Improving Counter-cryptanalysis

Marc Stevens
marc.stevens@cwi.nl

CWI Amsterdam
Part I – Weak signature schemes

Part II – Counter-cryptanalysis
Part III – Flame
Part IV – Improvements
To Conclude…
Digital signature schemes

- One of the pillars for P.K.I.s

- Used to ensure authenticity in/of
  - Browsers
  - Documents
  - Email
  - Software updates
  - Downloadable content
  - Currency transactions

- Hash-Then-Sign:
  \{MD5,SHA-1,SHA-2\}-\{RSA,DSA\}

- Hash collision $\text{MD5}(A) = \text{MD5}(B) \Rightarrow$ forgery
Collision attacks on MD5 & SHA-1

- Distinguish between 2 types
  - Identical prefix
    \[ H(P \mid C \mid S) = H(P \mid C' \mid S) \]
    \[ H(\quad ) = H(\quad ) \]
  
  - Chosen-prefix
    \[ H(P \mid C \mid S) = H(P' \mid C' \mid S) \]
    \[ H(\quad ) = H(\quad ) \]

- \( P, P', S \): Free to choose \( s/t \mid P \mid = \mid P' \mid \)
- \( C, C' \): Generated based on \( P \) and \( P' \), \( \mid C \mid = \mid C' \mid \in [64B,1KB] \)
## Introduction

**Collision attacks**

<table>
<thead>
<tr>
<th>Year</th>
<th>MD5</th>
<th>SHA-1</th>
<th>SHA-256</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Id.Pr.</td>
<td>Ch.Pr.</td>
<td>Id.Pr.</td>
</tr>
<tr>
<td>Birthday</td>
<td>$2^{64.3}$</td>
<td>$2^{64.8}$</td>
<td>$2^{80.3}$</td>
</tr>
<tr>
<td>2004</td>
<td>$2^{40}$</td>
<td></td>
<td>$2^{69}$</td>
</tr>
<tr>
<td>2005</td>
<td>$2^{37}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>$2^{32}$</td>
<td>$2^{49}$</td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td>$2^{25}$</td>
<td>$2^{42}$</td>
<td></td>
</tr>
<tr>
<td>2008</td>
<td>$2^{21}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>$2^{16}$</td>
<td>$2^{39}$</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td></td>
<td></td>
<td>$2^{61}$</td>
</tr>
<tr>
<td>today</td>
<td>$2^{16}$</td>
<td>$2^{39}$</td>
<td>$2^{61}$</td>
</tr>
</tbody>
</table>

**Published collision attacks on MD5 & SHA-1**
Notes

- Generate your own MD5 chosen-prefix collision attack in a day using Project HashClash: https://code.google.com/p/hashclash/

- No publicly known collision for SHA-1 has been found yet

- First SHA-1 collision more likely to be constructed by nation-states than academia due to required resources, see: http://www.schneier.com/blog/archives/2012/10/when_will_we_se.html
Strategies for meaningful colliding files

- Using identical-prefix collisions
  - Meaningful $C$ and $C'$
    
    $C = "...$ of money is $10,000.00..."
    $C' = "...$ of money is $20,000.00 ..."

    $C = "... OFFSET=X ..."
    $C' = "... OFFSET=Y ..."
  
  - IF-THEN-ELSE construct
    
    IF ( $C == C$ ) THEN ... ELSE ...
    IF ( $C' == C$ ) THEN ... ELSE ...

- Using chosen-prefix collisions
  - Meaningful different $P$, $P'$ & hide $C$ and $C'$ in message
    
    $P = "I owe you $20"
    P' = "You will inherit all my possessions"
    C = <hidden image>
    C' = <hidden image>
Introduction
Abuse examples

- Identical-prefix
  - Colliding Software [Kam04,Mik04]
  - Colliding PostScript documents [DL05,GIS05]
  - Colliding X.509 certificates (same ID, diff. RSA moduli) [LdW05]

- Chosen-prefix
  - Colliding PDF documents [SLdW07]
  - Colliding Software [SLdW07]
  - Colliding X.509 certificates (diff. IDs) [SLdW07]
  - Rogue Certification Authority [SSALMOdW09]
  - Rogue Windows Update signing certificate [Flame12]
What to do when a signature scheme is broken?

The easy answer: “migrate to a more secure scheme”
i.e., move from MD5-RSA to SHA-2-RSA

Who should migrate?

Signers: generate SHA-2-RSA signatures
Problems: compatibility/deployment issues, risk-cost trade-off, human,…
Result: forgeries can still be constructed till the last signer migrates

Verifiers: don’t accept MD5-RSA signatures
Problem: too many old signatures in use to just invalidate them all at once
Result: old and new forgeries can abused against nearly everyone

The easy answer is not a practical solution for the near future
Our answer: detect forged signatures

Verifiers: don’t accept *forged* MD5-RSA signatures
Results:
- Old legitimate signatures are still valid
- Verifier protected against forgeries
- Independent of migration by signers

How to detect forged signatures?: counter-cryptanalysis!
Part II – Counter-cryptanalysis

Part III – Flame
Part IV – Improvements
To Conclude…
New paradigm: counter-cryptanalysis
- Strengthen weak cryptographic primitives
- Detect cryptanalytic attacks at the cryptographic level
- Counter-cryptanalysis in principle enables the continued secure use of weak cryptographic primitives
- No strengthened redesign ⇒ no compatibility issues
- May be used during migration to strengthened redesigns in the real world

Why is that possible?
- Dedicated cryptanalytic attacks are highly specialized
- Active attacks may introduce subtle unavoidable anomalies
- Similar cryptanalytic techniques can be used to detect those anomalies
- This approach may detect an entire class of attacks that all introduce the same unavoidable anomalies
Counter-cryptanalysis
Collision detection

First practical example: collision detection
- Detect whether message was constructed using collision attack
- Single message of collision pair sufficient
- Application to MD5 & SHA-1
- Computational cost
  - MD5 factor x 224
  - SHA-1 factor x 15
  - Much less using early-abort: WIP

- Based on crucial properties of the known cryptanalysis on MD5 & SHA-1
  - Attacks exploit trivial differential steps with probability (close to) 1
to be able to obtain ‘low’ complexity
  - Very few message block differences result in attacks with ‘low’ complexity
Basic algorithm: detect last near-collision block
1. Guess message block difference & difference at trivial step $i$
2. Determine $M_k'$ from $M_k$ and $WS_i'$ from $WS_i$
3. Reconstruct computation
4. Check whether collision in chaining value is obtained

If guess was correct then collision is detected with certainty
If guess was incorrect then a false positive occurs with probability $\approx 2^{-N}$
Reference implementation to detect collision attacks

- Available at [http://marc-stevens.nl/research](http://marc-stevens.nl/research) (at the bottom)

- Library interface to replace existing MD5/SHA-1 implementation
  - $MD5Init/MD5Init\_unsafe$, $MD5Update$, $MD5Final$
  - $SHA1Init/SHA1Init\_unsafe$, $SHA1Update$, $SHA1Final$
  - $\{MD5,SHA1\}Final$ returns non-zero value if an attack is detected
  - $\{MD5,SHA1\}Init\_unsafe$ always results in correct (and possibly unsafe) hash
  - $\{MD5,SHA1\}Init$ results in correct hash if no attack has been detected, otherwise a safe hash is returned

- Command line program
  - `detectcollv <files>`
Anomaly detection for digital signatures

- Online: active protection
  - Signer: protection against malicious signature requests
  - Verifier: protection against forged signatures
  - E.g., for TLS/SSL, OSs (drivers, executables, updates), etc.

- Offline: forensic analysis
  - Main example: spyware Flame
Part III – Flame

Part IV – Improvements
To Conclude…
Cf. [Kas12, Sot12]

- Highly advanced malware
- Targeting the Middle-East
- Discovered in May 2012
- Active since 2007 or earlier
- Uncharacteristic features for malware
  - Up to 20 modules: each carefully selected prior to infection
  - Almost 20MB: includes Lua VM & libraries for compression, database, ...
  - Did not spread wildly & evaded discovery for ~5 years
  - Surgical-precision attacks: each target carefully selected
  - Spread itself illegitimately using the Windows Update platform
  - First cryptanalytic attack on hash function found in the ‘wild’
  - Developed new variant cryptanalytic attack to do so…
• Man-in-the-middle attack on Windows Update
• Local network attack
  – Registers itself as proxy server for update.microsoft.com using WPAD (Web Proxy Auto-Discovery)
  – Windows Update falls back to insecure HTTP
    • depends on digital signatures for security
    • no need to subvert TLS/SSL connection

• Propagation
  – Flame serves fake ‘security update’ using Windows Update platform
  – Requires properly-signed ‘security update’
  – Uses illegitimate sub-CA valid since Feb 2010
    ⇒ sub-CA invalid before that time
    ⇒ this attack was almost certainly done around Feb 2010 or later
Flame Certificate hierarchy

Microsoft Root Certificate Authority

- Microsoft Windows Verification PCA
- Microsoft Windows
- Patch_KBxxx.exe

Microsoft Enforced Licensing Intermediate PCA

- Microsoft Enforced Licensing Registration Authority CA
- Microsoft LSRA PA

MS

Terminal Services LS

WuSetupV.exe

MD5 collision attack to forge signature
Uses chosen-prefix collision attack [SLdW07]:

Flame’s certificate:
- Serial number, validity
- CN=MS
- 2048-bit RSA key (271 bytes)
- birthday bits
- issuerUniqueID data
- 4 near collisions blocks (computed)
- identical bytes (copied from issued cert)
- MD5 signature

Standard TLS certificate:
- Serial number, validity
- CN=Terminal Services LS
- chosen-prefix collision (difference)
- RSA key (509 bytes?)
- X509 extensions
- MD5 signature

publicly available
lost!?
Flame
Analyzing Flame’s attack

- Only Flame’s “MS” sub-CA certificate public
- The colliding “TSLS” certificate is not public (lost?)

- First example for counter-cryptanalysis
  - Assumed chosen-prefix collision attack
  - Only 1 of the 2 colliding certificates available to us
  - Ran proof-of-concept implementation (from 2008)
    - chosen-prefix collision detected
    - 4 near-collision blocks recovered
    - all differential paths reconstructed
    - <0.03 seconds
  - Differential paths expose use of new variant attack
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Q3:</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>!31!</td>
</tr>
<tr>
<td>Q2:</td>
<td>00..</td>
<td>.1.1.01.</td>
<td>...+</td>
<td>...+</td>
<td>...+</td>
<td>...+</td>
<td>...+</td>
<td>...+</td>
<td>!15!</td>
</tr>
<tr>
<td>Q1:</td>
<td>.10+</td>
<td>.1. .-0.0</td>
<td>+.+=</td>
<td>...-</td>
<td>...-</td>
<td>...-</td>
<td>...-</td>
<td>...-</td>
<td>!31!</td>
</tr>
<tr>
<td>Q0:</td>
<td>!+100.0</td>
<td>.0-0+</td>
<td>+1.0</td>
<td>.0-11.</td>
<td>.110+</td>
<td>...+</td>
<td>...+</td>
<td>...+</td>
<td>0.10</td>
</tr>
<tr>
<td>Q1:</td>
<td>!+0-0..</td>
<td>.0++0-</td>
<td>.0-1+</td>
<td>.1-0+</td>
<td>...+</td>
<td>...+</td>
<td>...+</td>
<td>...+</td>
<td>!31!</td>
</tr>
<tr>
<td>Q2:</td>
<td>!+00-00-</td>
<td>.0++0+</td>
<td>.0-++</td>
<td>.0--0-</td>
<td>...+</td>
<td>...+</td>
<td>...+</td>
<td>...+</td>
<td>!15!</td>
</tr>
<tr>
<td>Q3:</td>
<td>!+010-000</td>
<td>.0++0+</td>
<td>.0-++</td>
<td>.0--0</td>
<td>...+</td>
<td>...+</td>
<td>...+</td>
<td>...+</td>
<td>0.10</td>
</tr>
<tr>
<td>Q4:</td>
<td>!-00-10+</td>
<td>.12-+</td>
<td>+11-0</td>
<td>.0-12-0</td>
<td>...+</td>
<td>...+</td>
<td>...+</td>
<td>...+</td>
<td>!15!</td>
</tr>
<tr>
<td>Q5:</td>
<td>!0-+++-</td>
<td>.01101</td>
<td>.110+0</td>
<td>.0-001+!</td>
<td>...+</td>
<td>...+</td>
<td>...+</td>
<td>...+</td>
<td>!31!</td>
</tr>
<tr>
<td>Q6:</td>
<td>!++-+--</td>
<td>+------</td>
<td>++++++</td>
<td>!+++++</td>
<td>...+</td>
<td>...+</td>
<td>...+</td>
<td>...+</td>
<td>0.10</td>
</tr>
<tr>
<td>Q7:</td>
<td>!111-111</td>
<td>1101011</td>
<td>110-100</td>
<td>.0100.0.</td>
<td>...+</td>
<td>...+</td>
<td>...+</td>
<td>...+</td>
<td>!15!</td>
</tr>
<tr>
<td>Q8:</td>
<td>1100+.111</td>
<td>10111011</td>
<td>.1101001</td>
<td>.0100.0.0</td>
<td>...+</td>
<td>...+</td>
<td>...+</td>
<td>...+</td>
<td>0.10</td>
</tr>
<tr>
<td>Q9:</td>
<td>...0.1...</td>
<td>...0.1-</td>
<td>...0.1-</td>
<td>...0.1-</td>
<td>...0.1-</td>
<td>...0.1-</td>
<td>...0.1-</td>
<td>...0.1-</td>
<td>!31!</td>
</tr>
<tr>
<td>Q10:</td>
<td>!+0...</td>
<td>+0...</td>
<td>+0...</td>
<td>+0...</td>
<td>+0...</td>
<td>+0...</td>
<td>+0...</td>
<td>+0...</td>
<td>!15!</td>
</tr>
<tr>
<td>Q11:</td>
<td>!+0...</td>
<td>+0...</td>
<td>+0...</td>
<td>+0...</td>
<td>+0...</td>
<td>+0...</td>
<td>+0...</td>
<td>+0...</td>
<td>!31!</td>
</tr>
<tr>
<td>Q12:</td>
<td>!-+1...</td>
<td>+1...</td>
<td>+1...</td>
<td>+1...</td>
<td>+1...</td>
<td>+1...</td>
<td>+1...</td>
<td>+1...</td>
<td>0.10</td>
</tr>
<tr>
<td>Q13:</td>
<td>!-+0-1...</td>
<td>+0-1...</td>
<td>+0-1...</td>
<td>+0-1...</td>
<td>+0-1...</td>
<td>+0-1...</td>
<td>+0-1...</td>
<td>+0-1...</td>
<td>!31!</td>
</tr>
<tr>
<td>Q14:</td>
<td>!-+-+1...</td>
<td>+1-+1...</td>
<td>+1-+1...</td>
<td>+1-+1...</td>
<td>+1-+1...</td>
<td>+1-+1...</td>
<td>+1-+1...</td>
<td>+1-+1...</td>
<td>!15!</td>
</tr>
<tr>
<td>Q15:</td>
<td>!-+0+10...</td>
<td>+0+10...</td>
<td>+0+10...</td>
<td>+0+10...</td>
<td>+0+10...</td>
<td>+0+10...</td>
<td>+0+10...</td>
<td>+0+10...</td>
<td>0.10</td>
</tr>
<tr>
<td>Q16:</td>
<td>!-+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>!31!</td>
</tr>
<tr>
<td>Q17:</td>
<td>!-+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>!15!</td>
</tr>
<tr>
<td>Q18:</td>
<td>!-+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>!31!</td>
</tr>
<tr>
<td>Q19:</td>
<td>!-+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>!15!</td>
</tr>
<tr>
<td>Q20:</td>
<td>!-+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>!31!</td>
</tr>
<tr>
<td>Q21:</td>
<td>!-+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>!15!</td>
</tr>
<tr>
<td>Q22:</td>
<td>!-+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>!31!</td>
</tr>
<tr>
<td>Q23:</td>
<td>!-+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>!15!</td>
</tr>
<tr>
<td>Q24:</td>
<td>!-+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>+0-11...</td>
<td>!31!</td>
</tr>
<tr>
<td>Q25-32:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.10</td>
</tr>
<tr>
<td>Q25-32:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.10</td>
</tr>
<tr>
<td>Q25-32:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.10</td>
</tr>
<tr>
<td>Q25-32:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.10</td>
</tr>
<tr>
<td>Q25-32:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.10</td>
</tr>
<tr>
<td>Q25-32:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.10</td>
</tr>
<tr>
<td>Q25-32:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.10</td>
</tr>
<tr>
<td>Q25-32:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.10</td>
</tr>
<tr>
<td>Q25-32:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.10</td>
</tr>
</tbody>
</table>

**Flame**

**Recovered differential paths**

differential path 1
differential path 2
Yet unknown chosen-prefix collision attack

1. Other differential path family
   - Same message differences for all 4 near-collision attacks (up to sign)
     $\delta m_4 = 2^{31}$, $\delta m_{11} = \pm 2^{15}$, $\delta m_{14} = 2^{31}$ (same as [WY04])
   - No systematic elimination using $\delta m_{11} = \pm 2^b$ as in [SSA+09]

2. Yet unknown birthday search
   - Birthday search is preprocessing phase to find differences that can be cancelled by differential path family
   - Less flexible differential path family $\Rightarrow$ Higher complexity birthday search
   - Approx. $2^{49}$ MD5 compression function calls
   - To compare: our attack in total has average complexity of $2^{44.6}$ MD5 calls
Yet unknown chosen-prefix collision attack

3. Yet unknown differential path construction algorithm
   – Differences in all bit positions of Q6 in all 4 near-collision attacks
   – Not a characteristic of known construction algorithms
   – Their connection search-space strictly contained in our search-space
   – Initial tests show this approach to be significantly slower than our approach
   – Probability successful connection over 4 steps drops approx. by factor $2^{-13}$

For a more extensive analysis, see [http://eprint.iacr.org/2013/358](http://eprint.iacr.org/2013/358)
Part IV – Improvements

To Conclude…
Improving counter-cryptanalysis

- Reduce chance at false negatives
- Need ‘exhaustive’ list of $(\delta B, \delta WS)$

- Case SHA-1
  - Need ‘exhaustive’ list of feasible disturbance vectors
  - Heuristic searches have found only two interesting classes of D.V.s
  - Make selection based on cost function, e.g. [Ste12]

- Case MD5
  - Finding feasible $\delta B$ unfinished work
  - Literature focused on finding attacks better than Wang et al.’s
  - W.I.P. heuristic ‘exhaustive’ surveys for $\delta B$
  - Using cost function determining lower-bound for attack complexity
Improving counter-cryptanalysis

- Each $\delta B, \delta WS$-guess costs 1 full compression function call
- Speed-up by early stop
  - Use very fast pre-check and only do full work with low probability
  - Without introducing possible false negatives
  - Find **unavoidable** conditions on message and state bits
    (conditions necessary for all possible feasible attacks based on $(\delta B, \delta WS)$)

- Case SHA-1
  - Determine unavoidable message bitrelations over steps 30-70
  - E.g., using exact joint local-collision analysis [Ste12]
  - Expect at least 4 bitrelations per Disturbance Vector
  - Would result in total cost factor < 2 instead of 16
Improving counter-cryptanalysis

Case MD5 (per $\delta B, \delta WS_i$)

- More complex, multiple types of pre-checks
- Do check whether transitions $(0,0,0,0) \leftrightarrow (2^{31}, 2^{31}, 2^{31}, 2^{31})$ happen correctly
- Determine lower-bound on attack complexity $< 2^{64}$
- Determine $K =$ # last steps that may vary s/t complexity $< 2^{64}$
- Do check whether $(2^{31}, 2^{31}, 2^{31}, 2^{31}) \rightarrow (2^{31}, 2^{31}, 2^{31}, 2^{31})$ happen correctly for first 16-K steps of round 4
- Find unavoidable bitconditions in round 2 & 3
  - i.e., check whether forcing negated bitcondition results in complexity $\geq 2^{64}$
To conclude…
Conclusions

- Migrate away from MD5 and SHA-1 based signature schemes
- The easy answer of “migrate to more secure signature schemes” is not a practical solution for the near future
- Instead allow old signatures, but protect verifiers against forgeries
- Real-time signature forgery detection possible
  - works for collision attacks on MD5 & SHA-1 [Ste12]
  - recovers full differential paths
- Reference implementation available
  - Feedback requested!
  - Let me know where it is used!
Conclusions

- Flame uses chosen-prefix collision attack ‘in the wild’
  - But an entirely new variant!
  - Different differential path family than [SSA⁺09]
  - Yet unknown birthday search
  - Yet unknown block-wise elimination procedure
  - Yet unknown differential path construction algorithm
  - New attack has higher complexity than [SSA⁺09]
Open questions

- Who made Flame?
  - Evidence points to world-class cryptanalysts, not just hackers
  - Adds to predominant speculation of nation-state behind Flame

- Why develop a new variant attack before Feb 2010?
  - But our attack implementation is public since June 2009 (see [Ste12])
  - Requires large effort, done in parallel
  - Nevertheless: exposes their cryptanalytic knowledge
Open questions

- Will the first successful SHA-1 attack be due to scientific efforts or not?
  - Recent years have shown almost no public efforts on SHA-1.
  - Flame’s attack on MD5 was developed independently and in parallel to public scientific efforts.
  - Perhaps attacks on SHA-1 are as well...
  - Nation-states have more computing resources than academics...

- Counter-cryptanalysis
  - New paradigm against cryptanalytic attacks
  - Collision detection first practical example
  - Can we construct other (practical) examples against known cryptanalytic attacks?
Thank you for your attention

Questions?
References


[Kam04] MD5 considered to be harmful someday, Dan Kaminsky, 2004.


[LdW05] On the possibility of constructing meaningful hash collisions for public keys, A.K. Lenstra, B. de Weger


[SLdW07] Chosen-prefix collisions and colliding X.509 certificates for different identities, M. Stevens, A.K. Lenstra, B. de Weger,

[Sot12] Analyzing the MD5 collision in Flame, Alex Sotirov,
SummerCon conference, New York, June 2012.

[SSA+09] Short Chosen-Prefix Collisions for MD5 and the Creation of a Rogue CA Certificate, M. Stevens, A. Sotirov, J. Appelbaum, A.K. Lenstra, D. Molnar, D.A. Osvik, B. de Weger,

(See also the open-source project at: http://code.google.com/p/hashclash/)

[WY04] How to break MD5 and other hash functions, X. Wang, H. Yu,