Rainbow Tables: Past, Present, and Future

James Nobis (quel)

http://www.freerainbowtables.com

DFW Security Professionals, 2011

Overview

- Personal bio
- Ethics
- Export law
- Applications summary
- Background information
- History from 1978 to present
- Technical and theory 1978 - ?
- Applications and details

Personal bio

- What is freerainbowtables.com?
- How much money do you make?
- Who is this guy?
  - Official day job title: Senior System Administrator / Developer
  - *nix - Linux, OpenBSD, NetBSD
  - coder in several languages
  - amateur crypto hobbyist
  - BS Computer Science
  - not in the sec industry

Ethics

- Is this legal?
  - Yes!
- You are making money helping crack passwords!
- Why are you helping the bad guys?
- If it can be done, should it done?
  - equip itsec community with tools
  - educate people to salt hashes er or hash at all
  - basic theory is well established in published work
  - basic implementation source was public before us

Export law

- Failure to comply is effective equivalent to espionage
- Obama changed the rules
  - public publication 2010-06-25
  - rules went effective 2010-08-24
  - open source is no longer exempt
  - closed source even freeware almost always requires a full BIS review
- 740.17 (b)(1) - doesn't apply - this is cryptanalytic
- Key size doesn't matter except in 1 case!
- 740.17 (b)(2) - software covered
  - source code
  - modifiable crypto
    - only for effective key size greater 56bits close source
    - only for effective key size greater 64bits open source
  - penetration testing
  - cryptanalytic

Applications summary

- Block cipher keys
  - DES
  - rc4 used by Microsoft Office - see Elcomsoft
    - Don't use 40-bit keys for 128-bit RC4
- Password hashes
  - Salt your hashes!
  - Security auditing
    - Password policy enforcement
    - migrating unsalted hashes to SSHA
    - Windows hashes aren't going anywhere

Background information

- Keyspace
- Brute force
- Types of cryptographic attack
- Hashes

Keyspace Ciphers

$$\text{keySpace} = 2^{\text{keyLength}}$$
$$\text{DES 56bit} = 2^{56}$$
$$\text{RC4 128bit} = 2^{128}$$

Keyspace Passwords

$$\text{keySpace} = \sum_{i=minPassLen}^{maxPassLen} charSetLen^{i}{}_{[1]}$$

$$= \frac{charSetLen^{maxPassLen+1} - charSetLen^{minPassLen}}{charSetLen - 1}$$

$$\begin{aligned} \text{loweralpha} &= [a - z] \\ \text{charSetLen} &= 26 \\ \text{maxPassLen} &= 10 \end{aligned}$$

$$\frac{26^{10+1} - 10^{1}}{26 - 1} = \frac{26^{11} - 10}{25} \approx 146813779479510 \approx 2^{47.061}$$

Brute force

- OpenSSL 0.9.8p
  - i686 linux on a dual core Intel T2080 @1.73Ghz
    - 16 size blocks: 3005871 md5's in 3.01s
    - 16 size blocks: 6840081 des cbc's in 3.00s
    - 16 size blocks: 6307485 des-ecb's in 3.00s
  - x86_64 linux on an AMD Phenom II X6 1090T 3.2-3.6Ghz
    - 16 size blocks: 8331538 md5's in 3.01s
    - 16 size blocks: 13213076 des cbc's in 2.99s
    - 16 size blocks: 13100170 des-ecb's in 3.00s

Brute force 2

$$2^{47.061} / 8331538 \;/\; 3.01 \approx 68 \text{ days}$$

$$2^{56}/13100170 \;/\; 3.00 \;/\; \approx 21221 \; days \;\approx\; 58 \text{ years}$$

$$2^{47.061} * 16 \approx 2136 \text{ TiB}$$

$$2^{56} * 7 \text{ bytes} = 524288 \text{ TiB}$$

Types of cryptographic attack

- Ciphertext-only (COA)
  - no extra knowledge
- Known-plaintext (KPA)
  - knowledge of some plaintexts and their ciphertexs
- Chosen-plaintext (CPA)
  - plaintext may be selected
- Chosen-ciphertext (CCA)
  - plaintext or ciphertext may be select

DES

- Initial Permutation
- 16 rounds
- Final Permutation

- Modes
  - Electronic CodeBook (ECB)
    - encrypt each block by itself
  - Cipher-block chaining (CBC)
    - Initialization Vector (IV)
    - each block of plaintext XOR with previous ciphertext block

Hashes

- MD4
  - 3 rounds
  - 128-bit output - 16 bytes
- MD5
  - 4 rounds
  - 128-bit output - 16 bytes
- SHA1
  - 80 rounds
  - 160-bit output - 20 bytes

History from 1978 to present

Probabilistic forms of time and memory tradeoffs

- Hellman tables
- Rainbow tables

Hellman tables

- "A Cryptanalytic Time - Memory Trade-Off" [2]
  - by Martin Hellman
  - published 1980
- pre-computation N work
- Chosen Plaintext Attack (CPA)
- ECB allows also Ciphertext Only Attack (COA)
- Reduction function
- memory costs more than cpu time
- fixed length chains
- 128-bit keys are the minimum for any data

Rainbow Tables

- Philippe Oechslin 2003 [5]
- Fixed length chains
- less memory and time than previous methods
- chains may collide without merging
- higher success rates possible with less memory
- few tables
- reduction function per column

Technical and theory from 1978 - ?

- Hellman tables
- Rainbow tables

Hellman Tables Technical 1

- Starting Points (SPs) chosen randomly
- End Points (EPs)
- Generate chains
- Discard all intermediate points
- Sort on EP
- Reduction function
  - Hellman suggests dropping the last 8 bits of ciphertext
  - 64-bit to 56-bit
- lookups
  - cpu N ^ (2/3)
  - memory N ^ (2/3)

Hellman Tables Technical 2

If f(*) is a random function mapping the set 1, 2, ..., N onto itself, and the key K is chosen uniformly from the same set, the probability of success is bounded by

$$P(S) \geq (1 \; / \; N) \sum_{i=1}^{m} \sum_{j=0}^{t-1} [(N - it)/N]^{j+1}$$

brute force point: $mt^2 = N$

$m =$ numChains

$t =$ chainLength

$N =$ keySpace

Hellman Tables Technical 3

- For N ˆ (2/3) time complexity use:
  - m = t = N ˆ (1/3)
- P(S) is approximately N ˆ (-1/3) for a single table
- Generate N ˆ (1/3) tables or more
  - (2ˆ 56)ˆ (1/3) is a min 416127 tables
- False Alarm
  - a given ith element of a chain with multiple inverses
  - Expected false alarms

$$
\begin{aligned}
E(F) &\leq \sum_{i=1}^{m} \sum_{j=1}^{t} j/N \\
&= mt(t+1)/2N \\
&\leq 50\% \text{ total cryptanalytic attack}
\end{aligned}
$$

Rainbow Tables Technical 1

Original Hellman Formula as revised by Oechslin

$$P_{table} \geq \frac{1}{N} \sum_{i=1}^{m} \sum_{j=0}^{t-1} (1 - \frac{it}{N})^{j+1}$$

$$P_{success} \geq 1 - \left(1 - \frac{1}{N} \sum_{i=1}^{m} \sum_{j=0}^{t-1} (1 - \frac{it}{N})^{j+1} \right)^{\ell}$$

Diminishing returns on the success rate for more tables
Cost of time and memory rapidly grows
reduction function 1 to t - 1, where t is chain length [5]
Does this all look and sound familiar? It should

Rainbow Tables Technical 2

- rainbow chains may collide without merging
- if the collision is at the same point in the chains it's a merge
- chance that 2 specific colliding chains will merge

$$P_{collision\ merges} = \frac{1}{t}$$

- The chance of a merge for the entire table is given as a consequence of perfect tables and explained later
- False alarms may be up 125% of the cryptanalytic attack [6]

Oechslin gives us the success rate of this new approach for m chains of length t

$$P_{table} = 1 - \prod_{i=1}^{t}(1 - \frac{m_i}{N})$$

$$\text{where } m_1 = m$$

$$m_{n+1} = N(1 - e^{-\frac{m_n}{N}})$$

He refers in [5] and [6] as this being exact but it is not

This success rate probability is an average case

The lower bound is the original Hellman table success rate

Rainbow Tables Technical 4

For your sanity and mine lets rewrite the formulas [1]
success depends on predicting the expected unique chains [5]
success rate of a single table

$$euc(1) = chainCount$$

$$euc(i) = keySpace(1 - e^{-\frac{euc(i-1)}{keySpace}})$$

$$\text{tableSuccessRate} \approx 1 - \prod_{i=1}^{chainLength} (1 - \frac{euc(i)}{keySpace})$$

$$\text{totalSuccessRate} \approx 1 - \prod_{i=1}^{numTables} 1 - tableSuccessRate_i$$

$$\approx 1 - (1 - tableSuccessRate)^{numTables}$$

Rainbow Tables Technical 5

- Perfect Tables
  - general definition of perfect
    - no duplicate chains
    - no merging chains
- differences of perfect
  - some do not generate replacement chains
  - some require specific selection which merging chain is kept
    - pick the chain that merges where i is closest to chainLen
- Free Rainbow Tables definition
  - all general requirements
  - replacement chains generated for all discarded
  - no selection of the merging chain to discard
    - not ideal but has little to no impact on success rate
    - increases false alarm chain walks
    - ideal selection may create more costly false alarms

- Perfect Tables 2
  - A single perfect table is too expensive to generate
    - Oechslin gives us Nt (keyspace * chainLen) [5]

$$2^{47} = 140737488355328$$
$$2^{47} * 40000 = 5629499534213120000$$
$$\approx 2^{62.2877}$$

  - For a given keyspace multiple perfect tables
    - lower success rate for each one
    - Oechslin estimates max success rate of a table as 86% [6]
  - If we assume 86% table success rate

$$1 - ((1 - 0.86) * (1 - 0.86)) = 98.04\%$$
$$1 - ((1 - 0.86) * (1 - 0.86) * (1 - 0.86)) = 99.7256\%$$
$$1 - ((1 - 0.86) * (1 - 0.86) * (1 - 0.86) * (1 - 0.86)) = 99.961584\%$$

Applications and details

- Overview of some well known implementations
- Overview of some of the file formats
- Detailed focus on Free Rainbow Tables

Rainbow Table Implementations

- Ophcrack 1.0a by Philippe Oechslin
  - 2004-07
  - 2005-03-31 Ophcrack 2.0
  - 2009-07-29 Ophcrack 3.3.1
- RainbowCrack 1.0 by Zhu Shuanglei
  - 2003-09-09
  - 2003-11-21 RainbowCrack 1.2 - last release with source
- Winrtgen as part of Cain & Abel
- Free Rainbow Tables
  - 2006-12-06 first DistRTgen release
  - 2007-01-17 first linux release
  - 2007-12-02 Perfect table generation

File formats

- .rt - 2003
  - fixed 8 byte SPs and 8 byte EPs per chain
- Ophcrack - 2004
  - fixed 4 byte sequential SPs and 2 byte EPs
  - prefix index 4 bytes per [9]
  - prefix method first noted [6]
- .rti - RT Improved - 2008
  - prefix index
  - 8 bytes per chain
  - 11 bytes per index
- .rtc - RT Compact - 2009 August
- .rti2 - RT Improved v2 - 2009 June
- .rti2 - with headers - I have code to finish

Free Rainbow Tables

- Generation
- BOINC
  - Work Unit (WU) assignment
  - WU computation
  - WU upload
  - WU validation
  - WU assimilation
  - Table perfecting
  - Table completion
- Using the tables
- Distribution

Generation 1

- Picking a table set
  - user feedback and needs
  - algorithm or code change testing
  - regenerating non-sequential table sets for optimal packing
  - rivalries
    - project-rainbowcrack.com has this set at 96% success that's awful, 99.9% lets go

Generation 2

- Generation Parameters
  - A day of idle machines is better than generating poor tables
  - totalSuccessRate = 99.9%
  - pick chainLen
    - balance generation time and cryptanalysis time
    - balance total disk use
    - 40,000 has worked well for CPU only generate/crack
    - 20,000 at double the numChains for the first table for speed
  - pick number of tables
    - for the totalSuccessRate 4 is optimal
    - The 86% success per table is optimistic from Oechslin [6]
    - even at 86% success 3 tables at best yields 99.725%
  - pick character set
  - pick passLen
  - calculate ExpectedUniqueChains
  - pick algorithm - minor impact on stepSpeed

Generation 3

$$
\begin{aligned}
totalSuccessRate &= 99.9\% \\
chainLen &= 40,000 \\
charSet &= loweralpha[a - z] \\
charSetLen &= 26 \\
minPwLen &= 1 \\
maxPwLen &= 10 \\
keySpace &= 146813779479510 \\
chainCount &= 46417863961 \\
euc &= 6338323552 => euc = 6338500000 \\
totaleuc &= 25354000000 \\
expectedTableSuccessRate &= 1 - (1 - \frac{6338500000}{146813779479510})^{40000} \\
&\approx 82.218060806682030987\%
\end{aligned}
$$

Generation 4

$$expectedTableSuccessRate \approx 82.21806080668203\%$$
$$expectedTableSuccessRate * 2 \approx 96.83802638525142\%$$
$$expectedTableSuccessRate * 3 \approx 99.43773977451665\%$$
$$totalTableSuccessRate \approx 99.90001922859634\%$$
$$\text{md5\_loweralpha\#1-10\_0 } actualSuccessRate \approx 82.42099823126382\%$$
$$\text{md5\_loweralpha\#1-10\_1 } actualSuccessRate \approx 82.49167591074124\%$$
$$\text{md5\_loweralpha\#1-10\_2 } actualSuccessRate \approx 82.51509187185517\%$$
$$\text{md5\_loweralpha\#1-10\_3 } actualSuccessRate \gtrapprox 82.40308898901675\%$$
$$\text{md5\_loweralpha\#1-10\_[01] } actualSuccessRate \approx 96.92221139867314\%$$
$$\text{md5\_loweralpha\#1-10\_[012] } actualSuccessRate \approx 99.46185149067949\%$$
$$\text{md5\_loweralpha\#1-10 } actualSuccessRate \gtrapprox 99.90530248570794\%$$

BOINC 1

- Work Unit (WU) assignment
  - parameters including start point ranges
  - 500000 chains per WU for latest MD5 run
  - md5 loweralpha 1 10 2 40000 500000 27981500000 35100,35400,35700,36000,36300,36600,36900,37200,37500,37800 ,38100,38400,38700,39000,39300,39600
- WU computation
  - generate a chain for every start point
  - sequential start points
  - CPU
    - 2-3 hours for 1 WU on a single core (MD5)
  - GPU
    - 105 seconds GTX 470 x86_64 linux (MD5)
    - about 120 seconds on windows
    - yes it's the same code and no I'm not in charge of windows builds

BOINC 22

- WU upload
  - completed WU 9,000,000 bytes = (8*2+2) * 500000
    - traditional rt 8 bytes for SP and 8 for EP
    - 2 bytes for checkpoints
  - GPUs are fast
  - 5,000,000 bytes is better than 9,000,000
    - sequential SPs - let the server add the SPs and sort
  - my upstream is awful (384kbps)
    - 109 seconds per WU
- WU validation
  - regeneration of some chains

BOINC 3

- WU assimilation
  - combine WUs into parts
- Table perfecting
  - adding parts into the table in progress
  - sort on end points
  - no duplicates to remove with sequential SPs
  - remove merges (identical EPs)
    - generate replacement chains
- Table completion
  - convert to rti/rti2 format
  - upload to mirror seed
  - seed upload to primary mirror

Using the tables

```
[quel@paranoia ] cat hashes.txt
2c678f2e67902fb8294e15f6d44cc3e1
b172647b25385aef84620de9b5d194ad

[quel@paranoia ] time ./rcracki_mt -t 3 -l hashes.txt -o results.txt
/mnt/rainbow_tables/freerainbowtables/md5/md5_loweralpha#1-
10_?
Using 3 threads for pre-calculation and false alarm checking...
Found 194 rainbowtable files...

md5_loweralpha#1-10_0_40000x5284976_distrrtgen[p][i]_95.rti:
reading index... 19221191 bytes read, disk access time: 0.01 s
reading table... 42279808 bytes read, disk access time: 0.02 s
verifying the file... ok
searching for 2 hashes...
Pre-calculating hash 1 of 2.
```

Using the tables 2

md5_loweralpha#1-10_1_40000x67108864_distrrtgen[p][i]_04.rti:
reading index... 243700963 bytes read, disk access time: 1.61 s
reading table... 536870912 bytes read, disk access time: 3.49 s
verifying the file... ok
searching for 2 hashes...
plaintext of 2c678f2e67902fb8294e15f6d44cc3e1 is kpcjdbsdr
cryptanalysis time: 0.87 s

md5_loweralpha#1-10_1_40000x67108864_distrrtgen[p][i]_17.rti:
reading index... 243706705 bytes read, disk access time: 1.64 s
reading table... 536870912 bytes read, disk access time: 3.31 s
verifying the file... ok
searching for 1 hash...
plaintext of b172647b25385aef84620de9b5d194ad is lytoyswacd
cryptanalysis time: 0.45 s

Using the tables 3

plaintext found:	2 of 2 (100.00%)
total disk access time:	602.98 s
total cryptanalysis time:	91.45 s
total pre-calculation time:	356.57 s
total chain walk step:	3199760004
total false alarm:	55469
total chain walk step due to false alarm:	814610518

Using the Tables 4

- 2c678f2e67902fb8294e15f6d44cc3e1
- kpcjdbsdr
- hex:6b70636a6462736472

- b172647b25385aef84620de9b5d194ad
- lytoyswacd
- hex:6c79746f797377616364

real	18m9.844s
user	22m12.171s
sys	1m2.880s

Distribution

- A different trade off
  - fast
  - reliable
  - lots of space
  - cheap
- seed source
- mirrors
- torrents

Hybrids

- hybrid2
  - each charset is a sub-keyspace
  - code is complete and deployed for CPUs
  - code is nearly complete for GPUs
  - next up ntlm [A-Z][a-z]{5}[a-z0-9]{2}[0-9]{1,3}
    - yes that's length 9, 10, or 11!
    - at 99.9% success
  - give us feedback on what tablesets to do
- full sub-keyspace support
  - allows the table to be ordered for faster attacks
  - theory and sample code exist - all on our forums
  - possibilities get fairly interesting

Future

- GPU - for cryptanalytic attack side
- SSE2
- balacing CPU v GPU
- RTI2 with file headers - we're nearly there
- convert completed WU to rti2 prior to upload?
- selecting best merge to discard
- new system architecture for better resiliency
- distribute verifications

You can help!

- x86/x86_64 windows and linux users install BOINC
  - attach to the project
    - http://boinc.freerainbowtables.com/distrrtgen/
  - CUDA generation with current BOINC and video drivers
- spread the word
- donate
  - compute time
  - hosting
  - hardware
  - programming
  - testing
  - benchmarking
  - bug finding
  - bug fixing

FRT Links

- http://www.freerainbowtables.com
- http://gitorious.org/freerainbowtables-applications
- http://rcracki.sourceforge.net
- http://www.tbhost.eu
- http://boinc.berkeley.edu
- http://boinc.freerainbowtables.com/distrrtgen

Non-FRT Links

- http://www.cryptohaze.com
- http://www.project-rainbowcrack.com
- http://ophcrack.sourceforge.net
- http://www.iacr.org
- http://eprint.iacr.org/complete
- http://www.acm.org

Contact information

- James Nobis (quel)
- quel@freerainbowtables.com
- http://www.freerainbowtables.com
- GPG
  - pub 4096R/8B429E16 2010-02-05
  - 934B 3013 6826 BF6B BE93 750A 8081 124C 8B42 9E16
  - uid James Nobis ¡quel@freerainbowtables.com¿
  - sub 4096g/0312862A 2010-02-05
  - sub 4096R/A35ECB2E 2010-02-05
  - sub 4096R/F7C0F683 2010-11-25

QA

Sources

See my Passwords ^ 10 page as LaTex and I ran out of time.

Sources continued

Bruce Schneier. Applied Cryptography: Protocols, Algorithms, and Source Code in C. 2nd ed. New York, USA : Wiley, 1996. Print.

Niels Ferguson and Bruce Schneier. Practical Cryptography. 1st ed. New York: Wiley, 2003.Print.

Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno. Cryptography Engineering: Design Principles and Practical Applications. Indianapolis, IN: Wiley, 2010. Print.

OpenSSL timing data collection

- openssl-0.9.8p.tar.gz
- ECB: ./openssl speed -evp des-ecb

OpenSSL timing data detailed

- OpenSSL 0.9.8p
  - i686 linux on a dual core Intel T2080 @1.73Ghz
    - 16 size blocks: 3005871 md5's in 3.01s
    - 16 size blocks: 6307485 des-ecb's in 3.00s
    - 256 size blocks: 418564 des-ecb's in 2.99s
    - 1024 size blocks: 104909 des-ecb's in 3.00s
    - 8192 size blocks: 13119 des-ecb's in 3.00s
  - x86_64 linux on an AMD Phenom II X6 1090T @3.2Ghz - 3.6Ghz
    - 16 size blocks: 8331538 md5's in 3.01s
    - 16 size blocks: 13100170 des-ecb's in 3.00s
    - 256 size blocks: 853165 des-ecb's in 3.00s
    - 1024 size blocks: 213891 des-ecb's in 3.00s
    - 8192 size blocks: 26740 des-ecb's in 3.00s

Licensing