}$$

Security games for any PPT adversary A

- Pre: *pre-image resistance*: $f \leftarrow \mathcal{F}, h \leftarrow \mathcal{H}$, wins if $M \leftarrow A(f, h)$ and $f(M) = h$
- ePre: *everywhere Pre*: $h \leftarrow A; f \leftarrow \mathcal{F}$ instead
- aPre: *always Pre*: $f \leftarrow A; h \leftarrow \mathcal{H}$ instead
- Sec: *2nd preimage resistance*: $f \leftarrow \mathcal{F}, M \leftarrow \{0,1\}^{\leq n}$, win if $M' \leftarrow A(f, M), f(M) = f(M')$ and $M \neq M'$
- eSec: *everywhere Sec*: $M \leftarrow A; f \leftarrow \mathcal{F}$ instead
- aSec: *always Sec*: $f \leftarrow A; M \leftarrow \{0,1\}^{\leq n}$ instead
- Coll: *collision resistance* $f \leftarrow \mathcal{F}$, win if $M, M' \leftarrow A(f)$ and $f(M) = f(M')$ and $M \neq M'$

Cryptographic Hash Functions

- Key-less Symmetric functionality, and has many applications!
- Often called the swiss knife of cryptography

- Inside construction of
  - MACs: HMAC, KMAC, …
  - digital signatures: hash-then-sign, LMS, XMSS, SPHINCS+
  - Password authentication
  - Blockchain proof-of-work, Blockchain addresses
  - …

Cryptographic Hash Functions

Hash function standards $H: \{0,1\}^* \rightarrow \{0,1\}^n$:

- MD5: 128-bits hash function published in 1992
  - Widely used till ~~2010
  - Broken in 2004: first collision found [WY05],
- SHA-1: 160-bit hash function published in 1995
  - Widely used even today (TLS1.2, Git, …)
  - ‘Broken’ in 2005: first theoretical collision attack [WYY05]
    practical attack in 2017: first collision [SBKAM17]
- SHA-2 family: 224/256/384/512-bit hash functions published in 2001
- SHA-3 family: 224/256/384/512-bit hash functions published in 2015

Cryptographic hash functions

Fixed n-bit hash functions: $f: \{0,1\}^* \rightarrow \{0,1\}^n$

- Pre, ePre, Sec, eSec, Coll security notions ill-defined

- aPre: always pre-image resistance:
  - Given random $h \leftarrow \{0,1\}^n$ find M s/t $f(M) = h$
- aSec: always second pre-image resistance:
  - Given random $M \leftarrow \{0,1\}^{\leq n}$ find $M' \neq M$ s/t $f(M) = f(M')$

- Secure if there is no attack faster than a generic attack

Generic pre-image attacks

- Generic (2nd) pre-image attack
  - Given any hash output h, find x such that $f(x) = h$
  - Algorithm
    1. Define message space $\mathcal{M}$ with $|\mathcal{M}| \geq |\mathcal{H}|$
    2. Sample $x \leftarrow \mathcal{M}$
    3. If $f(x) \neq h$ then go to step 2
    4. Return x
  - Each attempt is Bernoulli trial with $p = 2^{-|h|}$
  - $\Rightarrow$ Time: Geometric Distribution with $p = 2^{-|h|} \Rightarrow$ average time: $2^{|h|}$

- Low-entropy pre-image attack
  - Old practice: store password hashes & compare hash to authenticate
  - Problem: password space has low entropy
  - E.g. there are only $2^{47.7}$ alphanumeric (a-zA-Z0-9) passwords of length ≤ 8.
  - $\Rightarrow$ Can use Hellman's time-memory trade-off attack to invert arbitrary function
  - Solution: *salting* each password p:
    - Choose random salt $s \leftarrow \{0,1\}^{64}$ to prepend to password
    - Store salt & hash: $(s, f(s|p))$

Collision conundrum

How to define collision resistance for fixed hash functions?

Mathematical existential security definitions?:

> "There should exist no attack that is feasible/faster than generic attack/PPT that finds a collision with non-negligible probability"

Conundrum:

- Pigeon-hole principle ⇒ collisions exist
- Any collision $f(M) = f(M')$ with $M \neq M'$ leads to a trivial attack:
  - Algorithm $A_{M,M'}$: simply outputs the pair M, M'
  - Such algorithms exist and break security definitions
  - However, we can't actually write down such algorithms unless we first compute a collision… (i.e., its non-uniform)

Foundations of Hashing Dilemma:

- No formal definition of collision resistance exists
- Informal definition relies on human ignorance:
- "There exists no *known* attack that is better than the generic collision attack"

Generic collision attack

- Generic collision attack

  For $i = 1, \ldots$

  Sample $M_i \leftarrow \{0,1\}^{\leq n}$, $h_i = f(M_i)$

  If $\exists j < i : h_j = h_i$ then return (M_j, M_i)

- Cost analysis:
  - Let X be the number of samples needed before a collision is found
  - $E[X] = \sum_{k=1}^{\infty} k \cdot \Pr[X = k] = \sum_{k=1}^{\infty} k \cdot (\Pr[X > k-1] - \Pr[X > k])$

    $= \sum_{k=0}^{\infty} (k+1) \cdot \Pr[X > k] - \sum_{k=1}^{\infty} k \cdot \Pr[X > k]$

    $= \sum_{k=0}^{\infty} \Pr[X > k]$
  - $\Pr[X > k] = 1 \frac{N-1}{N} \frac{N-2}{N} \ldots \frac{N-k+1}{N} = 1\left(1 - \frac{1}{N}\right)\left(1 - \frac{2}{N}\right) \ldots \left(1 - \frac{k-1}{N}\right)$ ("no collision after k samples")

    $\approx 1\, e^{-\frac{1}{N}} e^{-\frac{2}{N}} e^{-\frac{3}{N}} \ldots e^{-\frac{k-1}{N}} = e^{-\frac{k(k-1)}{2N}}$ $\qquad e^x = 1 + x + \frac{x^2}{2!} + \cdots$

    $\approx e^{-k^2 2^{-n-1}}$
  - Estimate:

    $E[X] \approx \sum_{k=0}^{\infty} e^{-k^2 2^{-n-1}} \approx \int_0^{\infty} e^{-k^2 2^{-n-1}}\, dk = \sqrt{\pi/2} \cdot 2^{n/2}$ time cost
  - Memory cost: $O(2^{n/2})$

Generic collision attack

- Memory cost improvements
  - Idea: compute trails and only store begin/end-points
    like Hellman's time-memory trade-off attack

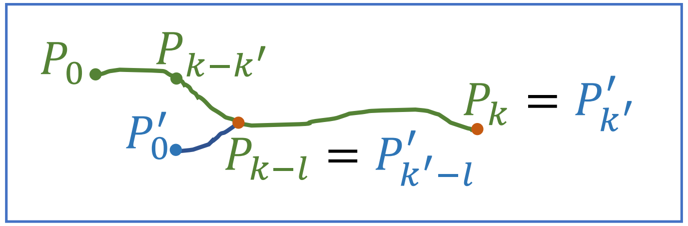
  - Define search space: $\mathcal{H} \coloneqq \{0,1\}^n$
  - Choose injective embedding $\phi: \mathcal{H} \rightarrow \{0,1\}^*$
    - Let $g \coloneqq f \circ \phi$: $\mathcal{H} \rightarrow \mathcal{H}$
    - $\Rightarrow$ a collision of $H \neq H'$ of g (i.e., $g(H) = g'(H)$) is a collision $\phi(H) \neq \phi(H')$ of f

  - Choose set of 'distinguished points' $S \subset \mathcal{H}$:
    - Easily distinguishable: e.g. last l-bits are zero

  - Compute trails:
    - Choose random starting point P_0
    - Iterate $P_i = g(P_{i-1})$ until a distinguishable point $P_i \in S$ is encountered
    - Then only store begin/end-point & length (P_0, P_i, i)

Generic collision attack

- Compute trails:
  - Choose random starting point P_0
  - Iterate g: $P_i = g(P_{i-1})$ until a distinguishable point $P_i \in S$ is encountered
  - Then only store begin/end-point & length (P_0, P_i, i)
- What happens when a collision occurs:
  - $P_i \neq P_j$ and $g(P_i) = g(P_j)$
  - Since g is deterministic, the two trails merge: $g^k(P_i) = g^k(P_j)$
  - $\Rightarrow$ End at the same distinguished point: $g^k(P_i) = g^k(P_j) \in S$
- Resolving a collision:
  - Consider two trails (P_0, P_k, k), $(P'_0, P'_{k'}, k')$ with $P_k = P'_{k'} \in S$ (wlog $k \geq k'$)
  - Assume collision occurs l iterations before end
  - First synchronize: iterate longest trail $k - k'$ iterations
  - Exceptional case: $P_{k-k'} = P'_0 \Rightarrow$ 'robin-hood' failure
  - Iterate for $i = k' - 1, \ldots, 0$:
    - If $P_{k-i} = P'_{k'-i}$ then return $\phi(P_{k-i-1}), \phi(P'_{k'-i-1})$



Generic collision attack

Memory cost

- Expected total evaluations before collision occurs:

  $E[X] = \sqrt{\pi/2} \cdot 2^{n/2}$

- Expected trail length $t \coloneqq |\mathcal{H}|/|S|$ (geometric distribution with $p = |S|/|\mathcal{H}|$)
- We expect $\approx \sqrt{\pi/2} \cdot 2^{n/2}/t$ trails to store
  - If S consists of points with last l-bits zero
  - then $t = 2^n/2^{n-l} = 2^l$ and $O\left(2^{n/2-l}\right)$ memory cost
- Additional costs:
  - Once a collision occurs, need to finish the trail: t evaluations (expected trail length is memoryless)
  - To compute the actual collision point: $2.5\ t$ evaluations (analysis see link in lecture notes)
  - Total $O(3.5t) = O(3.5 \cdot 2^l)$ time cost
- Suggested choice $l = n/2 - 20$
  - Memory cost $\approx$1M trails
  - Expected additional time cost: $O\left(2^{n/2}/2^{20}\right) \ll O\left(2^{n/2}\right)$

Generic collision attack

- See lecture notes for full collision attack algorithm
- Unlikely problematic case:
  - A trail enters a cycle without ever reaching a distinguished point
  - $\Rightarrow$ collision attack would loop forever
- Solution:
  - Discard trail if $20\ t$ iterations is reached
  - Discard case 1: no cycle reached
    - Probability $\left(1-\frac{1}{t}\right)^{20\,t} \approx \left(e^{-\frac{1}{t}}\right)^{20t} = e^{-20} \approx 2^{-29}$ ("no distinguished point found")
  - Discard case 2: cycle reached: internal collision
    - Probability: $1 - e^{-(20t)^2\ 2^{-n-1}}$ ("collision within $20t$ samples")
    - Need $(20t)^2 \ll 2^n$ for this probability to be small enough
  - Both negligible losses

Summary

- Cryptographic hash functions
  - Theoretical cryptography: hash function families
  - Practice: fixed hash function standards

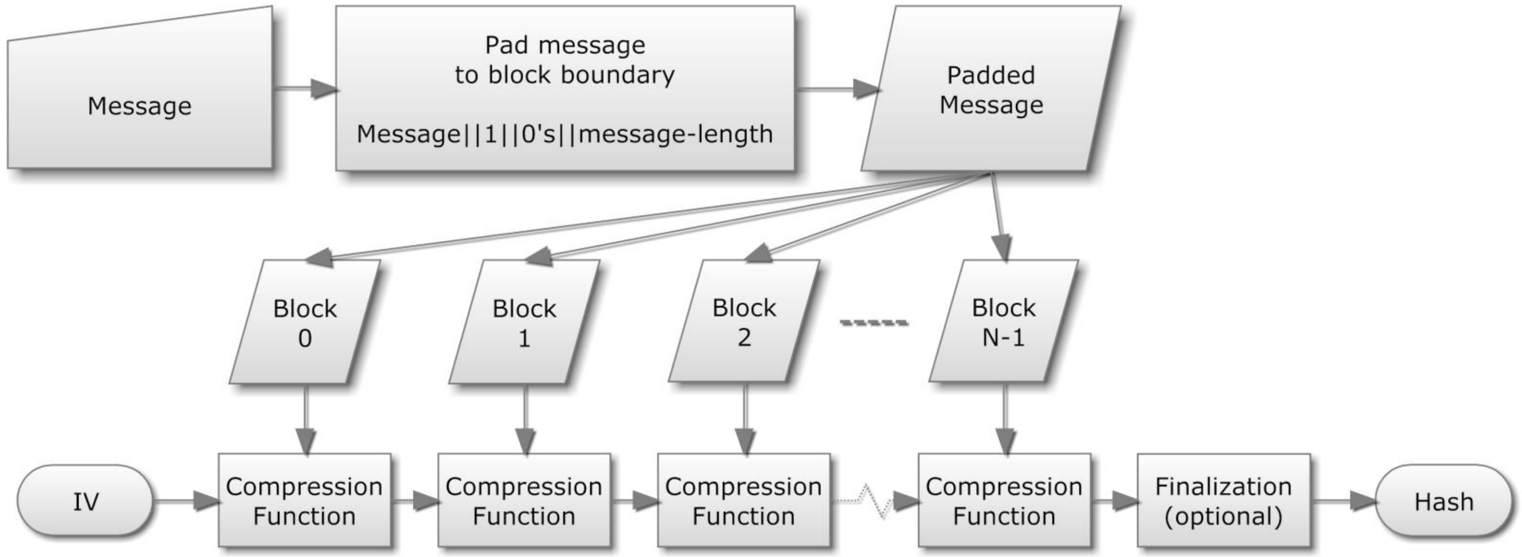
- Foundations of Hashing Dilemma:
  - No security definition possible for collision resistance for fixed hash functions
  - Informal definition: “no known attack”

- Generic collision attack:
  - Birthday paradox
  - Use trails and distinguished points to reduce memory cost

Merkle-Damgard Framework

- Merkle-Damgard Iterative Design:
  - Pad & split message M into pieces $M_1| \ldots |M_n$ (last block includes bitlength)
  - Internal state: CV_i with fixed initial value $CV_0 = IV$
  - Update internal state with compression function

    $$CV_i = Compress(CV_{i-1}, M_i)$$



- Many standards: MD4, MD5, SHA-1, SHA-2-224/256/384/512

Merkle-Damgard Framework

- Reduction proof:
  - Given a collision $f(M) = f(M')$ with $M \neq M'$
  - If $|M| \neq |M'|$ then
    - The last blocks are different: they include the bitlength
    - Thus they form a $Compress$ collision:
      $f(M) = Compress(CV_{n-1}, M_n) = Compress\left(CV_{n'-1}, M'_{n'}\right) = f(M')$
  - Otherwise, if $|M| = |M'|$ then there must exist an $i \in \{1, \dots, n\}$ such that:
    - There is a compression function collision:
      $Compress(CV_{i-1}, M_i) = Compress(CV'_{i-1}, M'_i)$ with $(CV_{i-1}, M_i) \neq (CV'_{i-1}, M'_i)$
    - Because if no such i exists then $M = M'$
  - Hence, if $Compress$ is collision resistant then so is f

Merkle-Damgard Framework

- Types of collision attacks on $Compress$
  - Collision: Given CV find $Compress(CV, M) = Compress(CV, M')$
  - Pseudo-collision: find $Compress(CV, M) = Compress(CV', M')$
  - Free-start pseudo-collision: find $Compress(CV, M) = Compress(CV, M')$
  - Near-collision: Given $CV, CV', \mathcal{D}$ find $Compress(CV, M) - Compress(CV', M') \in \mathcal{D}$

- Weaknesses
  - Length extension:
    - Given $h = f(M)$ and $|M|$ one can compute $f(pad(M)|S)$ for any S without knowing M
    - This implies that certain MAC constructions using f are insecure:
      - $MAC(K, M) = f_K(M)$, here f_K denotes that $CV_0 = K$
      - $MAC(K, M) = f(K \mid M)$
    - Also implies: 1 known collision ⇒ infinitely many known collisions by appending S

Merkle-Damgard Framework

- Weaknesses
  - Joux's Multi-collisions
    - Assume a collision attack with given prefix P
    - I.e., it outputs $f(M) = f(M')$ with $M = P|S$, $M' = P|S'$ with $S \neq S'$
    - Then one can chain collision attacks:
      - $f(S_0) = f(S_0'), f(S_0|S_1) = f(S_0|S_1'), f(S_0|S_1|S_2) = f(S_0|S_1|S_2')$, ….
      - Note that there are now 2^3 colliding messages:

        $$f(S_0|S_1|S_2) = f(S_0'|S_1|S_2) = f(S_0|S_1'|S_2) = f(S_0'|S_1'|S_2) = \cdots$$
    - More general: t chained collision attacks give a 2^t multi-collision

  - Concatenating hash functions is only as secure as the most secure one
    - Let $f(x) = (f_1(x), f_2\,(x))$ with f_1 and f_2 Merkle-Damgard hash functions
    - Wlog let f_1 be the easiest to find collisions for, and let $f_2: \{0,1\}^* \rightarrow \{0,1\}^n$
    - Then generate a 2^n multi-collision for f_1 with n collision attacks
    - Now perform a generic collision attack over this 2^n size message space for f_2
    - The found collision $f_2(x) = f_2(x')$ is by construction also a collision $f_1(x) = f_1(x')$

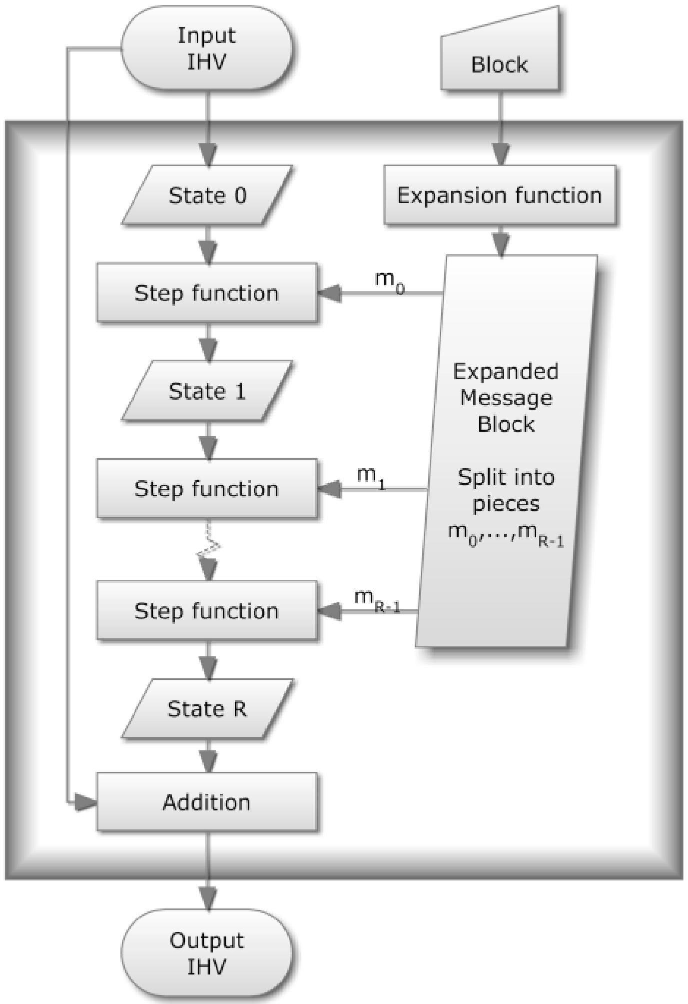
Compression function

- How to construct a secure compression function?
  - Davies-Meyer Feed-Forward:
    - Use a block cipher $E_K(P) = C$
    - Input message block as key, chaining value as plaintext
    - Feed-forward: also add input chaining value to output

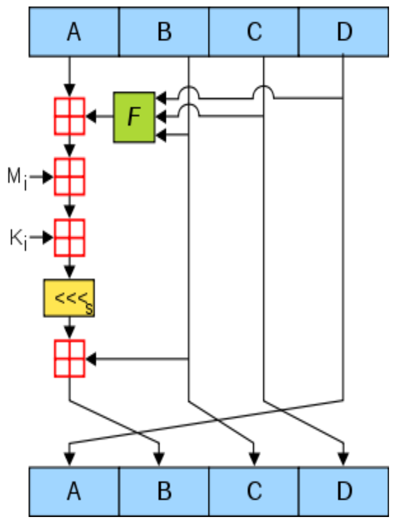
$$Compress(CV, M) = E_M(CV) + CV$$

    - [Winternitz 1984]:
      - if E is an *ideal* block cipher
      - $\Rightarrow f$ is collision resistant

  - MD4 style-compression function:
    - Davies-Meyer Feed-Forward
    - Message is expanded into pieces
    - Step function uses a single message piece
    - Entire message is processed at least 4 times



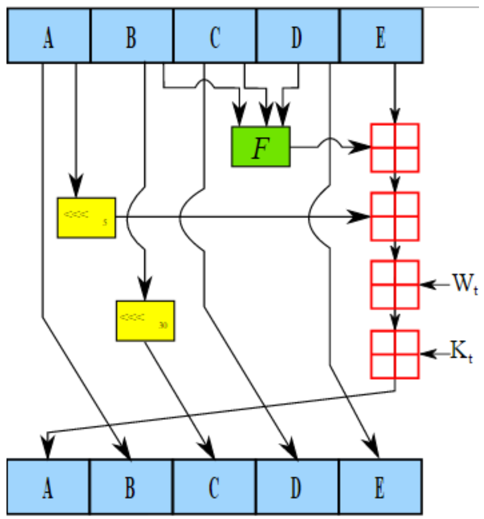
MD5 / SHA-1 / SHA-256 compression function



MD5

very cheap
collision attacks

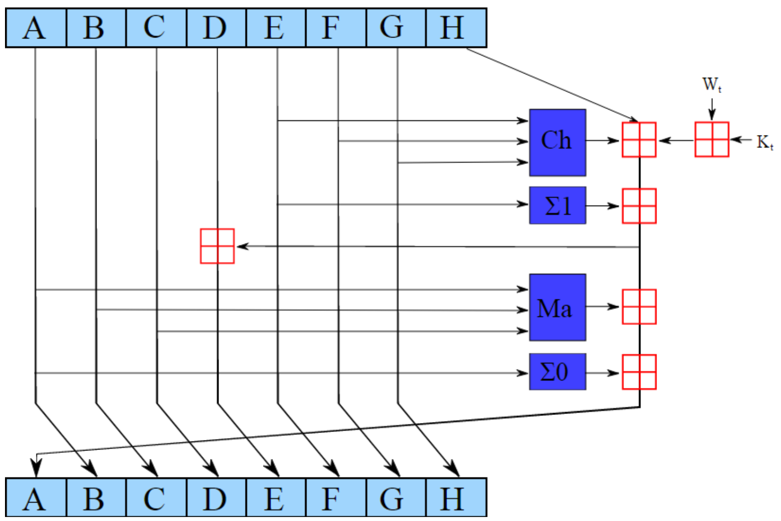
permutation



SHA-1

practically broken

linear recurrence



SHA-2

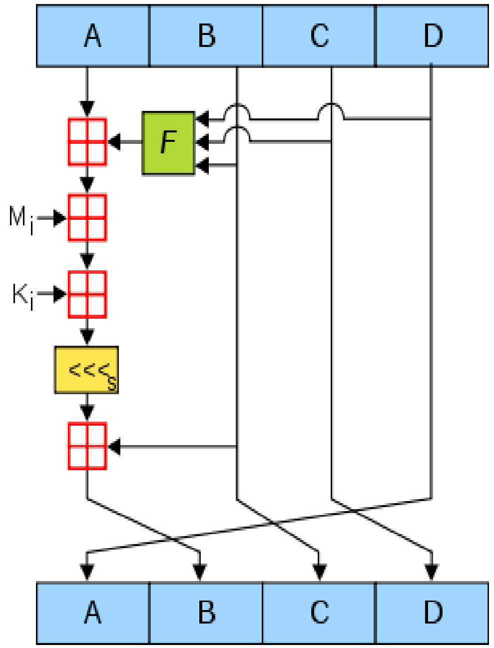
secure

non-linear

message expansion
16 x 32-bit → {64,80} x 32-bit

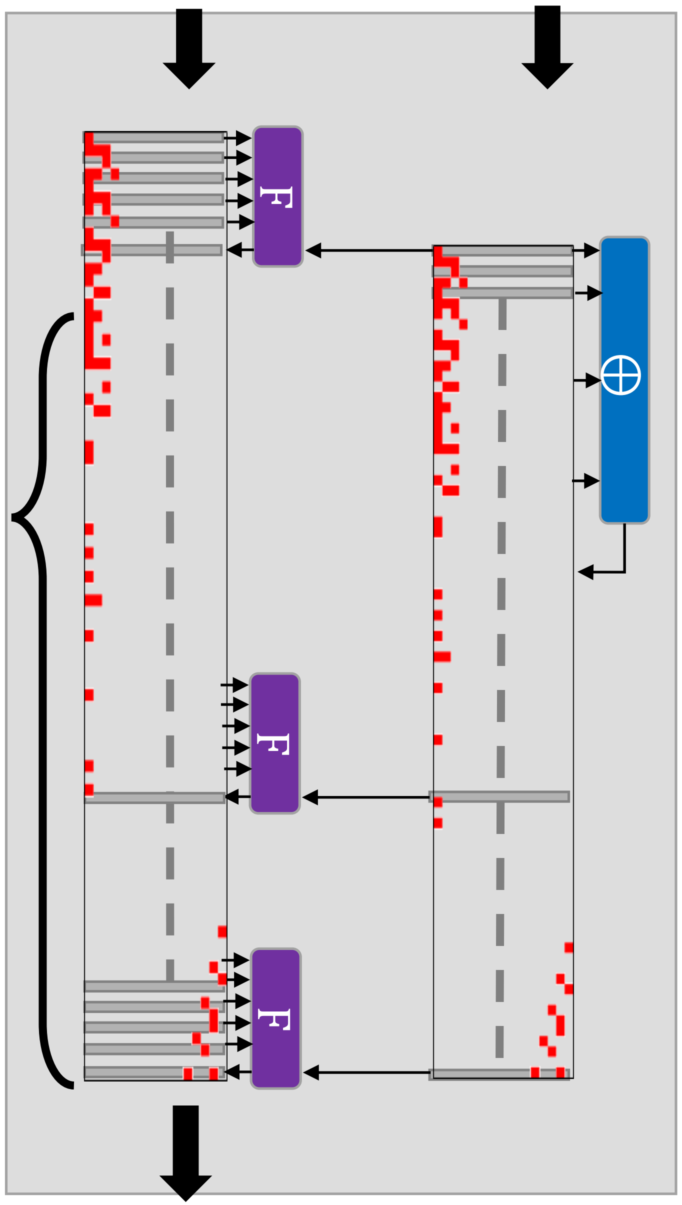
MD5 compression function

- [Rivest, 1991]
- 128-bit hash: 4 x 32-bit words
- 512-bit message block: 16 x 32-bit words
- There are 4 rounds
  - Each round has 16 steps
  - and uses a permutation of the message
- Starting state: $(Q_{-3}, Q_{-2}, Q_{-1}, Q_{-0}) \leftarrow CV$
- There are 64 steps:
  - $F_i = f_i(Q_i, Q_{i-1}, Q_{i-2})$ bitwise function
  - $T_i = F_i + M_{\pi(i)} + Q_{i-3} + AC_i$ addition mod 2^{32}
  - $Q_{i+1} = Q_i + RR(T_i, RC_i)$ bitwise cyclic rotation
  - Constants: AC_i, RC_i
- Output: $(Q_{-3}, Q_{-2}, Q_{-1}, Q_0) + (Q_{61}, Q_{62}, Q_{63}, Q_{64})$
- Each step is a bijection between multiple pairs of variables
  - (when fixing all others)
  - $M_{\pi(i)} \leftrightarrow Q_{i-3}$ compute backward
  - $M_{\pi(i)} \leftrightarrow Q_{i+1}$ compute forward
  - $Q_{i-3} \leftrightarrow Q_{i+1}$



Differential Path

- Differential cryptanalysis
  - Consider two related computations
  - Right column: message expansion
  - Left column: state computation
- Differential path
  - Precise description of how differences propagate through compression function
  - Use signed difference of bits
  - Last ~44 steps determine most of attack's complexity
  - Translate differential path into system of equations to solve



System of equations

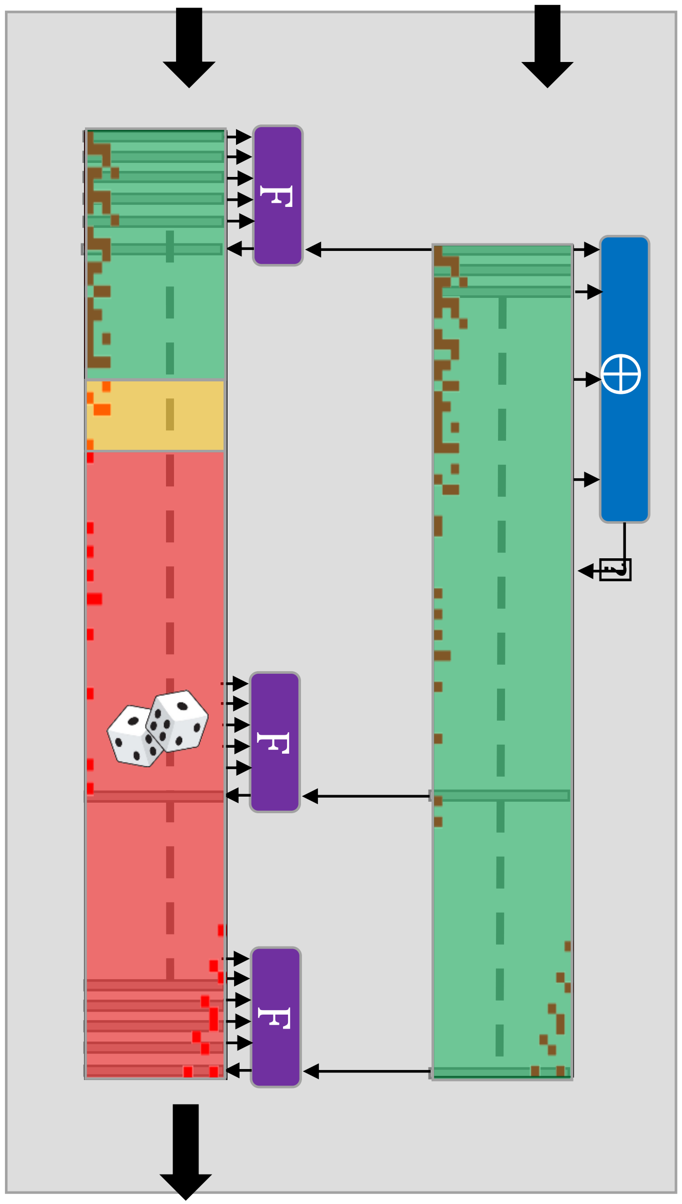
System of equations

- Simple equations on state bits $Q_i[b], Q'_i[b]$
- Chosen Message differences automatically hold

- First 16 steps easily solved: exploit control of message
  ⇒ determines remaining 48 steps

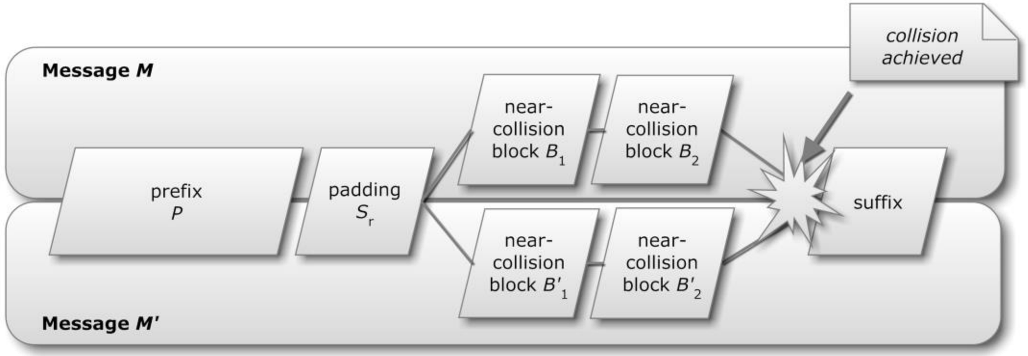
- Make predictable small changes to solve up to step 25
  (amortizes cost of earlier steps)
  ⇒ only control about 39% of MD5

- Find many solutions up to step 25
  to probabilistically fulfill remaining steps



Example Differential Path

- Example differential path
  - First MD5 differential path [Wang et al, 2004]
  - Made by hand !
  - Note: near-collision attack: there is a difference in the end
  - Just need a second near-collision attack to negate it again

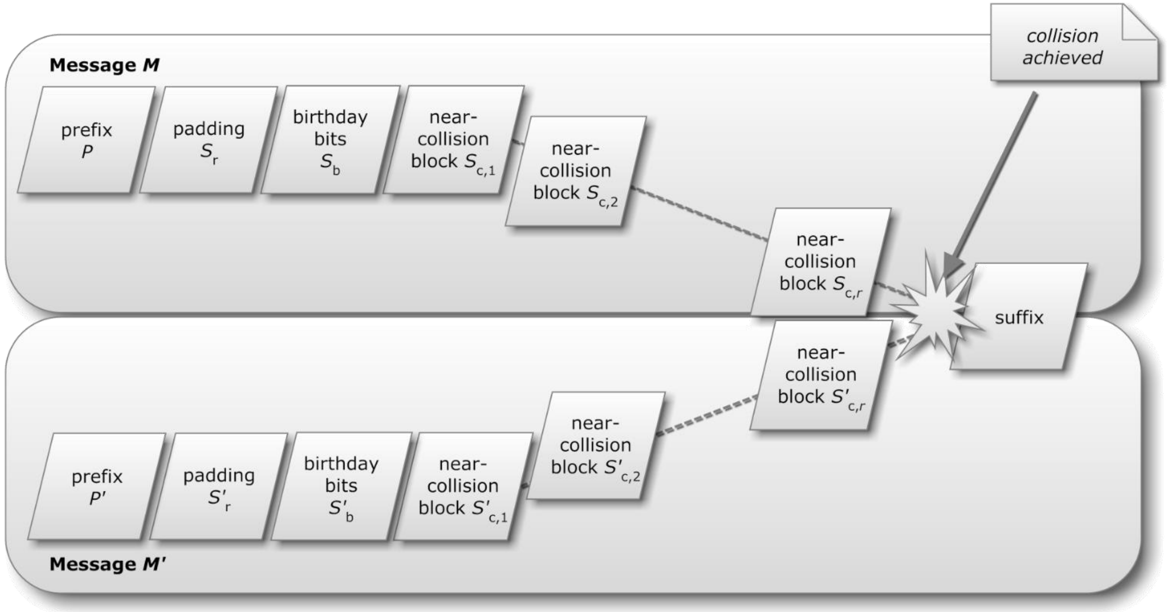


- Nowadays easy to create differential paths
  - Project HashClash [Stevens, 2007]
  - https://github.com/cr-marcstevens/hashclash
  - Build and execute your own collision attack:
    - scripts/poc_no.sh

t	Bits Q_t: $b_{31} \ldots b_0$	#
−3	10001010 11111100 01010110 11011110	32
−2	11000100 10011010 01100010 -0-10110	32
−1	01111101 01010011 01101110 -0-11110	32
0	11101011 00011111 000010-+ +0-11010	32
1	..0..0.. ....11.. 0-0-..01 -1-.....	13
2	.1.!0+.. ....1+.. -0++..00 --+.....	15
3	.1!.01.. .0..+-.. 1-0-.... --+.....	14
4	!-..-1.. .0..+-!. +1++.... .+......	13
5	!-00-.00 ^-001-.0 101+0000 1+000000	30
6	!+11-011 ++11--01 1.+-1111 1.111111	30
7	!1..-... 00.^--.! -.01.... .1^.^...	15
8	!1..+... 10!-.... -.0-.0.. ..+0+..0	15
9	..!.1... ...010.. -..+.0.. .!001^.0	14
10	00.!-010 00.1..10 .00+!+.0 .01+1-1-	25
11	110.-111 1100^011 01110+01 001-000+	31
12	.11^00+1 0010+1+^ 00^1111. 1-0-0+-0	30
13	^1+----0 1-0+0+0- +++++++1 +--+-++0	32
14	--1110-+ +++++0+1 00000010 +--0---.	31
15	1+1+1-1- 011-1+10 0000000- 011-.10.	30
16	01...00+ 10111+1. ..+..1.. 100-^01.	21
17	.0.^.+.1 .1.^.+.. ..-..1.^ .0.0..0.	13
18	.......1 .+...+.. ..1..+.. .1.1..1.	8
19	0....^.+ ....0-.. ..-..... .-....-.	8
20	1...0... .^..1-.. 0....^.. .1.....0	9
21	+...1..^ ...0-0.. 1.^..... .0....^0	11
22	....+... ...1.^.. +....... .1.....+	6
23	^......0 ..0-^... 1....... .+.0....	8
24	....^..1 ..10.... 0.....0. ...1...^	8
25	.......- ..-+.... ....^.-....	5
26	..0..... ...-.... .....0+.	4
27	..1....^ ..^1.... .....1+. ...^....	7
28	..+..... ...0.... .....+-.	4
29	..0..... ....0.	2
30	..-..... ....^1.	3
31	..-.....	1
32	....	0
33	..!.....	1
34 – 60	....	0
61	....	
62	.... +.......	
63	.... +.......	
64	.... +.......	

Chosen-prefix collisions

- Chosen-prefix collision [Stevens et al, 2007]
  - More powerful attack than a simple collision attack
  - Make any 2 files collide by appending data
  - Family of near-collision attacks
    that combined sequentially can eliminate any $\Delta CV \in \mathcal{D}$
  - But first use a birthday search to find $f(M|S) - f(M'|S') \in \mathcal{D}$



Real world attacks

- Chosen-prefix collisions attacks have been demonstrated in the real world
  - [Stevens et al, 2007]
    - Rogue Certificate Authority: can impersonate any website
  - [Flame malware, 2012]
    - Windows Update Certificate
    - They could create malicious windows updates & push to arbitrary windows machines
  - [Peled, Rozenshein, 2023]
    - Confuse Windows CryptoAPI's MD5 based indexing
    - Exploit confusion with invalid certificate to impersonate websites
  - [GHHMSSS, 2024]
    - Real-time chosen-prefix collision attack against RADIUS authentication protocol
    - RADIUS used in network equipment: ISP equipment, routers, WiFi controllers, …
    - Replace reject message with colliding accept message
  - Big collection of collision attacks for various file formats:
    - https://github.com/corkami/collisions [AS]