



Cryptanalysis of MD5 & SHA-1

Marc Stevens

CWI, Amsterdam

Overview



- Part I: introduction
 - Merkle-Damgard and compression functions
 - Cryptanalytic history of MD5 & SHA-1
- Part II: collision search algorithm
 - Differential paths & sufficient bitconditions
 - Collision search algorithm
 - Massively-parallel architectures
- Part III: new cryptanalysis SHA-1
 - Local collisions & disturbance vectors
 - New exact joint local collision analysis
 - Deriving sufficient conditions
 - New attacks
 - HashClash: open-source project



Part I

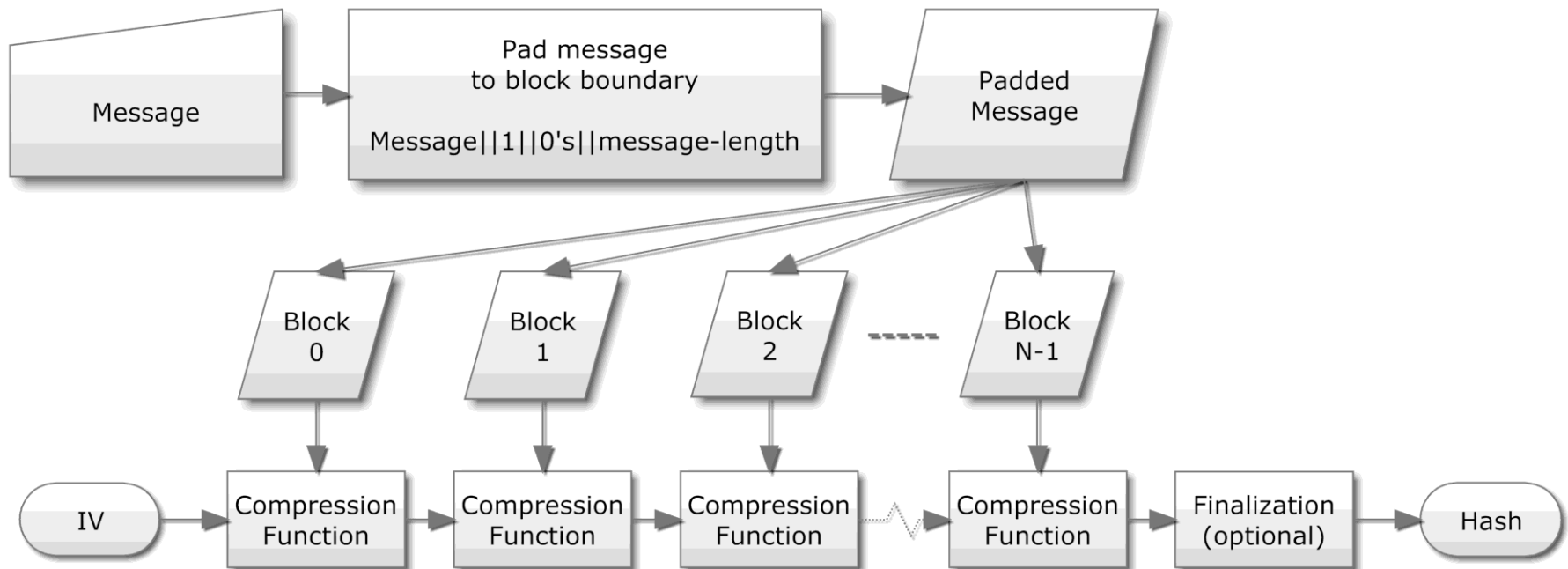
introduction

- Merkle-Damgard and compression functions
- Cryptanalytic history of MD5 & SHA-1

Merkle-Damgard



- Message M split into pieces M_0, \dots, M_{N-1}
- Iteratively processed w/ compression function
- Internal state: I_{HV} (initialized with IV)



Compression function attacks



- Collision attack
 - Given IHV : compute $M \neq M'$ s/t
$$CF(IHV, M) = CF(IHV, M')$$
- Near-collision attack
 - Given IHV, IHV', D : compute $M \neq M'$ s/t
$$CF(IHV', M') - CF(IHV, M) \in D$$
- Pseudo-collision attack
 - Compute $(IHV, M) \neq (IHV', M')$ s/t
$$CF(IHV, M) = CF(IHV', M')$$
 - Called “free-start” if $IHV = IHV'$

Short history of MD5 attacks



- 1992 MD5 published [Riv92]
- 1993 pseudo-collision attack [dBB93]
- 1995 free-start pseudo-collision attack [Dob95]
- 2004 *identical-prefix* collision found: 2^{40} calls [WY04]
- 2006 *chosen-prefix* collision: 2^{49} calls [SLdW07]
- 2009 identical-prefix: 2^{16} calls [SSA+09]
 - chosen-prefix: 2^{39} calls [SSA+09]
 - realistic abuse scenario: rogue CA [SSA+09]

Short history of MD5 attacks



Shortest collision attacks

2009 short chosen-prefix collision: $2^{53.2}$ calls [SSA+09]

- birthday-search + 1 near-collision
- # collision bits: 80+512 bits
- # prefix bits = 432 + 512 · N bits

2010 compression function collision found [XF10]

- 512-bit collision
- no details published
- \$10,000 challenge

2012 challenge broken: $2^{49.8}$ calls [S12]

Short history of SHA-1 attacks



1995 SHA-1 published [NIST95]

2005 first SHA-1 collision attack: 2^{69} calls [WYY05]

– two near-collision attacks: $2 \cdot 2^{68}$ calls

2005 claim: 2^{63} calls [WYY05]

2007 claim: 2^{61} calls [MRR07]

2009 paper: 2^{52} calls [MHP09]

2011 [RFC6194]: first attack is best attack

2012 New results in [[thesis](#)]

- Exact joint local-collision analysis
- Preliminary near-collision attack: $2^{57.5}$ calls
- Extends to identical- & chosen-prefix collision



Part II

collision search algorithm

- Differential paths & sufficient bitconditions
- Collision search algorithm
- Massively-parallel architectures

Preliminaries – MD5



- Compression function: $(IHV_{\text{in}}, B) \rightarrow IHV_{\text{out}}$
- Uses 32-bit words $\{0, 1\}^{32} \leftrightarrow \mathbb{Z}_{2^{32}}$
- Initialization
 - B expanded into 64 words: W_0, \dots, W_{63}
 - Working state: 4 words $(Q_{t-3}, Q_{t-2}, Q_{t-1}, Q_t)$
for $t=0$ set to IHV_{in}
- Step function:
$$F_t = f_t(Q_t, Q_{t-1}, Q_{t-2});$$
$$Q_{t+1} = Q_t + (F_t + Q_{t-3} + W_t + AC_t) \lll RC_t. \quad t = 0, \dots, 63$$
- Finalization:
$$IHV_{\text{out}} = IHV_{\text{in}} + \Pi(Q_{61}, Q_{62}, Q_{63}, Q_{64})$$

Preliminaries – SHA-1



- Compression function: $(IHV_{\text{in}}, B) \rightarrow IHV_{\text{out}}$
- Uses 32-bit words $\{0, 1\}^{32} \leftrightarrow \mathbb{Z}_{2^{32}}$
- Initialization
 - B expanded into 80 words: W_0, \dots, W_{79}
 - Working state: 5 words $(Q_{t-4}, Q_{t-3}, Q_{t-2}, Q_{t-1}, Q_t)$
for $t=0$ set to IHV_{in}
- Step function:
$$F_t = f_t(Q_{t-1}, Q_{t-2}^{\lll 30}, Q_{t-3}^{\lll 30});$$
$$Q_{t+1} = Q_t^{\lll 5} + F_t + Q_{t-4}^{\lll 30} + W_t + AC_t.$$
$$t = 0, \dots, 79$$
- Finalization:
$$IHV_{\text{out}} = IHV_{\text{in}} + \Pi(Q_{76}, Q_{77}, Q_{78}, Q_{79}, Q_{80})$$

Differential analysis



- Analyze two instances of computation
 - First instance: variables X
 - Second instance: variables X'
 - Modular difference: $\delta X = X' - X$
 - Bitwise difference: $\Delta X = (X'[b] - X[b])_{b=0}^{31} \in \{-1, 0, 1\}^{32}$
 - Bitwise to modular: $\delta X = \sum_{b=0}^{31} 2^b \cdot \Delta X[b]$
- Differential path
 - Precise differences for all variables
$$\Delta Q_i, \quad \Delta F_t, \quad \delta W_t$$
 - Satisfying step function
 - MD5 $\delta Q_{t+1} = \delta Q_t + (\delta F_t + \delta Q_{t-3} + \delta W_t) \lll RC_t$
 - SHA-1 $\delta Q_{t+1} = \delta(Q_t \lll^5) + \delta F_t + \delta(Q_{t-4} \lll^{30}) + \delta W_t$

Sufficient conditions



- Derive bitconditions from differential path
 - Conditions on first instance variables W_t, Q_i
s/t differential path holds using given $\delta W_t, \delta IHV_{in}$
- Benefits collision finding algorithm
 - Only needs to consider one instance (mostly)
 - Bitconditions are easily tested

Sufficient conditions



Sufficient bitconditions

- Working state bitconditions $Q_t[b] = \dots$
 - Free
 - Constant: 0,1
 - Previous bits
 - E.g. $Q_{t-1}[b], \overline{Q_{t-1}[b]}$
 $Q_{t-1}[b+2], \overline{Q_{t-1}[b+2]}$
 $Q_{t-2}[b+2], \overline{Q_{t-2}[b+2]}$

Sufficient conditions



Sufficient bitconditions

- Message bitconditions

- MD5

- Message expansion permutation
 - Desired δW_t are immediate

- SHA-1

- Bitwise linear message expansion

$$W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) \lll 1$$

- Need linear bitrelations to achieve desired δW_t

$$W_t[b] = c + \sum_{i=0}^t \sum_{j=0}^{32} c_{ij} \cdot W_i[j] \bmod 2$$

- All linear bitrelations can be satisfied in first 16 steps

Collision finding algorithm



- Basic depth-first search
 - Start at step 0
 - At step t find W_t, Q_{t+1} satisfying conditions
 - For each valid pair: continue with step $t+1$
 - After first 16 steps message fully determined
 - Verify remaining Q_i conditions
- Apply speedup: tunnel/boomerang/neutral-bit/...
 - At step $k \geq 16$: conditions on steps $0, \dots, k-1$ hold
 - Apply small changes in first 16 steps
s/t conditions on steps $0, \dots, k-1$ still hold
 - (Partially) recompute steps $16, \dots, k$
 - Verify bitconditions on Q_{k+1}

Massively-parallel architectures



- Collision search freely parallelizable
 - Splitting entire search space
- Massively-parallel architectures
 - Higher performance/cost-ratio
- Target architecture: NVIDIA GPUs
 - 32 threads of computation grouped in 1 warp
 - Many active warps on GPU
 - Same instruction path per warp: requires coherency
 - Very suitable for birthday search
 - Complete compression functions computations
 - Less suitable for collision search
 - Split into individual small steps
 - many loops and branches

Massively-parallel architectures



Ideas for collision search on GPU

- First 16 steps
 - Per instance:
 - Buffers of (W_t, Q_{t+1}) -pairs for each step + pointer
 - Exhaustively go through freedoms for one step
 - Store valid (W_t, Q_{t+1}) in buffer
 - Move pointer through buffer while processing next step
 - Option 1: process many instances in 1 warp
 - Many uncoalesced reads and writes
 - Option 2: process 1 instance in 16 threads
 - Coalesced reads and writes
 - Need to orchestrate writing in shared list
 - Smaller memory footprint (less active instances)

Massively-parallel architectures



- Remaining steps

- Basic idea: split into tasks: blocks at same step
 - Warp: read very similar tasks for same step
 - Process tunnel & verify conditions
 - Write successes as new tasks for succeeding step
- Option 1: process 1 task in 16 threads
 - Coalesced reads
 - Divide k-bit tunnel over 16 threads, $k \geq 4$
- Option 2: process many tasks in 1 warp
 - Combine very similar tasks together to get large coalesced/uncoalesced-read ratio
 - Loop k-bit tunnel
 - Possible free-start next step
 - Combine these two steps within 1 task
 - If on average 1 or more successes per thread

Massively-parallel architectures



- Further considerations
 - Optimal: groups of 16 very similar tasks
 - Maximize coalesced reads & writes
 - What if: groups of 15 very similar tasks + 1 task
 - Reads and writes uncoalesced
 - Extra overhead: up to 2x slower reads & writes
 - Skip +1 task: only 1/16 loss
 - Threshold? $15+1$ / $14+2$ / $13+3$?
 - What if: single task without very similar siblings
 - Expensive on GPU (as per above case)
 - Handle by CPU
 - Avoid loss of tasks

Massively-parallel architectures



- Further considerations
 - Goal is to maximize performance/cost ratio
 - At least above p/c ratio for CPU
 - Significantly slower than raw compression function
 - Need many loops & tests
 - Overhead due to tasks
 - Additional reads & writes
 - Less time spent in actual step computations
 - Expect to gain at least a small factor
 - Very happy to be $\sim 20x$ faster than CPU core



Part III

new cryptanalysis SHA-1

- Local collisions & disturbance vectors
- New exact joint local collision analysis
- Deriving sufficient bitconditions & bitrelations
- New attacks
- HashClash: open-source project

Deriving sufficient conditions



Deriving sufficient conditions for collision search

- First 20 steps
 - Differential path construction
 - [dCR06] Coding theory principles
 - [YSN+07][[thesis](#)] Forward, backward & join in the middle
 - Message bitrelations (uni-variable)
 - Working state bitconditions
- Last 60 steps
 - Disturbance vector analysis
 - Combine local collisions

Local collisions




- Local collision

- single disturbance: $\delta W_t = 2^b$
- 5 corrections: $\delta W_{t+1}, \dots, \delta W_{t+5}$
- Any step, any bit

- Variations

- signs
- carries

i	ΔQ_i	ΔF_i	δW_i
$t-4$	0	0	$+2^b$
$t-3$	0		
$t-2$	0		
$t-1$	0		
t	0		
$t+1$	$+2^b$		
$t+2$			
$t+3$			
$t+4$			
$t+5$			
$t+6$			

$$f_i(\underline{Q_{i-1}}, Q_{i-2}^{\lll 30}, Q_{i-3}^{\lll 30})$$

$$\underline{\delta Q_{i+1}} = \sigma((\Delta Q_i)^{\lll 5}) + \sigma(\Delta F_i) + \delta(Q_{i-4}^{\lll 30}) + \underline{\delta W_i}$$

Disturbance vector

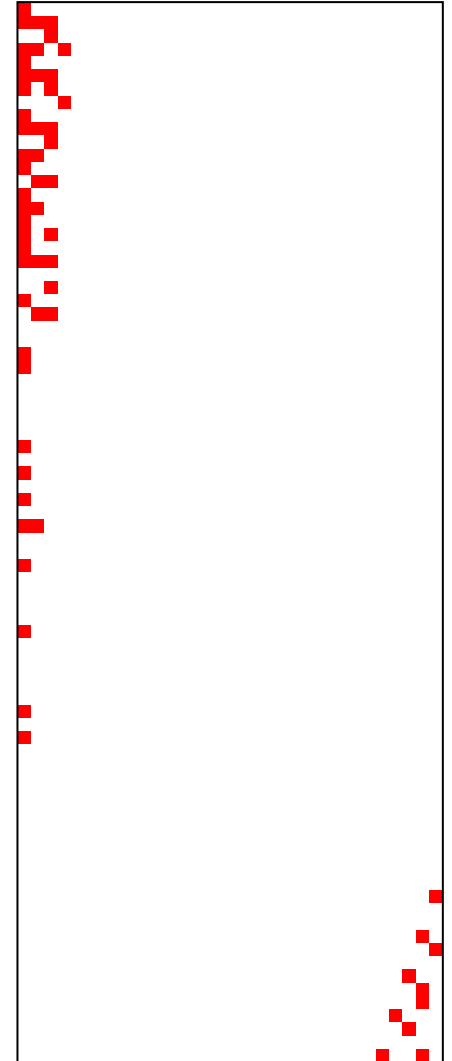


- Linear message expansion

$$W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) \lll 1$$

- Combine local collisions

- Disturbance vector
- Vector $(W_t \oplus W'_t)_{t=0}^{79}$
 - Linear combination of D.V.
 - Forward-shifted & rotated
 - Also satisfies msg.exp.
- XOR difference
 - Need linear message bitrelations to obtain desired δW_t
 - More precise: set of desired δW_t
 - Same success probability
 - More freedoms



Disturbance vector



- Disturbance vector analysis
 - Estimating collision attack complexity
 - Various cost functions
 - Hamming weight: # local collisions
 - Sum of # bitconditions per local collision
 - Product of max. success probability per local collision
 - All assume independence of local collisions
 - Inaccurate [Man11][[thesis](#)]
 - Affects choice for “optimal” disturbance vector
 - May lead to sub-optimal complexity
 - May even lead to discrepancies between theoretical and actual attack complexity

D.V.-allowed differential paths



- Differential path \mathcal{P} over steps $20, \dots, 79$
 - message differences (precondition)
$$w = (\mathcal{P}) = (\delta W_t)_{t=20}^{79}$$
 - differences at step 20 (precondition)
$$\Lambda = \Phi(\mathcal{P}) = (\delta(Q_{16}^{\lll 30}), \Delta Q_{17}, \Delta Q_{18}, \Delta Q_{19}, \Delta Q_{20})$$
 - ending differences (postcondition)
$$\delta IHV_{\text{diff}} = \Psi(\mathcal{P}) = (\delta Q_{80}, \delta Q_{79}, \delta(Q_{78}^{\lll 30}), \delta(Q_{77}^{\lll 30}), \delta(Q_{76}^{\lll 30}))$$
- Set $\mathcal{D}_{[20,79]}$ of allowed differential paths
 - Matching D.V. disturbances (up to carries)
 - With message differences possible under given $W_t \oplus W'_t$
 - Non-zero probability
 - Theoretical set: never directly computed

D.V. - maximum success probability



- Success probabilities
 - Group diff. paths by pre-/post-conditions
 - Sum of probabilities of diff. paths within group

$$p_{w,\Lambda,\delta IHV_{\text{diff}}} = \sum_{\substack{\mathcal{P} \in \mathcal{D}_{[20,79]} \\ \Lambda = \Phi(\mathcal{P}) \\ w = (\mathcal{P}) \\ \delta IHV_{\text{diff}} = \Psi(\mathcal{P})}} \Pr[\mathcal{P}]$$

- Deterministic algorithm
- Maximum success probability

$$p_{\text{max}} = \max_{\substack{w \\ \Lambda \\ \delta IHV_{\text{diff}}}} p_{w,\Lambda,\delta IHV_{\text{diff}}}$$

Deriving optimal sufficient conditions



- Differences at step 20
 - Select set \mathcal{I} of Λ -values achieving p_{\max}
 - Use \mathcal{I} to construct differential path over first 20 steps
 - Let $\tilde{\Lambda}$ match the found differential path
- First near-collision
 - No restriction to specific δIHV_{diff} -value
 - Speedup by allowing many values
 - Look at all pairs $(w, \delta IHV_{\text{diff}})$ leading to p_{\max}
 - Keep only w with N_{\max} pairs: speedup by N_{\max}
- Second near-collision
 - Restriction to specific δIHV_{diff} -value: no similar speedup
 - Keep only w that lead to p_{\max}
- Determine message bitrelations from set of w

New D.V. cost function



- New disturbance vector cost function

$$FDC((DV_t)_{t=0}^{79}) = \max_{\substack{w \\ \Lambda \\ \delta IHV_{\text{diff}}}} p_{w, \Lambda, \delta IHV_{\text{diff}}} \cdot 2^{\overbrace{w(\Delta Q_{17}) + w(\Delta Q_{18})}^{\uparrow \quad \uparrow}}$$

- correction due to fulfillment of ΔQ_{17} and ΔQ_{18} before fulfillment of ΔF_{20} in attack implementation

- Comparison cost function

$$FIC((DV_t)_{t=0}^{79}) = \prod_{Y \in \Gamma((DV_t)_{t=0}^{79})} FDC(Y)$$

where Γ breaks D.V. into separate D.V.s

- Each containing 1 local collision
- Using local collision compression

Comparing effect of dependent L.C.s



- Comparison for selected disturbance vectors

<i>DV</i>	FDC	FIC	diff
I(48, 0)	71.4	80.5	9.1
I(49, 0)	72.2	79.6	7.4
I(50, 0)	71.9	81.4	9.5
I(51, 0)	73.3	85.8	12.5
I(48, 2)	73.8	75.7	1.9
I(49, 2)	73.8	74.1	0.3
II(50, 0)	73.0	77.4	4.4
II(51, 0)	71.9	77.7	5.8
II(52, 0)	71.8	79.4	7.6

- Results: $-\log_2$
- Selection by (near-)optimal FDC
- Note: maximum success probability only obtained using the optimal message differences

Computing success probabilities



Computing $p_{w,\tilde{\Lambda},\delta IHV_{\text{diff}}}$

- Set $\mathcal{D}_{[20,79]}$ too big to compute directly
- Observation:
 - Effect of disturbance is local
 - Many differential paths equivalent under change of signs
- Idea:
 - Differential path reduction
 - Remove differences 'independent' from pre-/post- conditions
 - Set $\mathcal{R}_{[20,79]}$ of all reduced paths from $\mathcal{D}_{[20,79]}$
 - Iteratively computable
 - Success probabilities $p_{w,P}$ over w and $P \in \mathcal{R}_{[20,79]}$
 - Iteratively computable
 - Together used to determine $p_{w,\tilde{\Lambda},\delta IHV_{\text{diff}}}$

Near-collision attack construction



- Preliminary first near-collision attack
 - 192 possible δIHV_{diff}
 - 6 possible δIHV_{diff} -values per w : speedup factor 6
 - runtime complexity of about $2^{57.5}$ calls
 - Publicly verifiable
 - improves upon 2^{68} by [WYY05]
- Second near-collision attack
 - at least 6 times slower: $2^{60.1}$ calls
 - also more restrictions: slightly more slower

Collision attack construction



- Identical-prefix collision attack
 - First + second near-collision attack
 - Complexity
 - Estimated complexity: approx. 2^{61} calls
 - Improves upon 2^{69} calls
- Chosen-prefix collision attack
 - Birthday-search + second near-collision attack
 - Complexity
 - Birthday-search: average $2^{77.06}$ calls
 - Near-collision attack complexity negligible
 - Average complexity: approx. $2^{77.1}$ calls
 - First chosen-prefix collision attack on SHA-1

Project HashClash



- HashClash @ Google Code
 - <http://code.google.com/p/hashclash>
 - Published sources and binaries
 - MD5
 - Differential path construction
 - Collision finding
 - Birthday-search for chosen-prefix collisions (supporting CPU, CUDA and CELL)
 - Chosen-prefix collision GUI
 - SHA-1
 - Differential path construction
 - Near-collision attack
 - Soon: disturbance vector analysis



Thank you for your attention

Questions?

More information



- Contact: marc@marc-stevens.nl
- Website: <http://marc-stevens.nl/research>
- HashClash: <http://code.google.com/p/hashclash>
- Information on MD5 attack applications:
<http://www.win.tue.nl/hashclash>