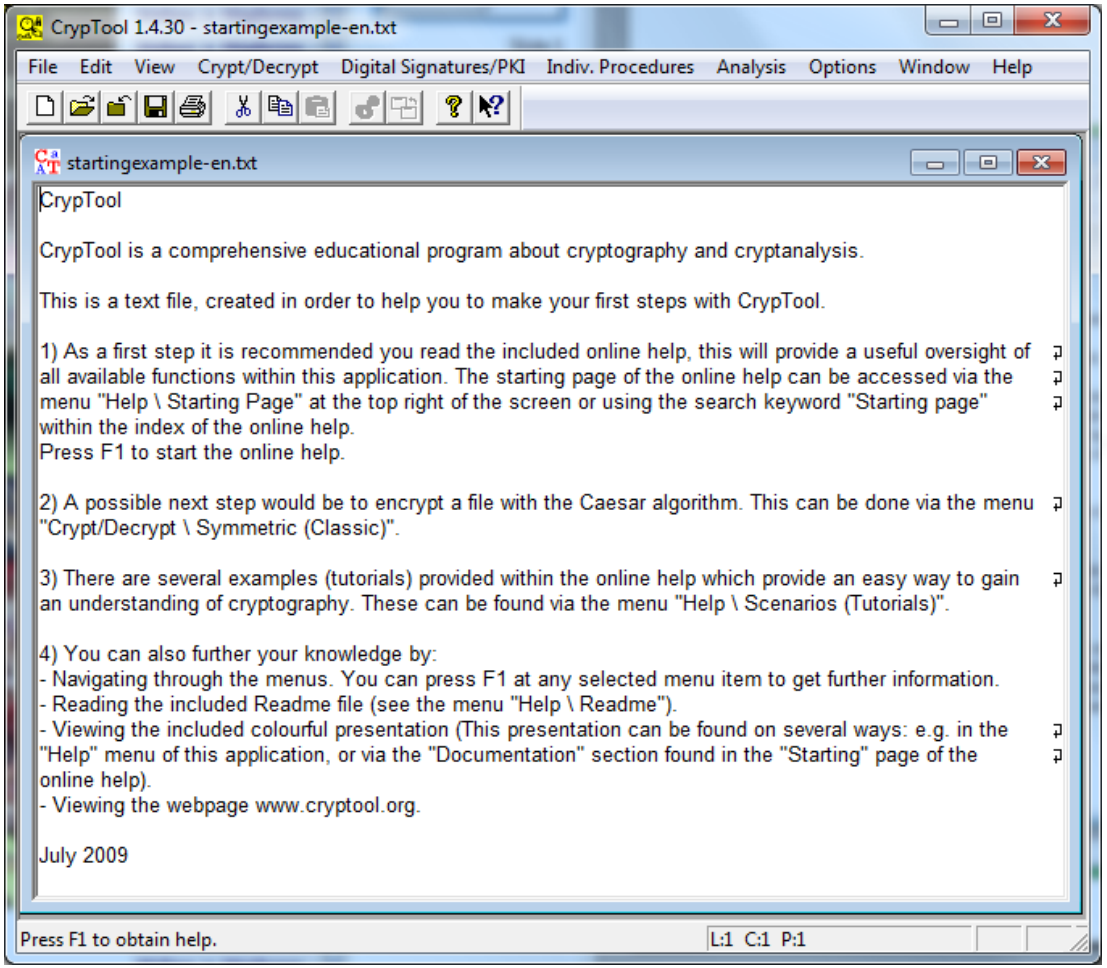
The golden age of hacking

Parallel Computing
GPGPU and
password/hash/crypto
attacks

CrypTool

Cryptography for the masses

Recommended!

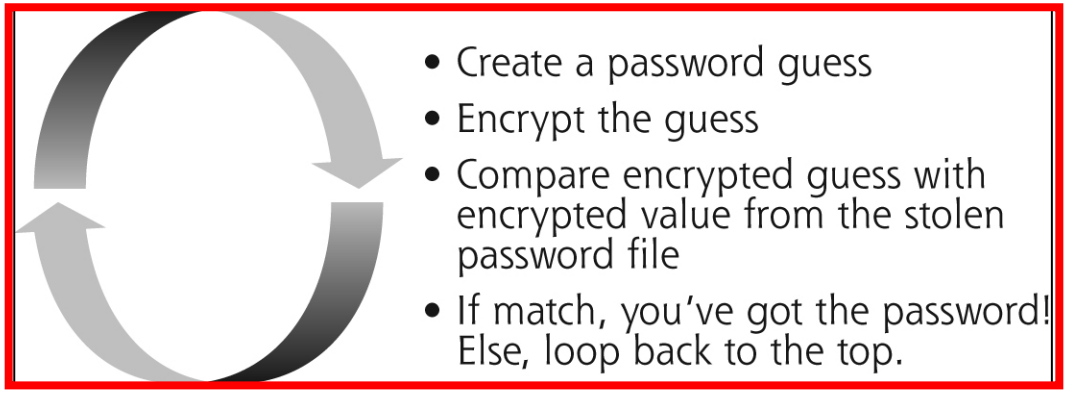


CrypTool (Freeware)
Crypt methods
Analysis
Visualisations
Etc. etc....
http://www.cryptool.org/

Password attacks I

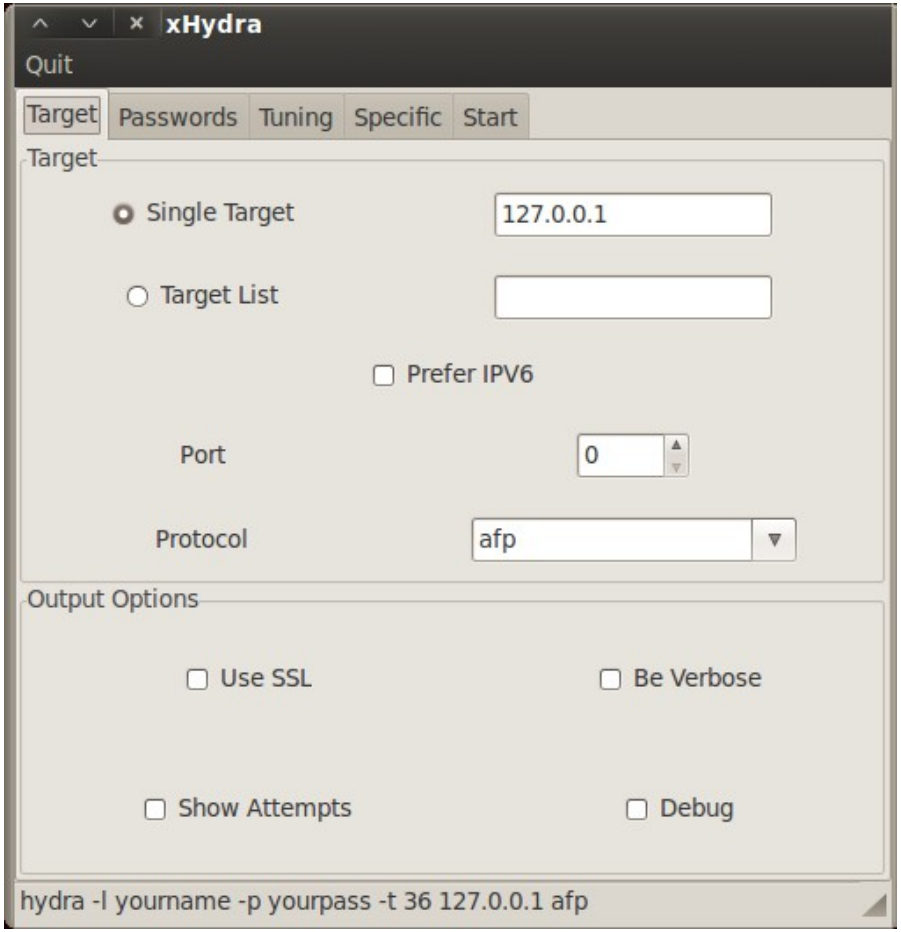
http://en.wikipedia.org/wiki/Password_cracking

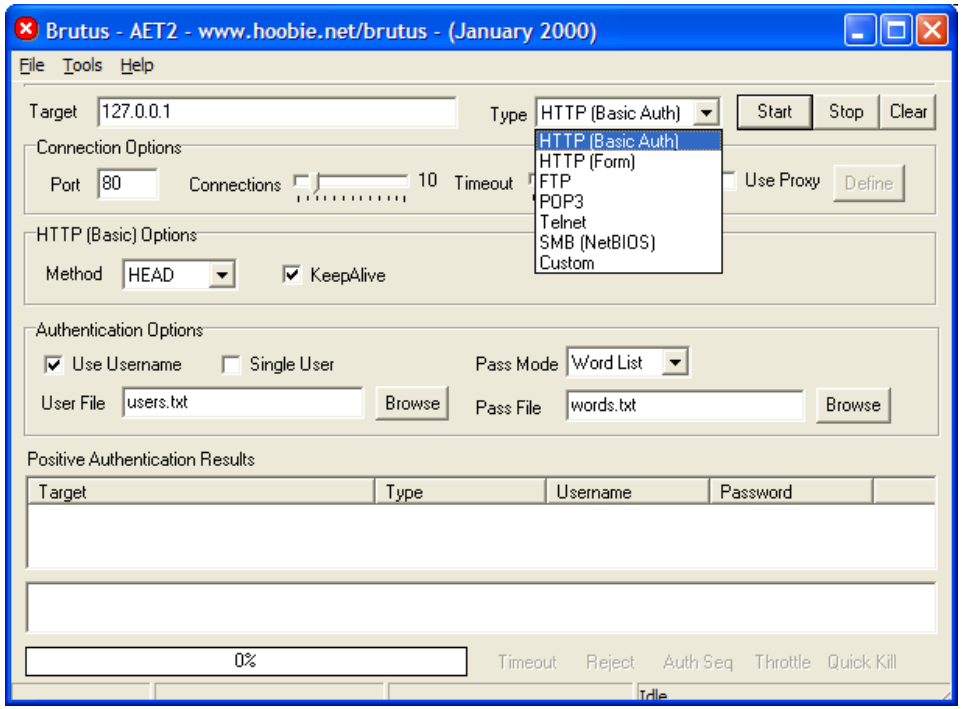
- Topic is already rather known by most of you!
  - At least the cracking part
- Default built in passwords in systems
- Network login tools (**password guessing**)
  - Brutus
  - THC Hydra
    - Supported services includes: TELNET, FTP, HTTP, HTTPS, HTTP-PROXY, SMB, SMBNT, MS-SQL, MYSQL, REXEC, RSH, RLOGIN, CVS, SNMP, SMTP-AUTH, SOCKS5, VNC, POP3, IMAP, NNTP, PCNFS, ICQ, SAP/R3, LDAP, PostgreSQL, Teamspeak, Cisco auth, Cisco enable, and Cisco AAA.
  - Reset attack
    - Account lockout (DoS)
- Passwords are either
  - Hashed or encrypted
  - Flaw in algorithm?
  - Entropy?



Network login tools

- THC (The Hackers Choice) Hydra Xhydra, #3
- Brutus, old, #10 on http://sectools.org





Entropy (password entropy)

2^6,56=95

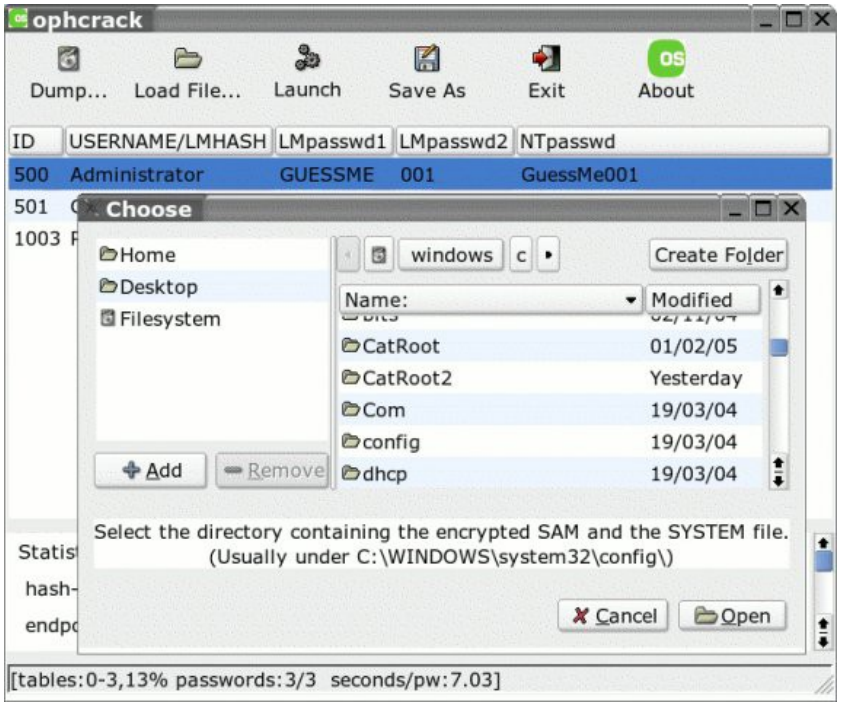
- Measuring password strength (disorder)
- For a completely random password, each character is worth approximately 6.56 bits
- With a user-chosen, first sign give 4 bits, characters 2-8 gives 2 bits, characters 9-20 gives 1.5 and 21-... provides 1 bit per character
- Entropy (bits) table for various lengths of passwords

Length (chars)	8	20	63
User-choosen (freely choosen)	18	36	79
User-choosen (according to rules)	30	42	85
Random	52	131	413

- The number of variations for a password is 2^(number of bits)
- However (number of possible characters)^(number of characters) is basically flawed because a short passwords can be complex and long passwords can be of an easily guessable character
- http://en.wikipedia.org/wiki/Password_strength

Password attacks II

- Dictionary → Wordlists
  - Google search ”wordlist compilation”
- Wordlists vs brute-force
  - keyspace_password.xls
  - Hybrid attacks (permutations)
  - Distributed attacks - botnets
- The best free tools
  - Cain & Abel (#1)
    - Swiss army knife
  - John the ripper (#2)
    - Multi platform, permutation
- SAM registry/DB file attacks
  - Online
    - Cain, fgdump, etc.
  - Offline
  - Bootdisks
    - Ophcrack (Vista/7)
  - Reset

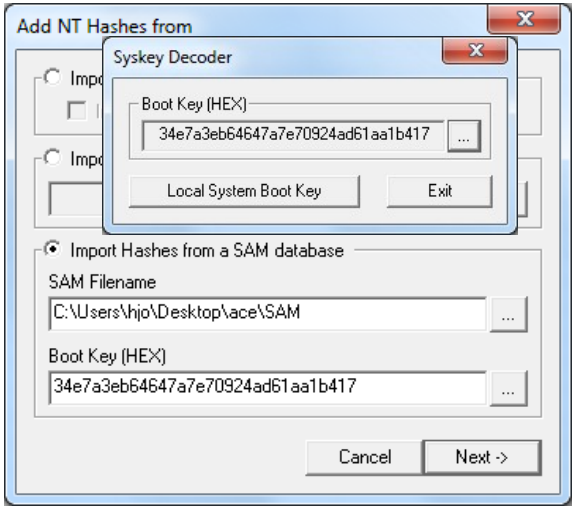


Offline extraction of credentials

syskey.exe

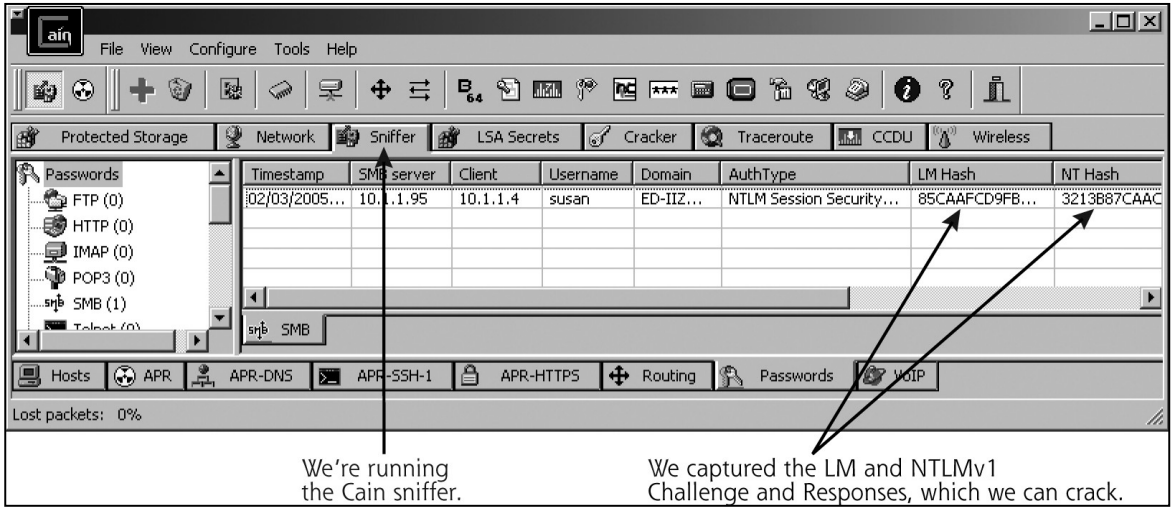
- Hash encryption in article “Syskey and SAM” at: http://moyix.blogspot.com/2008/02/syskey-and-sam.html
- Creddump (Python scripts)
  - LM and NT hashes (Syskey protected – 128 bits)
  - Cached domain credentials and LSA secrets
  - http://code.google.com/p/creddump/
- Other tools
  - Cain – from Forensic 1, lab 4.8
    - Add NT Hashes, Syskey Decoder (System), ...
  - SAMInside
  - Bkhive (dump syskey), Samdump2 etc.
- Tutorials: IronGeek

http://www.irongeek.com/i.php?page=security/cracking-windows-vista-xp-2000-nt-passwords-via-sam-and-syskey-with-cain-ophcrack-saminside-bkhive-etc

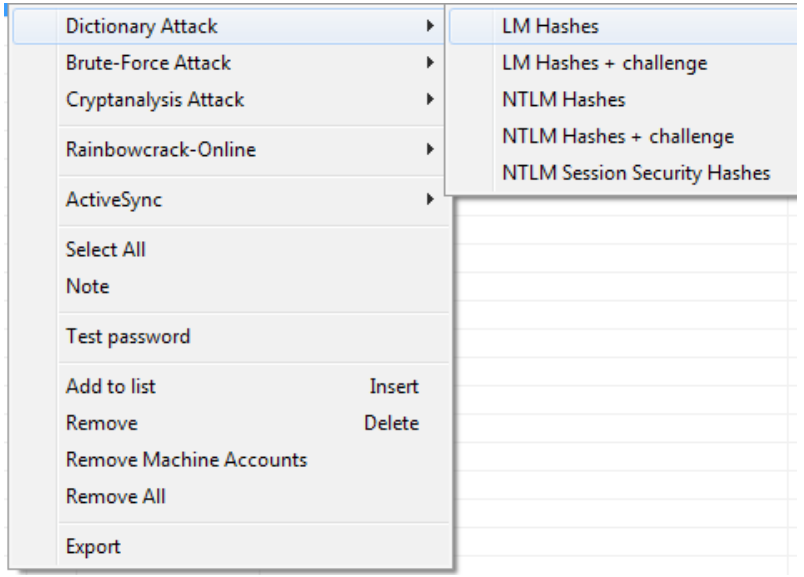


http://www.insidepro.com/

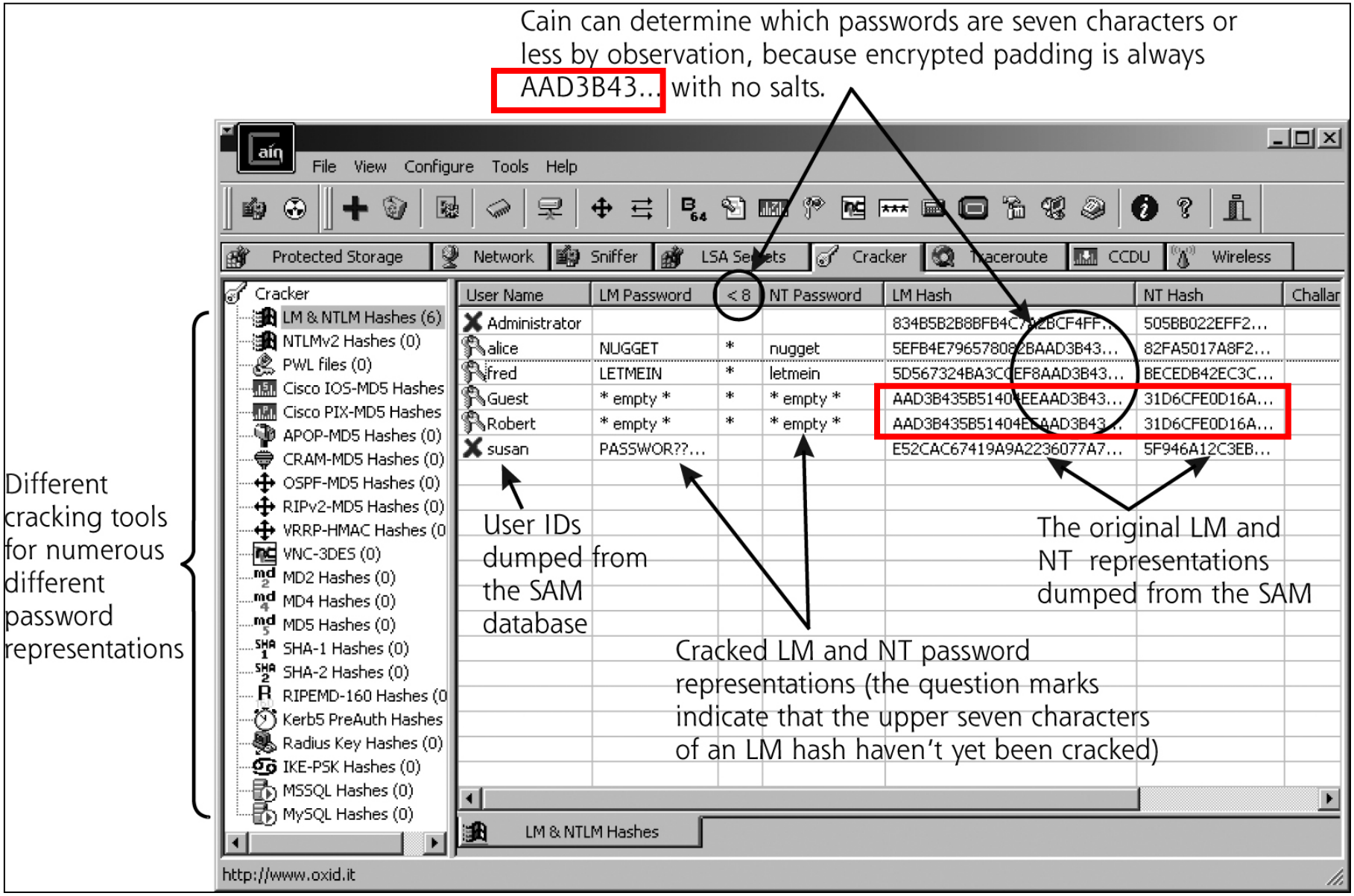
Cain and Windows hashes



- Microsoft LM or LanMan
  - The weak uppercase fixed 14 digit chopped …
- Windows NT(LM) hash
  - Much stronger than LM (MD4 x 3)
- Network sniffed authentication packets
  - LM challenge-response
  - NTLMv1 challenge-response
    - Stronger than LM challenge
  - NTLMv2 challenge-response
    - Stronger than NTLMv1
  - MS-Kerberos5 pre auth
    - The MS version used in most AD networks
- CHR page 137 and 386 – 396

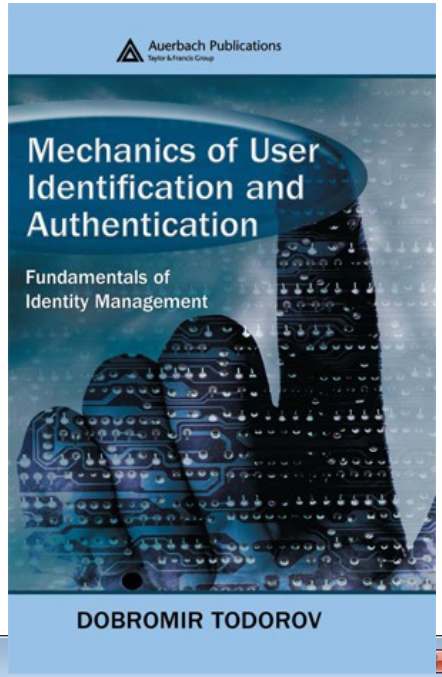


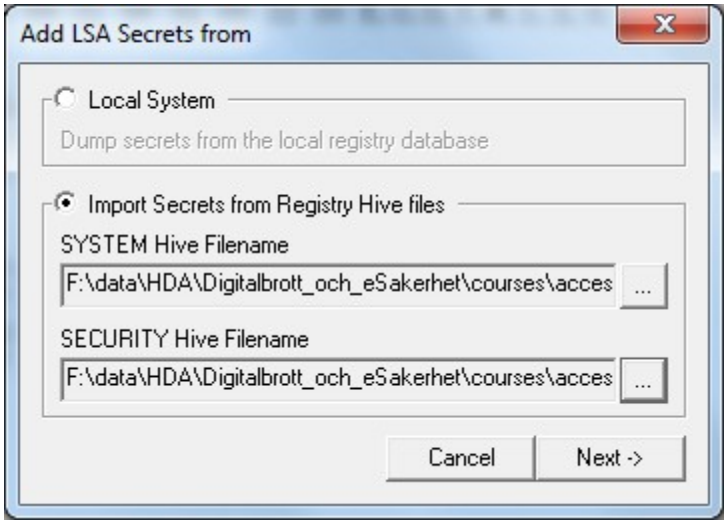
Cain and Windows hashes cont.

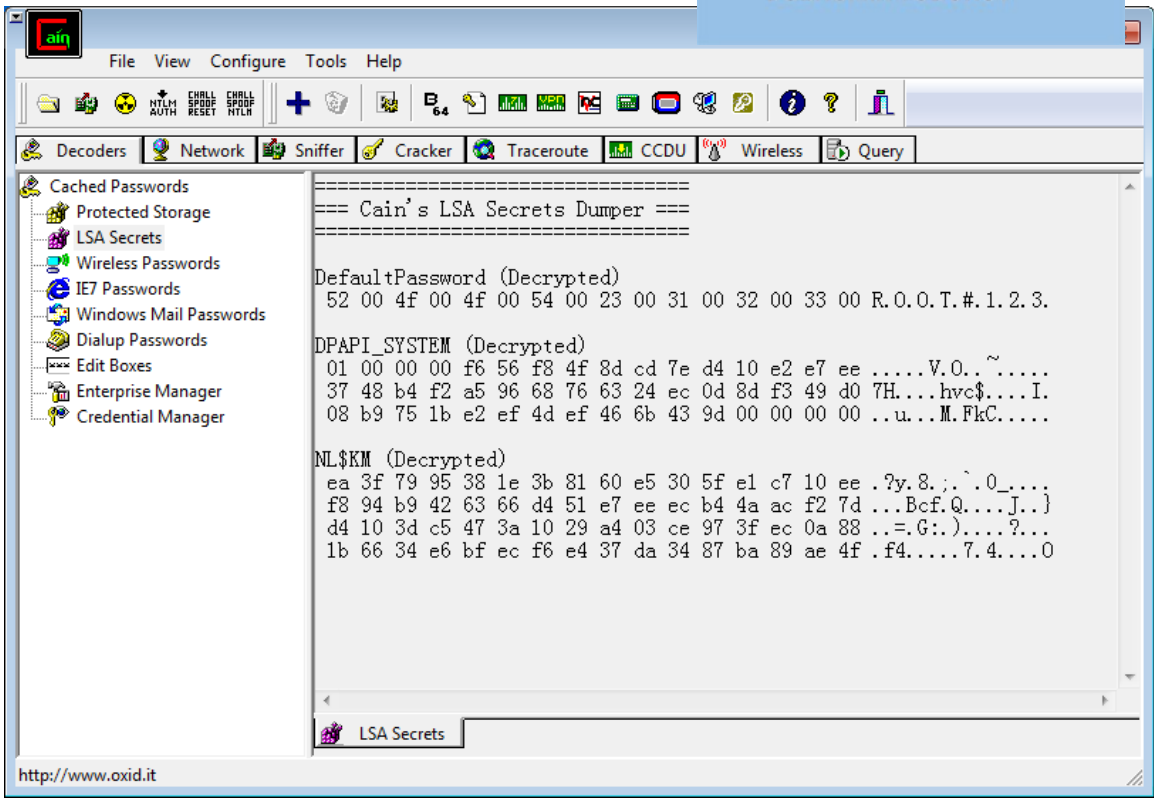


Cain LSA secrets

- Decrypting Local Security Authority (LSA) secrets you may find
  - DefaultPassword – used if auto-login is enabled
  - NL$KM – secret key used to encrypt cached domain passwords
  - Various service account secrets, $MACHINE.ACC, etc...
- DPAPI_SYSTEM is a legacy backup key that is used to recover DPAPI (Data Protection Application Programming Interface) data
- Very good book describing algorithms



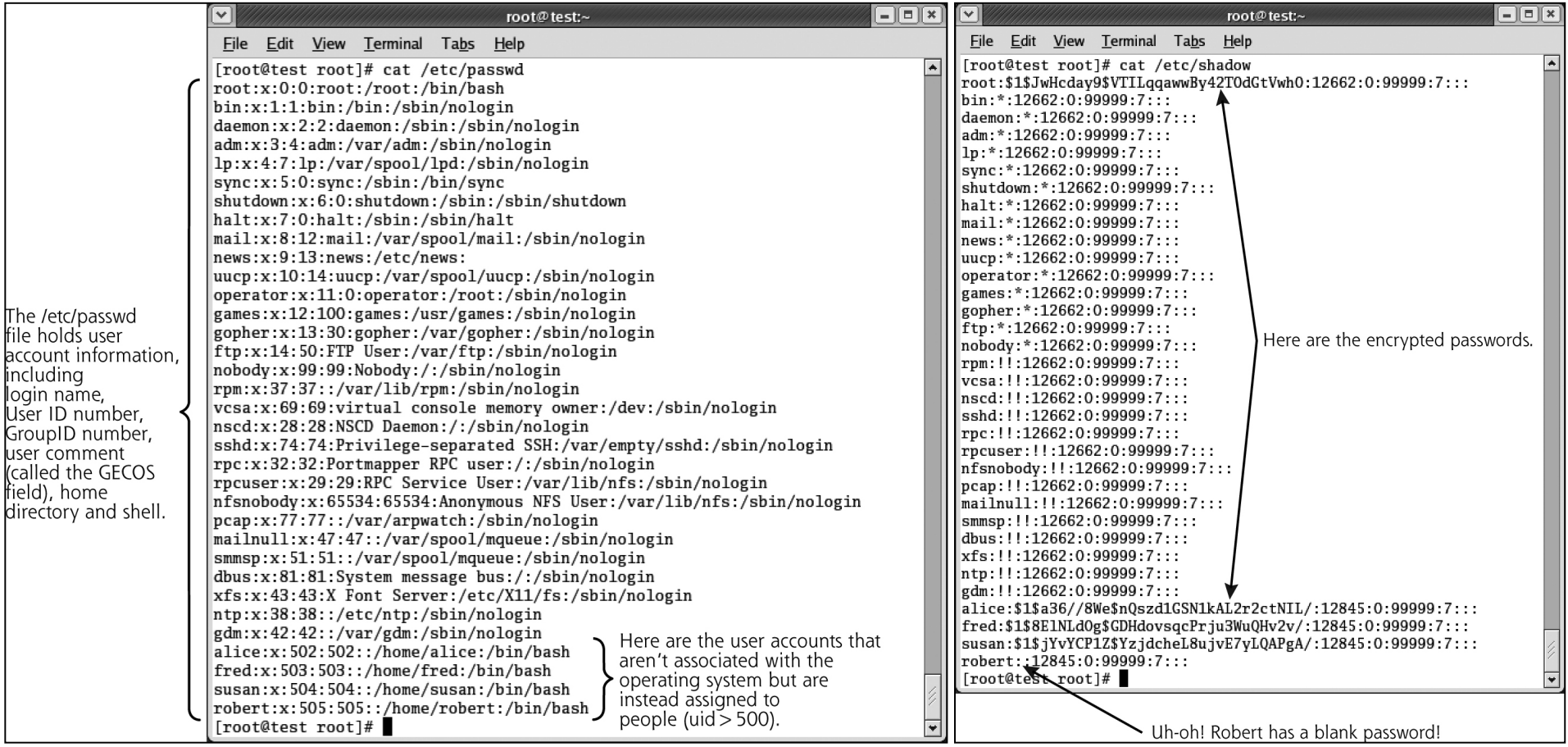




Unix passwd and shadow files

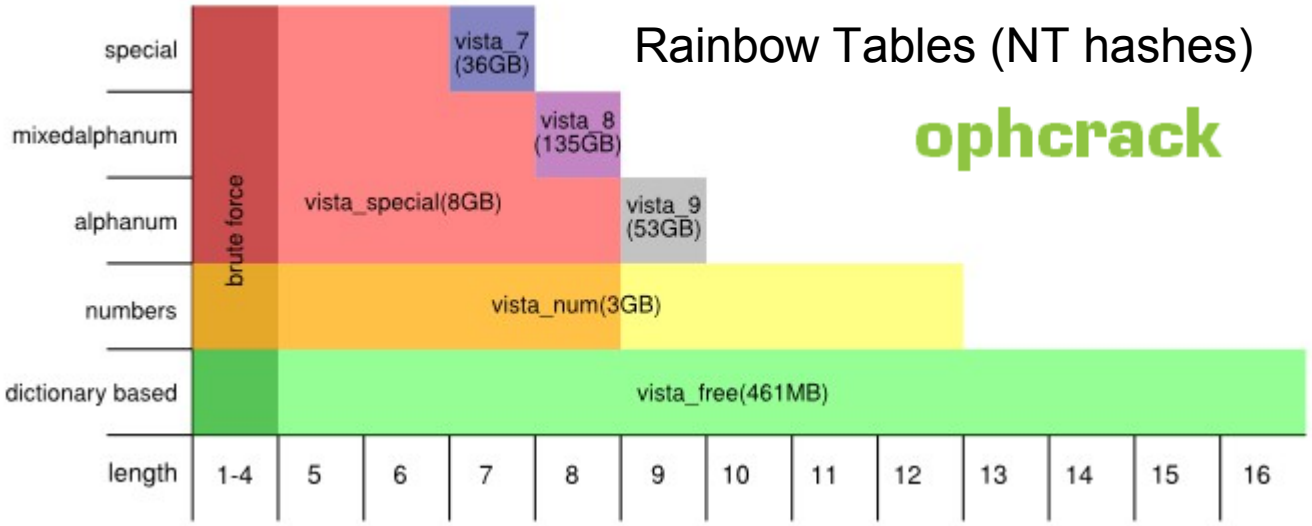
- Salt (a small extension, 2-8 byte) often used to complicate rainbow attacks
  hash = OWF(password + salt) - Unix use salt, Windows does not
  http://en.wikipedia.org/wiki/Salt_%28cryptography%29

"1"=MD5, "5"=sha-256, "6"=sha-512, (man shadow, crypt (3))



Rainbow tables

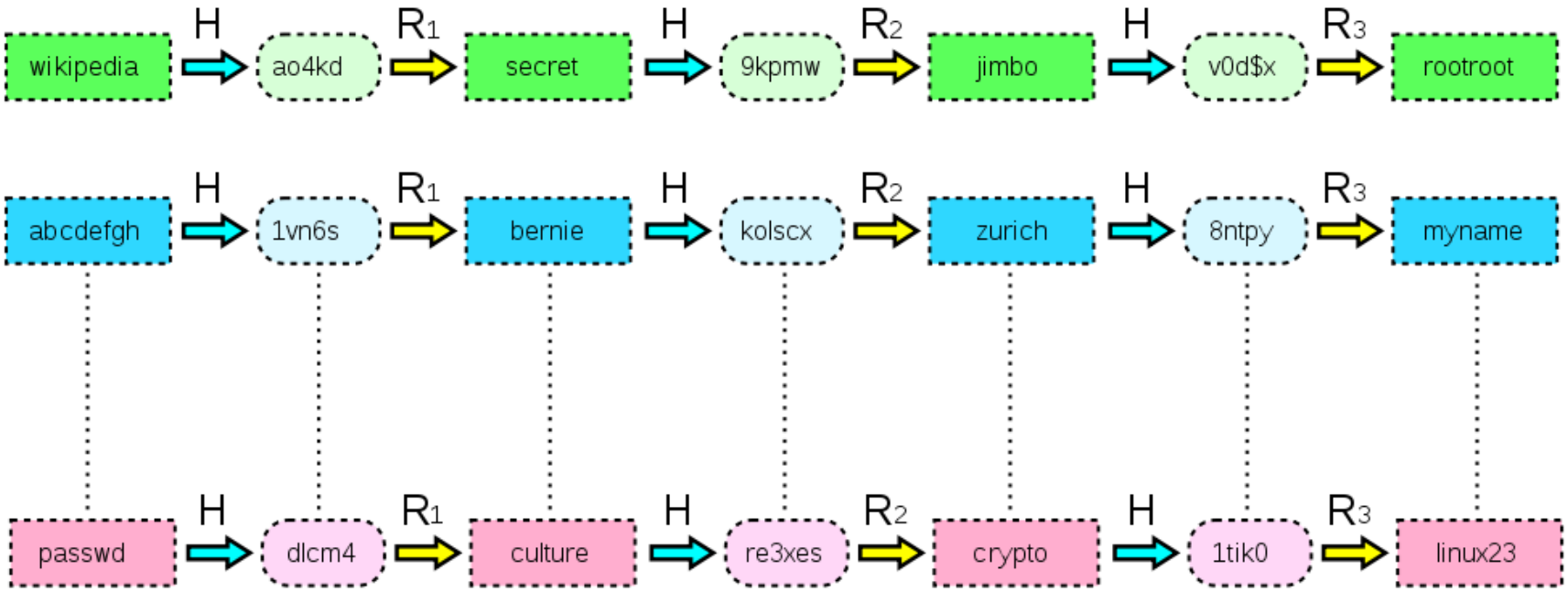
- A refinement (by Philippe Oechslin) of an earlier, simpler algorithm by Martin Hellman that used the inversion of hashes by looking up precomputed hash chains – **note!** not as in: h(h(h(password)))
  - Cryptoanalytic time-memory trade-off (fast attack but use more memory)
  - Reduction functions - only the first and last password of a chain stored in table
  - A hit means that chain contains hash - not 100% guarantee to crack password
- Ophcrack – Multi platform/core, special tools as live CD etc.
- Free Rainbow Tables – Windows (src), multi core support, slow updates
- RainbowCrack – Multi platform/core and GPU (CUDA) accelerated
- Cain – Winrtgen, can use other tables as well



Rainbow table example

http://en.wikipedia.org/wiki/Rainbow_table

- A simplified rainbow table with 3 reduction functions
- Chain length is usually up to around 3 – 4 thousand and number of rows is usually around 40 million when expanded fully - if needed
- Functions
  - H = hash function, R = reduction function

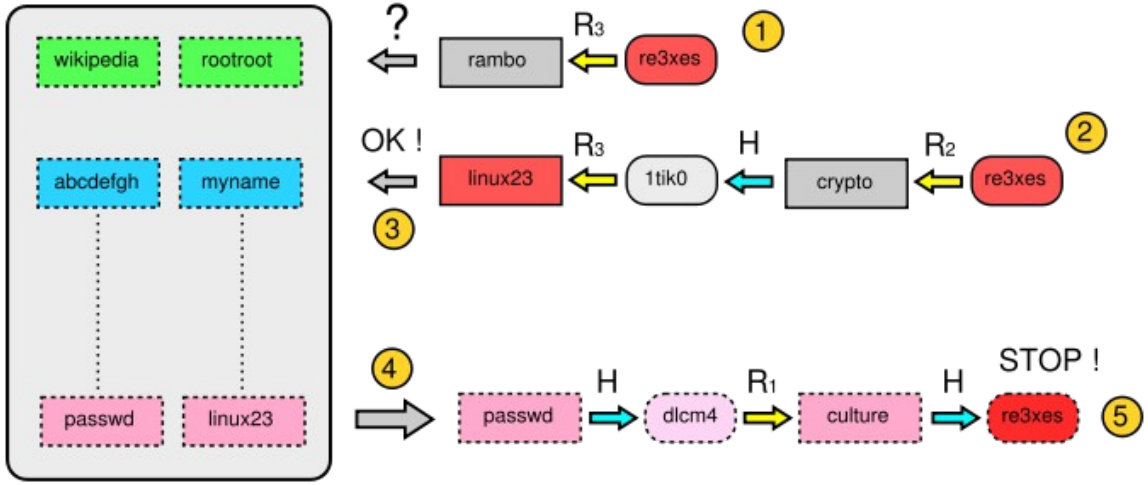


Rainbow table example cont.

http://en.wikipedia.org/wiki/Rainbow_table

We have a hash (re3xes) and we want to find the password that produced that hash

Table with only first and last password stored in chain for every row



Functions
H = hash
R = reduction

1. Starting from the hash ("re3xes"), one computes the last reduction used in the table and checks whether the password appears in the last column of the table (step 1).

2. If the test fails (rambo doesn't appear in the table), one computes a chain with the two last reductions (these two reductions are represented at step 2)

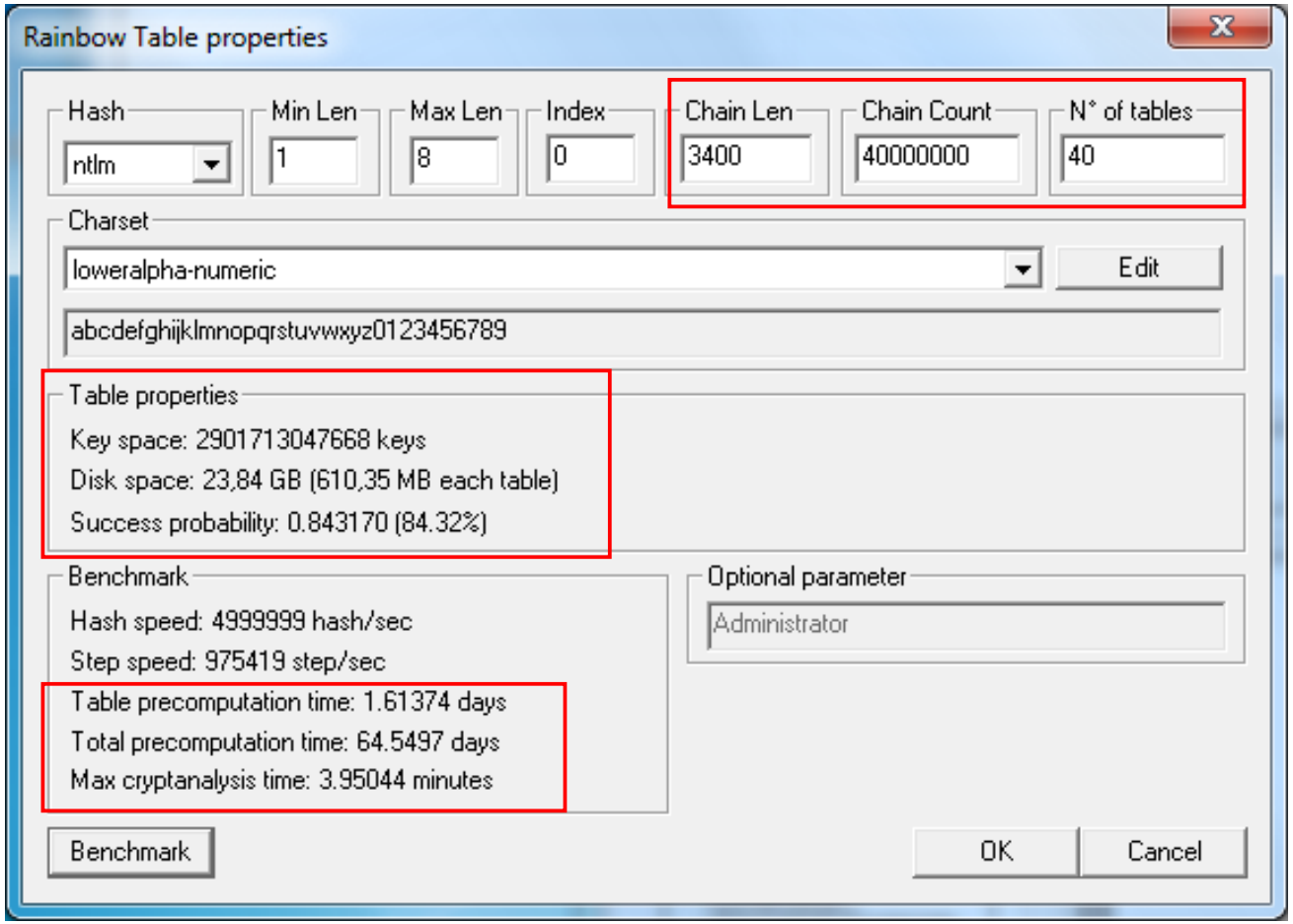
Note: If this new test fails again, one continues with 3 reductions, 4 reductions, etc. until the password is found. If no chain contains the password, then the attack has failed.

3. If this test (step1) is positive (as in step 3, linux23 appears at the end of the chain and in the table), the password is retrieved at the beginning of the chain that produces linux23. Here we find passwd at the beginning of the corresponding chain stored in the table.

4. At this point (step 4), one generates a chain and compares at each iteration the hash with the target hash. In this case the test is valid and we find the hash re3xes in the chain (step 5). The current password (culture) is the one that produced the whole chain : the attack was successful!

Cain Winrtgen v2.9

- Rainbow Table properties



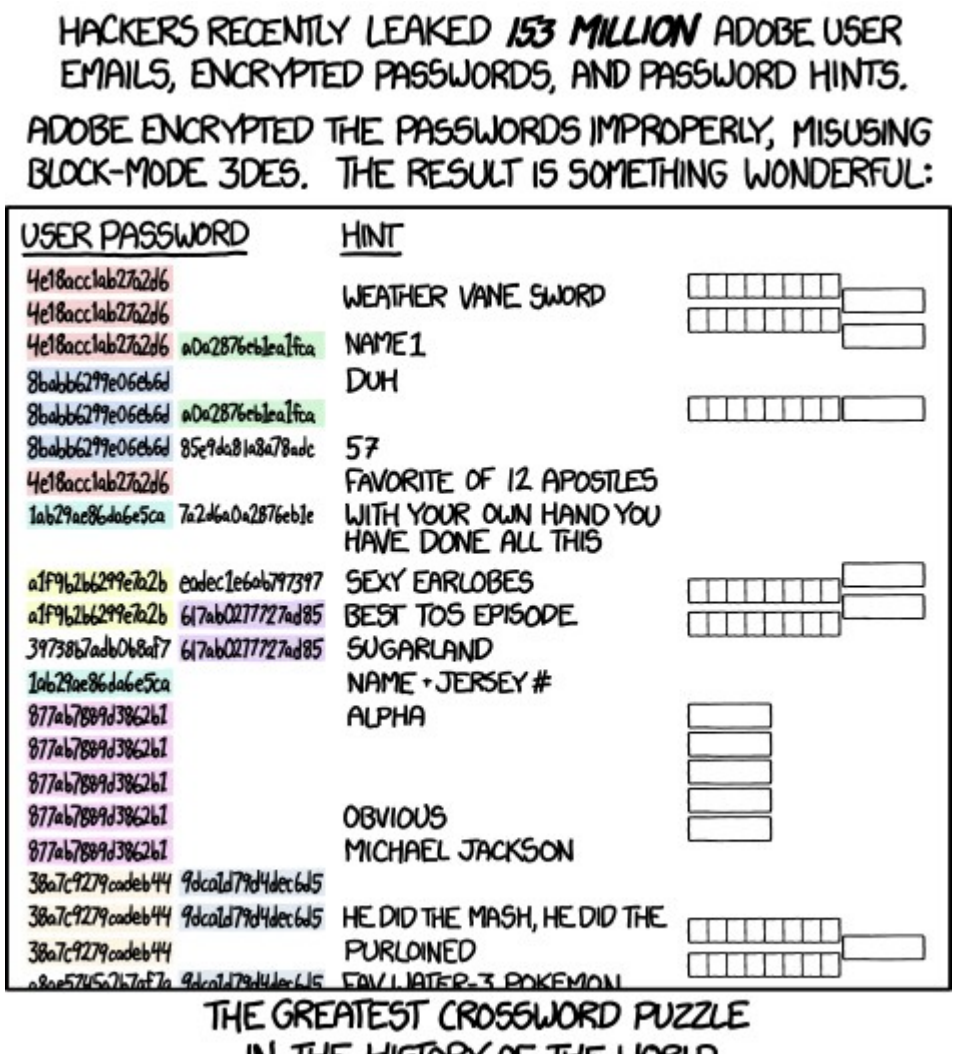
Password attack defense

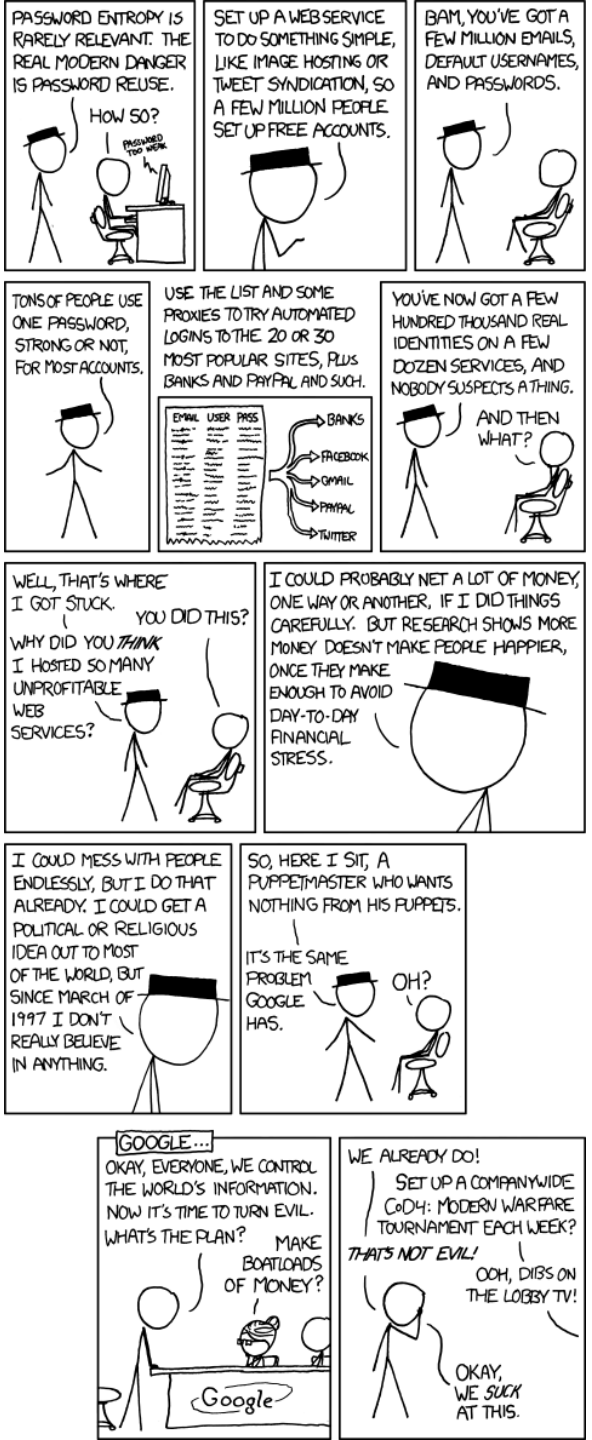
- Strong password policy
- User awareness
- Password filter
  - Force the use of strong passwords
- Do password-cracking tests
- Protect the hashed password files
- Get rid of LM hashes
  - HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa NoLMHash = 1
- Two-factor (or even three) authentication - strong authentication
  - http://en.wikipedia.org/wiki/Two-factor_authentication
  - Something you have (token), you know (passwd), you is or does (fingerprint)
  - SecurID server connected to AD etc.
  - Pin code (number) is generated which matches the RSA SecureID token
  - Token is syncronized with SecurID server
  - Login with both pin code and pass code (displayed in token for limited time)
  - http://en.wikipedia.org/wiki/SecurID

Password reuse

http://xkcd.com/792/

http://www.xkcd.com/1286/





Death of the web user/password?

http://en.wikipedia.org/wiki/OpenID

Authenticate with Google, FB, etc.

- OpenID is a decentralized single sign-on system
  - Authenticate once and gain access to the resources of multiple software systems
- Builds on digital identity from a OpenID provider
  - URL or XRI (eXtensible Resource Identifier)
  - Log in with for example: http://alice.myopenid.com
  - Web site will check with OpenID provider if valid (or my own domain which redirects)
- There are two modes in which the relying party can communicate with the identity provider:
  - checkid_immediate, which is machine-oriented and in which the relying party requests that the provider not interact with the user. All communication is relayed through the user's browser, but presumably without the user's knowledge;
  - checkid_setup, in which the user communicates with the provider server directly using the very same web browser used to access the relying party site.
- OpenID does not provide its own form of authentication, but if an identity provider uses strong authentication, OpenID can be used for secure transactions such as banking and e-commerce (a token or other hardware is needed)
- More reading
  - http://stacktrace.se/2007/10/04/openid-en-introduktion/
  - http://www.intertwingly.net/blog/2007/01/03/OpenID-for-non-SuperUsers
  - http://www.idg.se/2.1085/1.338358/google-oppnar-for-openid (2010-09-08)

yubico
Trust the Net.

Death of the web user/password?

YubiKey Core features

- two-factor authentication with one-time passwords
- Works instantly, no need to re-type pass codes from a device
- Works on Windows, Mac, Linux, iPad, Firefox, Chrome, etc
- Identified as a USB-keyboard, no client software or drivers needed
- Minimized size; 2 mm thin, 3 grams
- Practically indestructible; waterproof, crush safe, no battery
- Integration within minutes with free and open source server software
- Two slots for multiple configurations: OATH*, Challenge-Response etc.
- Also available with NFC (NEO) and minimized form factor (Nano)
- Lowest total cost of ownership for strong two-factor authentication

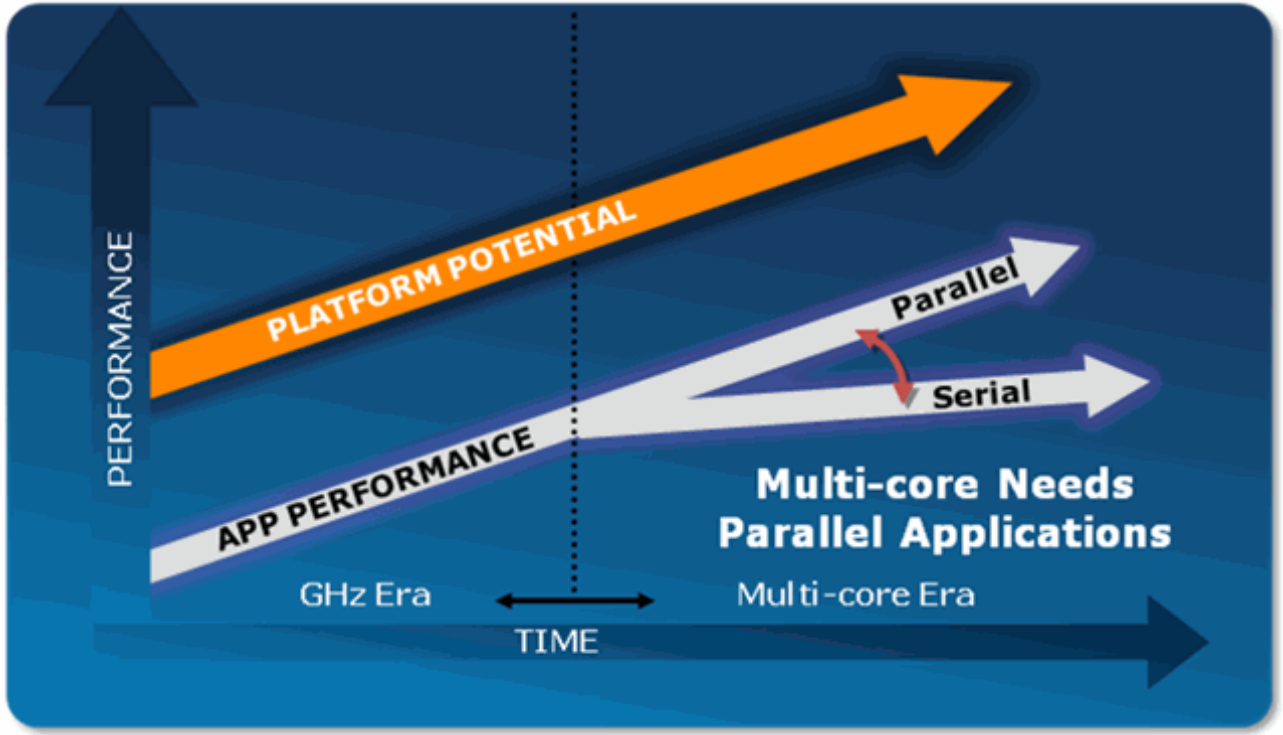
* OATH (Open Authentication) open standard from VeriSign

How it works

With a simple touch of the gold disc, the YubiKey sends a One Time Password (OTP) as if it was typed in from a keyboard. The unique passcode is verified by a YubiKey compliant application.

Parallel computing 1

- Good intro to parallel computing
  - https://computing.llnl.gov/tutorials/parallel_comp/
- FASTRA II = **12 TFLOPS**, 13x GPU (6 NVIDIA GTX295 dual-GPU cards and one GTX275 single-GPU card) – GT200, 6000 € (2009)
  - http://en.wikipedia.org/wiki/Fastra_II
- **46 TFLOP** - 2008-06, 14 mil. SEK - http://www.hpc2n.umu.se/resources/akka/

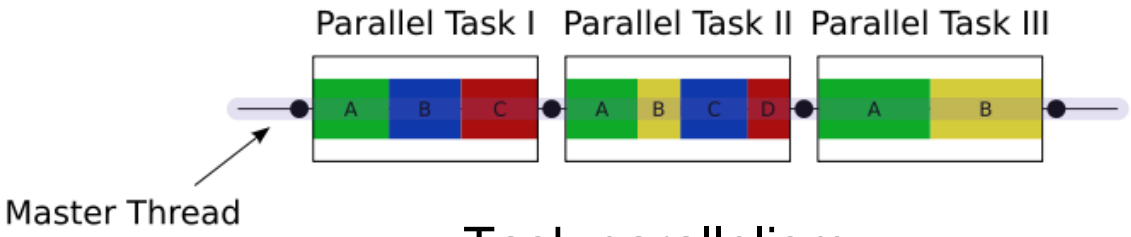


Parallelism

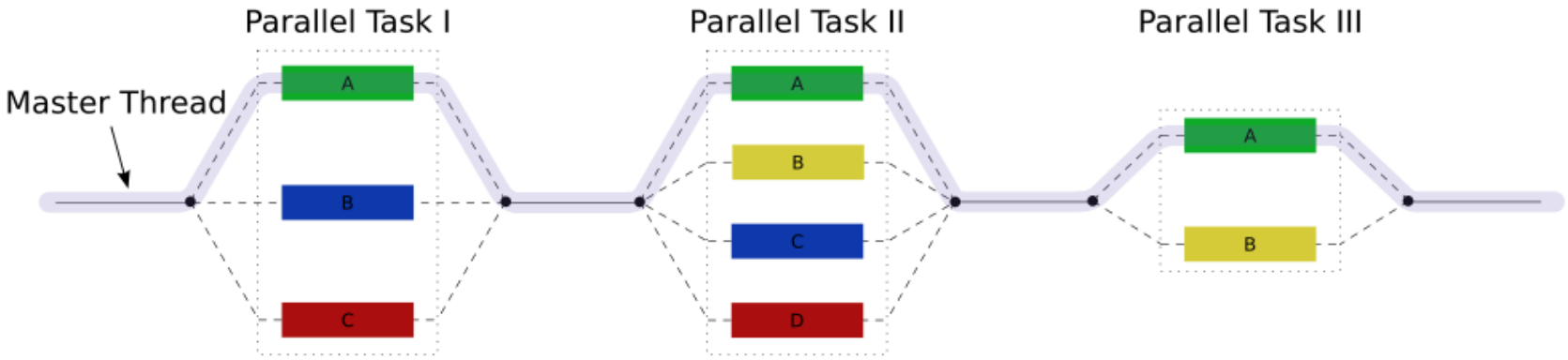
- Data parallelism
  - http://en.wikipedia.org/wiki/Data_parallelism
- Task parallelism
  - http://en.wikipedia.org/wiki/Task_parallelism

Software threads vs. hardware threads

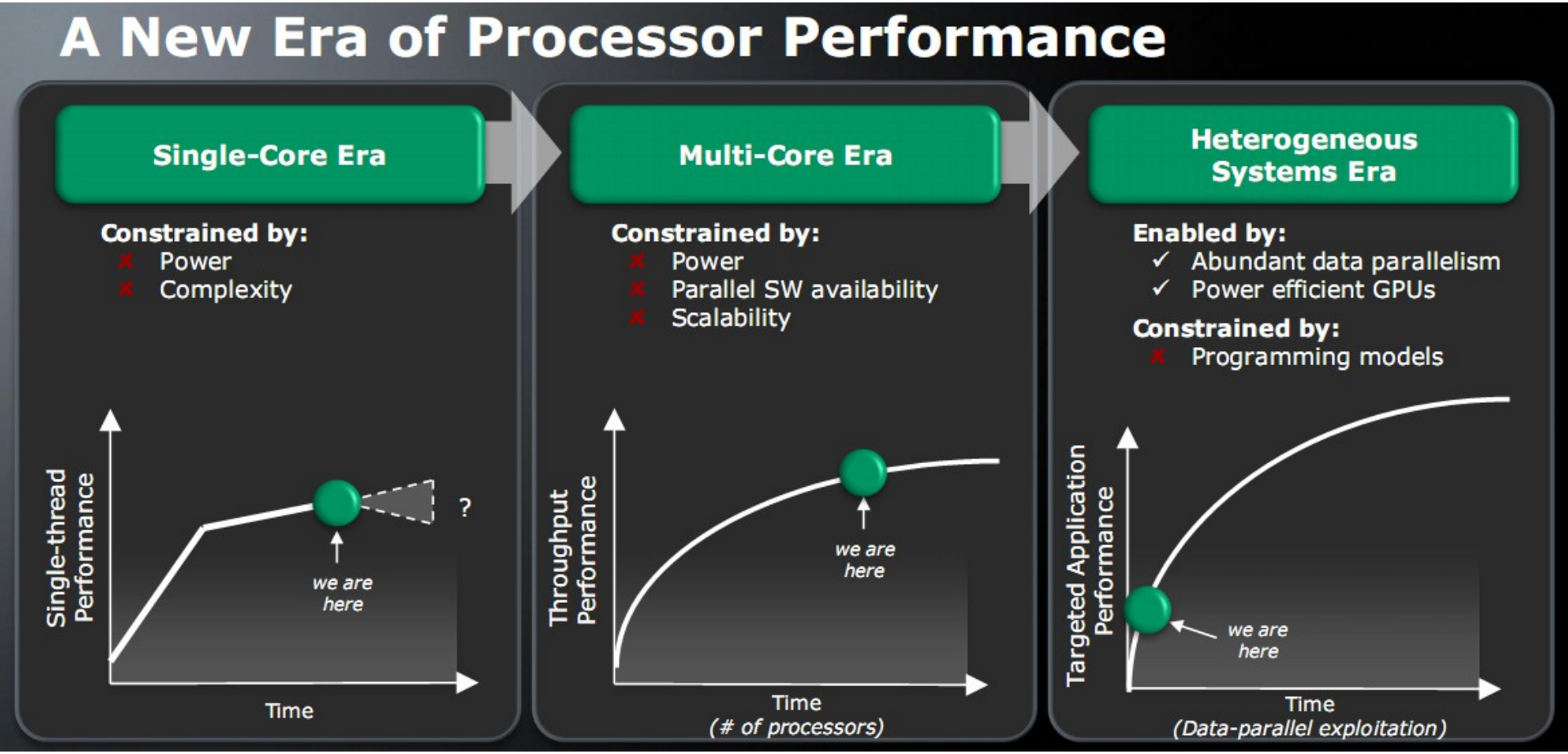
Data parallelism



Task parallelism



Parallel computing 2

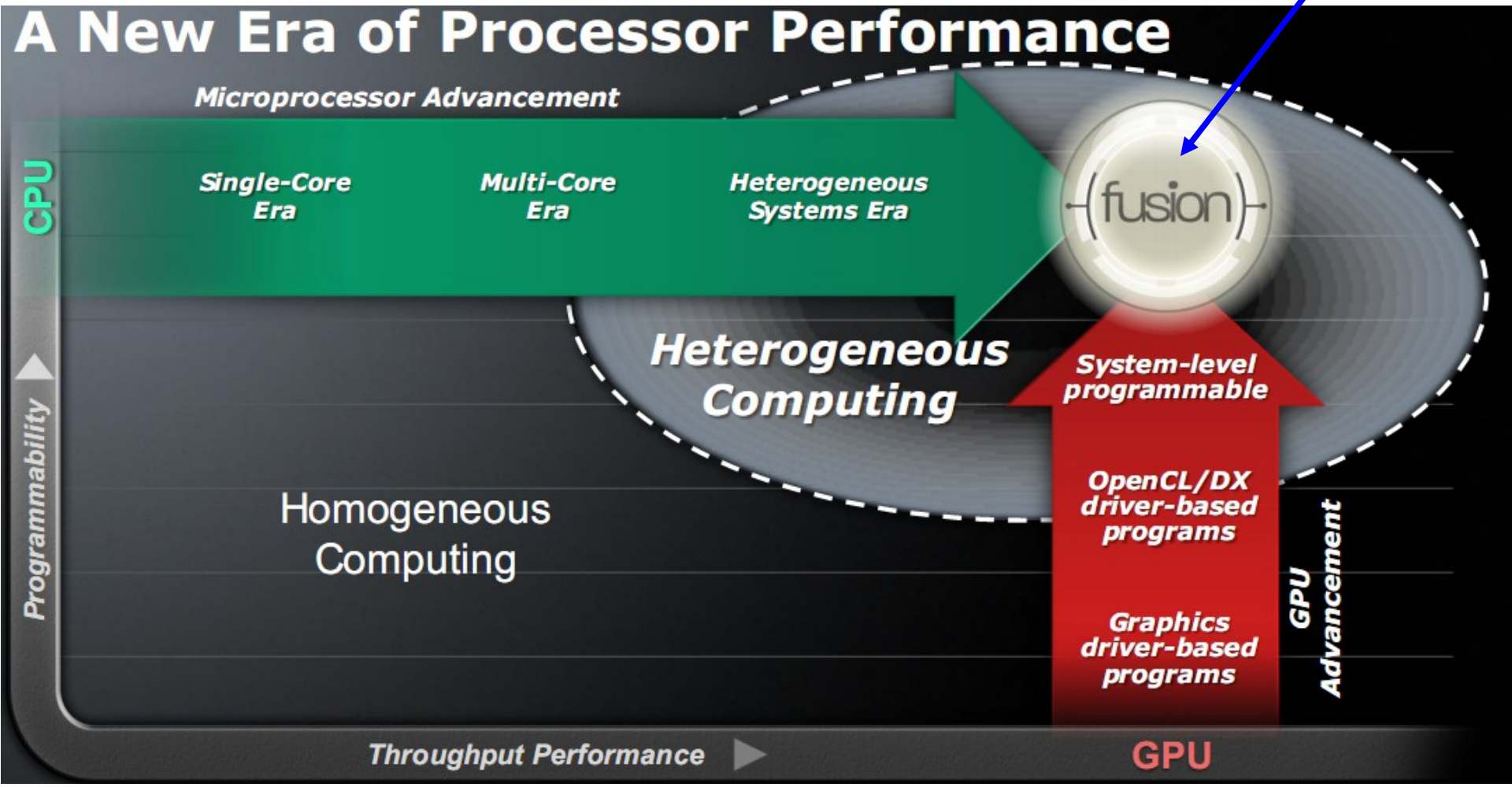


Just now 6, 8, 10 and 12 ”many core” CPUs can be bought

Parallel computing 3

AMD Llano
X4 CPU, 400 SP

Radeon HD 6450 in room 348 have 160 SP



Fusion → APU (Accelerated Processing Unit), CPU + GPU

Tools for multicore development

- Writing native threads is now considered a relic from the early days of parallelism
- Over 40% of all programmers now work on data parallel applications according to ZDNet (2011)
  - A popular solution then were Intel Threading Building Blocks (& OpenMP)
- Most of the new libraries and extensions builds on the work-stealing algorithm pioneered by the Cilk project (now Cilk++)
  - http://en.wikipedia.org/wiki/Cilk - http://www.cilk.com/
- Work-stealing algorithm in short
  - Each worker thread maintains runnable tasks in its own scheduling queue
  - When a worker thread has no local tasks to run, it attempts to take("steal") a task from another randomly chosen worker thread, using FIFO (oldest first) rule
  - When a worker thread encounters a join operation, it processes other tasks, if available, until the target task is noticed to have completed
  - When a worker thread has no work and fails to steal any from others, it backs off (via yield, sleep, and/or priority adjustment)

OpenMP (Open Multi-Processing) API

- Supports multi-platform shared memory multiprocessing programming in C/C++/Fortran with both task and data parallelism via preprocessor pragma directives
  - Threads usually joins (barrier) with the master thread which got id=0
  - Makes it simple to add multi-core support into existing programs
  - MSDN Magazine article: ”Reap the Benefits of Multithreading without All the Work”

```
// compile with: /openmp to enable OpenMP 2.0 language extensions
// http://msdn.microsoft.com/en-us/library/tt15eb9t.aspx
#include <stdio.h>
#include <omp.h>
int main()
{
   printf("OMP Max Threads: %d\n", omp_get_max_threads());
   printf("OMP Num Cores: %d\n", omp_get_num_procs());
  #pragma omp parallel sections num_threads(2)
  {
      #pragma omp section
      printf_s("Hello from thread %d\n", omp_get_thread_num());
      #pragma omp section
      printf_s("Hello from thread %d\n", omp_get_thread_num());
  }
    // Implicit barrier syncronize the threads
}
```

```
// output
OMP Max Threads: 2
OMP Num Cores: 2
Hello from thread 0
Hello from thread 1
```

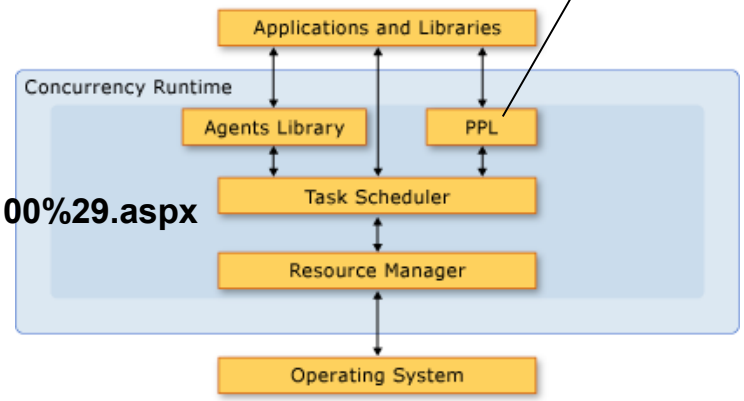
http://openmp.org

http://en.wikipedia.org/wiki/OpenMP

MS VS 2010 and Parallel Computing

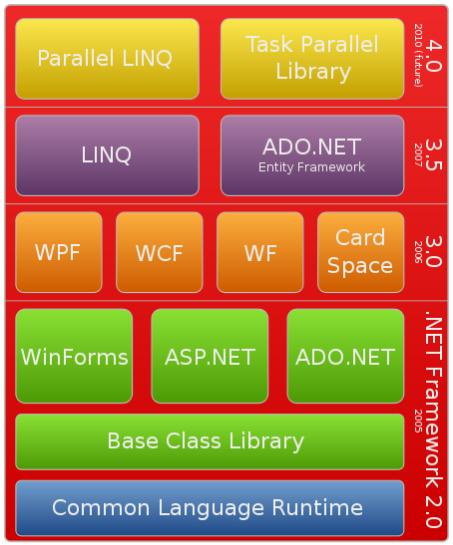
- **.NET4.0 have Parallel Extensions as a key part of the framework**
  - **Use of a ThreadPool as in OpenMP eliminate costly thread creation**
- **Parallel Programming in Native Code (with PPL)**
  - **http://blogs.msdn.com/nativeconcurrency/**
- **Concurrency Runtime**
  - **http://msdn.microsoft.com/en-us/library/dd504870%28VS.100%29.aspx**
- **Parallel Programming with .NET**
  - **http://blogs.msdn.com/pfxteam/**
- **Parallel Computing Developer Center**
  - **http://msdn.microsoft.com/en-us/concurrency/default.aspx**

Parallel Patterns Library, C++ templates



MSDN sites got all info!

```
using System.Threading; // C# - Parallel Ext. work in VB and F# as well
// Equivalent to: for(int i = 0; i < 10; i++)
Parallel.For(0, 10, int i => {
   Console.WriteLine(i);
});   // starting 3 parallel operations
Parallel.Invoke( () => MethodA(), () => MethodB(), () => MethodC() );
// Sequential version of foreach
foreach(var item in sourceCollection) { Process(item); }
// Parallel equivalent of foreach
Parallel.ForEach(sourceCollection, item => Process(item));
```



The .NET Framework Stack

Parallel Java

- Parallel Java has arrived with the Java JDK7 (1.7) release
  - Java has coarse-grained concurrency since JDK5 and threads since start
  - Fork/join framework (fine-grained parallelism)
    - java.util.concurrent.ForkJoin* – JSR166y/z (updated revisions)
  - http://www.javac.info/jsr166z/ - http://gee.cs.oswego.edu/dl/jsr166/dist/
- Base class examples
  - RecursiveAction
  - RecursiveTask
  - CyclicAction
  - Parallel***Array
  - AsyncAction

```
/* Creates a ForkJoinPool with a pool size equal to the number
of processors available on the system and using the default
ForkJoinWorkerThreadFactory */

ForkJoinPool fjpool = new ForkJoinPool();

// Creates a new parallel array with given array and executor

ParallelLongArray lparray =
ParallelLongArray.createUsingHandoff(array, fjpool);

long max = lparray.max();
```

http://www.oracle.com/technetwork/java/7-138633.html

http://developers.sun.com/learning/javaoneonline/2008/pdf/TS-5515.pdf

http://www.ddj.com/go-parallel/blog/archives/2009/04/java_7_will_evo.html

ShmooCon 2010

GPU vs. CPU Supercomputing Security Shootout

Collin Brack

You have the fastest Intel/AMD processor in a 500 mile radius thanks to your custom built quad-core, liquid nitrogen cooled, overclocked 5.0Ghz CPU monster. Prepare to be summarily beat down, computationally speaking, by the kid next door who just bought the latest Nvidia GPU to play WOW at 80fps. Video cards, fueled by the gaming industry, have leap-frogged (pun intended) the processing power of the general purpose CPU for certain computational tasks. The rise of the multi-processor based general purpose GPU (GPGPU) platform is taking academia by storm due to its low costs and low barrier to entry into modern day supercomputing. The security community has already embraced the GPU for heavy lifting as have other fields especially when coupled with the sleek marketing efforts by Nvidia and their CUDA development environment, and competing GPU computing platforms from ATI and OpenCL. This 20 minutes session will chronicle the rise of the GPU in high performance computing and will highlight GPU vs. CPU benchmarks of well known security tools including: aircrack (10x speed-up), Pyrit (8x), CUDA Multiforcer, BarsWF MD5 cracker (3x), RainbowCrack multi-GPU CUDA version, and more. Finally, links and tips regarding implementing CUDA in Back|Track 4 are shared.

Collin Brack is a healthcare informatics and medical imaging consultant with experience in computational clusters. He works in academia where he focuses on high performance computing with medical physics researchers. His latest cluster is based on high-end graphics processors to achieve performance gains previously only available to multi-million dollar big iron. He has published and presented on the topics of system design, grid computing, and disaster recovery.

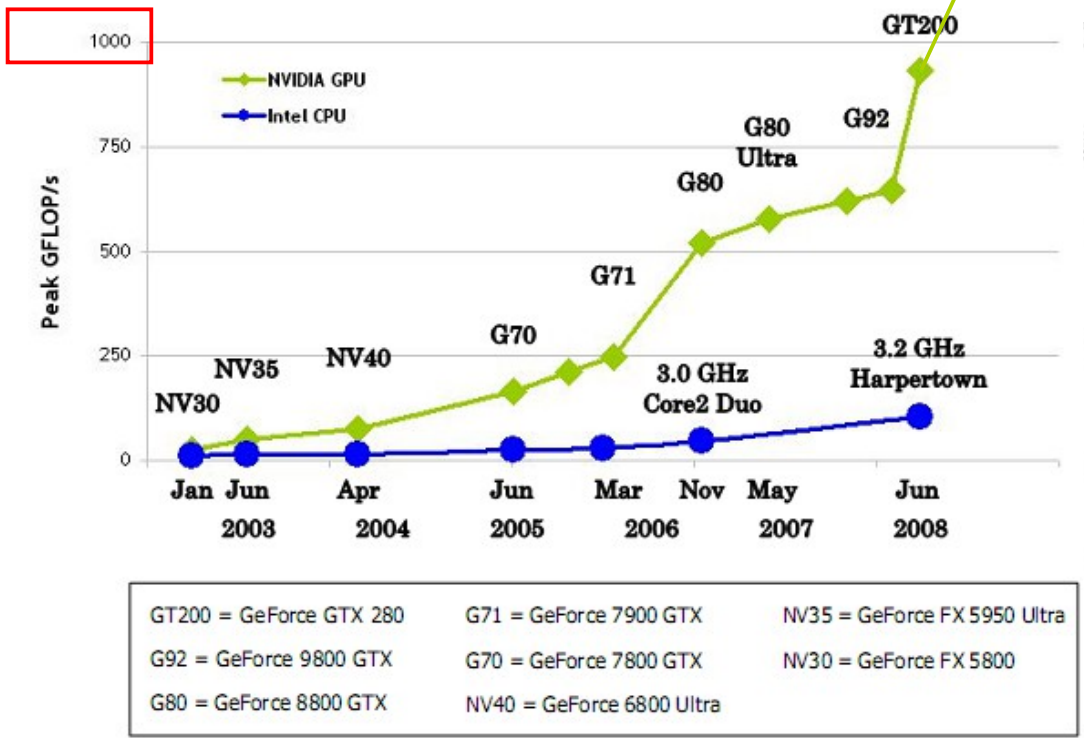
http://www.shmoocon.org

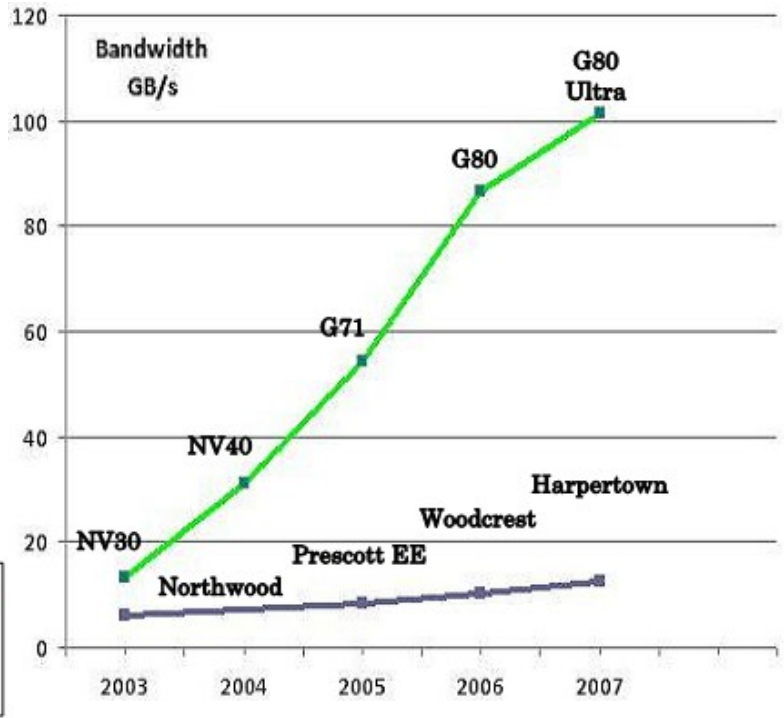
GPGPU

We are way off the slide now!

General-Purpose computation on Graphics Processing Units

- The GPU is a **massively parallel** device – **cores / shaders**
  - ATI 5970 got 1600x2 SP (Stream Processors) - 2320x2 GFLOPs single precision
  - A 2.66 GHz Intel Core 2 duo can perform about 25 GFLOPs single precision

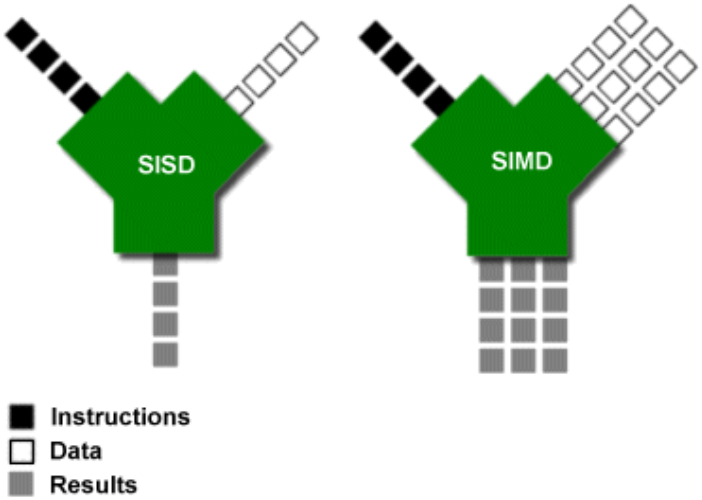




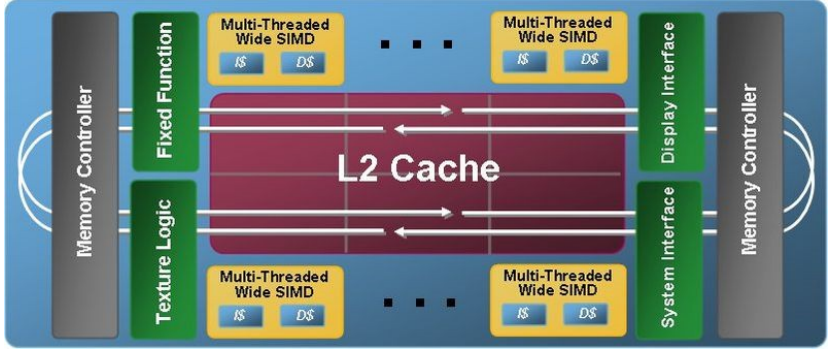
NVIDIA Tesla/Fermi - 512x4 SM(Stream Multiprocessors) - 1200x4 GFLOPs single precision

CPU/GPU architecture

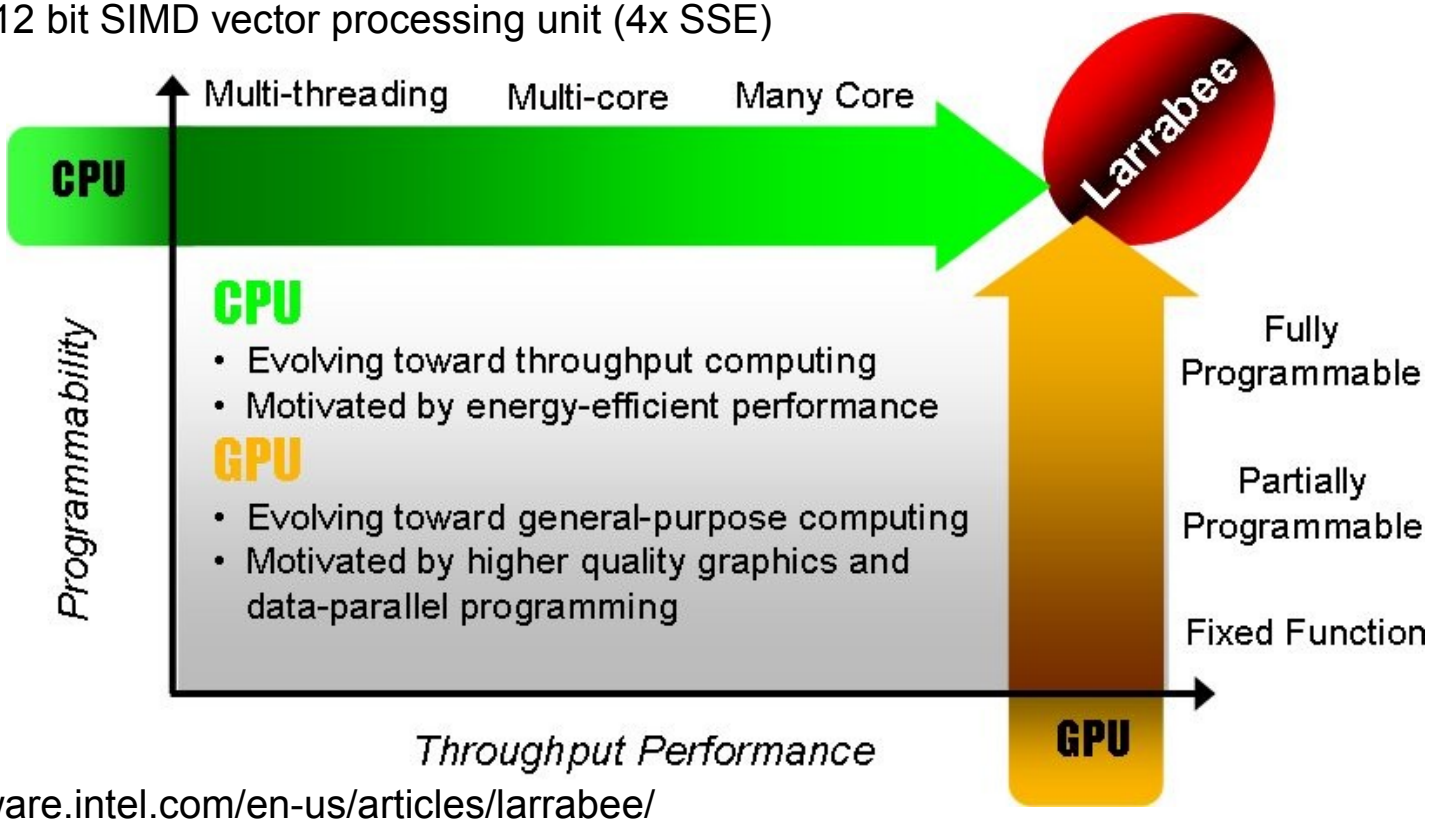
- Parallel processor architecture
  - Instruction Level Parallelism (ILP) http://en.wikipedia.org/wiki/Instruction_level_parallelism
    - Scalar vs. Superscalar vs. Very Long Instruction Word (VLIW)
    - Instruction pipelining, out-of-order execution, branch prediction, etc.
  - Streaming SIMD Extensions (SSE*)
- Flynn's taxanomy
  - SISD vs. SIMD vs. MISD vs. MIMD
  - http://en.wikipedia.org/wiki/Flynn%27s_taxonomy
- NVIDIA – scalar SIMT (Single Instruction Multiple Thread)
  - Brute force approach, simpler compilers, stabler performance
- AMD/ATI - VLIW MIMD (Multiple Instruction Multiple Data)
  - More advanced/efficient, needs advanced compilers, fall back to SIMD



Intel Larrabee



- Differences with current GPUs
  - Very little specialized graphics hardware
  - Use of x86 instruction set, cache coherency across all its cores
- Differences with current CPUs
  - Pentium design with updated features as x64 etc. using at least 32 cores
  - 512 bit SIMD vector processing unit (4x SSE)

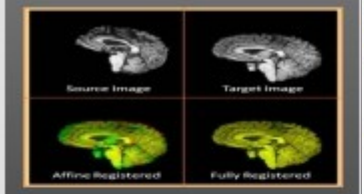


http://software.intel.com/en-us/articles/larrabee/

NVIDIA CUDA Zone

LATEST CUDA NEWS Sony Pictures Imageworks Creates Tornado Out Of Spaghetti Sauce With The Help Of NVIDIA Technology

GPU	G80	GT200	Fermi
Transistors	681 million	1.4 billion	3.0 billion
CUDA Cores	128	240	512
Double Precision Floating Point Capability	None	30 FMA ops / clock	256 FMA ops /clock
Single Precision Floating Point Capability	128 MAD ops/clock	240 MAD ops / clock	512 FMA ops /clock
Warp schedulers (per SM)	1	1	2
Special Function Units (SFUs) / SM	2	2	4
Shared Memory (per SM)	16 KB	16 KB	Configurable 48 KB or 16 KB
L1 Cache (per SM)	None	None	Configurable 16 KB or 48 KB
L2 Cache (per SM)	None	None	768 KB
ECC Memory Support	No	No	Yes
Concurrent Kernels	No	No	Up to 16
Load/Store Address Width	32-bit	32-bit	64-bit



AIRWC

100 x

Interactive Visualization of Billion Point Cosmological Simulations

9 x

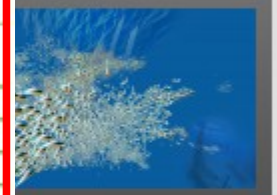
15 x

the GPU

12 x

Carlo eXtreme (MCX)

300 x

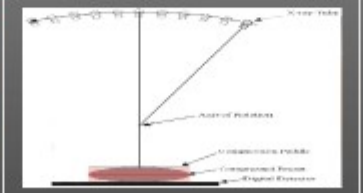


ient GPU implementation for large scaleindividual-based

Task and Data Parallel Framework for GPU Computing

Fast pattern classification of ventricular arrhythmias using graphics processing units

53 x



Digital Breast Tomosynthesis Reconstruction

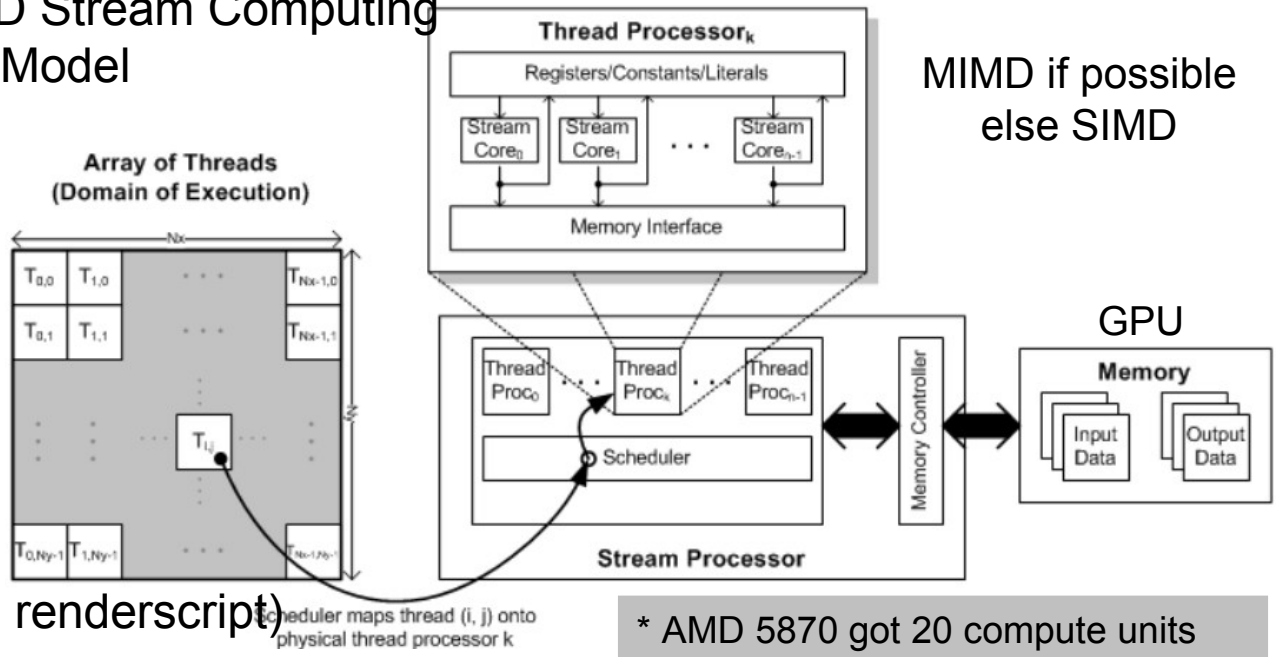
85 x

* Fermi contains 512 CUDA-cores/SM:s (16 SIMT units with 32 cores each).
* 1 warp = 32 threads which use a minimum of 2 clock cycles.
* Each SM can handle a maximum of 48 warps and schedule them freely.
* Track max 512 * 48 = 24576 threads/GPU!

Fermi: http://www.youtube.com/watch?v=fYuH2Kl_b98

GPGPU Frameworks - with regard to simplicity

Simplified AMD Stream Computing Programming Model



MIMD if possible else SIMD

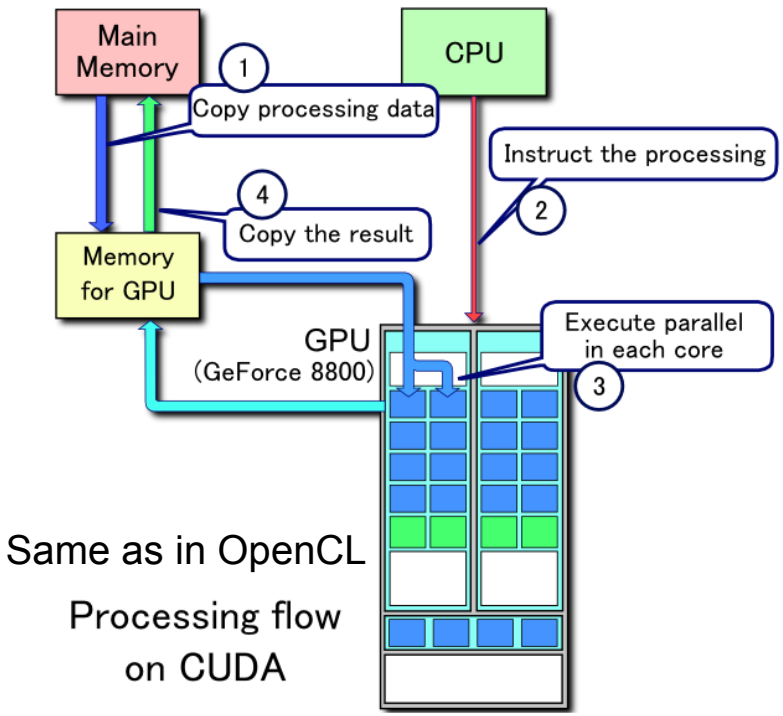
- Legacy
  - OpenGL, Direct3D
- Early high level
  - Sh/Rapidmind (Intel Ct now)
  - BrookGPU (free)
  - PeakStream (Google renderscript)
  - NVIDIA CUDA (NVCC) - C/C++, Nexus VS plugin
  - AMD CAL (CTM, Brook+) - C/C++
- Now
  - OpenCL (Open Computing Language) - C (C99) with extensions
    - Platform and hardware independent GP parallel programming
  - Microsoft DirectCompute, part of DirectX 11 (10.x will run to)

* AMD 5870 got 20 compute units with 16 TPs each, which have a execute unit that can execute 5 different instructions in 1 cycle in a packed VLIW = 1600 SPs
* 1 wavefront=64 threads @ 4 cycles

OpenCL programming model

- Develop your program with normal source code
- For the parts where you want GPU support you create *_kernel.cl files with OpenCL C-source code
  - The OpenCL code can be embedded in program image as well
- When the program executes the OpenCL driver loads and compile the .cl sources on-demand
  - Binaries can be cached or written to disk avoiding lengthy loads
  - OpenCL scale apps automatically, 1-n CPUs/GPUs etc.
- AMD, nVidia and Intel support OpenCL via the graphics driver
  - Since 2009 - 2011

http://www.youtube.com/watch?v=7PAiCinmP9Y



OpenCL ”Hello World” (AMD/Apple)

Runs on: high-performance compute servers, desktop computer systems and handheld devices using a diverse mix of multi-core CPUs, GPUs, Cell-type architectures and other parallel processors such as DSPs.

https://developer.apple.com/search/?q=opencl

http://developer.amd.com/gpu/ATIStreamSDK/pages/TutorialOpenCL.aspx

```
/* Simple compute kernel which computes the
   square of an input array */
__kernel void square(
    __global float* input,
    __global float* output,
    const unsigned int count)
{/* Returns the unique global work-item ID
   value for dimension identified by dimindx */
    int i = get_global_id(0);
    if(i < count)
        output[i] = input[i] * input[i];
}
```

// complement code to the compute kernel

1. Get and select the devices to execute on
2. Create and open an OpenCL context
3. Create a command queue to accept the execution and memory requests
4. Allocate OpenCL memory objects to hold the inputs and outputs for the compute kernel
5. Online LLVM (Low Level Virtual Machine) compile and build the compute kernel code
6. Set up the arguments and execution domain
7. Kick off compute kernel execution
8. Collect the results
9. Shutdown and cleanup

// Note! 6, 7 and 8 may need to iterate if job is big – i.e. have a chance to stop the job...

PyOpenCL

Python bindings/wrappers

Example with kernel right →

Benchmark test with CPU ⤵

C:\pyopencl-0.91.4\examples>python benchmark-all.py

Execution time of test without OpenCL: 23.0160000324 s

==========

Platform name: ATI Stream

Platform profile: FULL_PROFILE

Platform vendor: Advanced Micro Devices, Inc.

Platform version: OpenCL 1.0 ATI-Stream-v2.0.1

Device name: AMD Turion(tm) 64 X2 Mobile Technology TL-60

Device type: CPU

Device memory: 1024 MB

Device max clock speed: 1995 MHz

Device compute units: 2

Execution time of test: 0.00329344 s

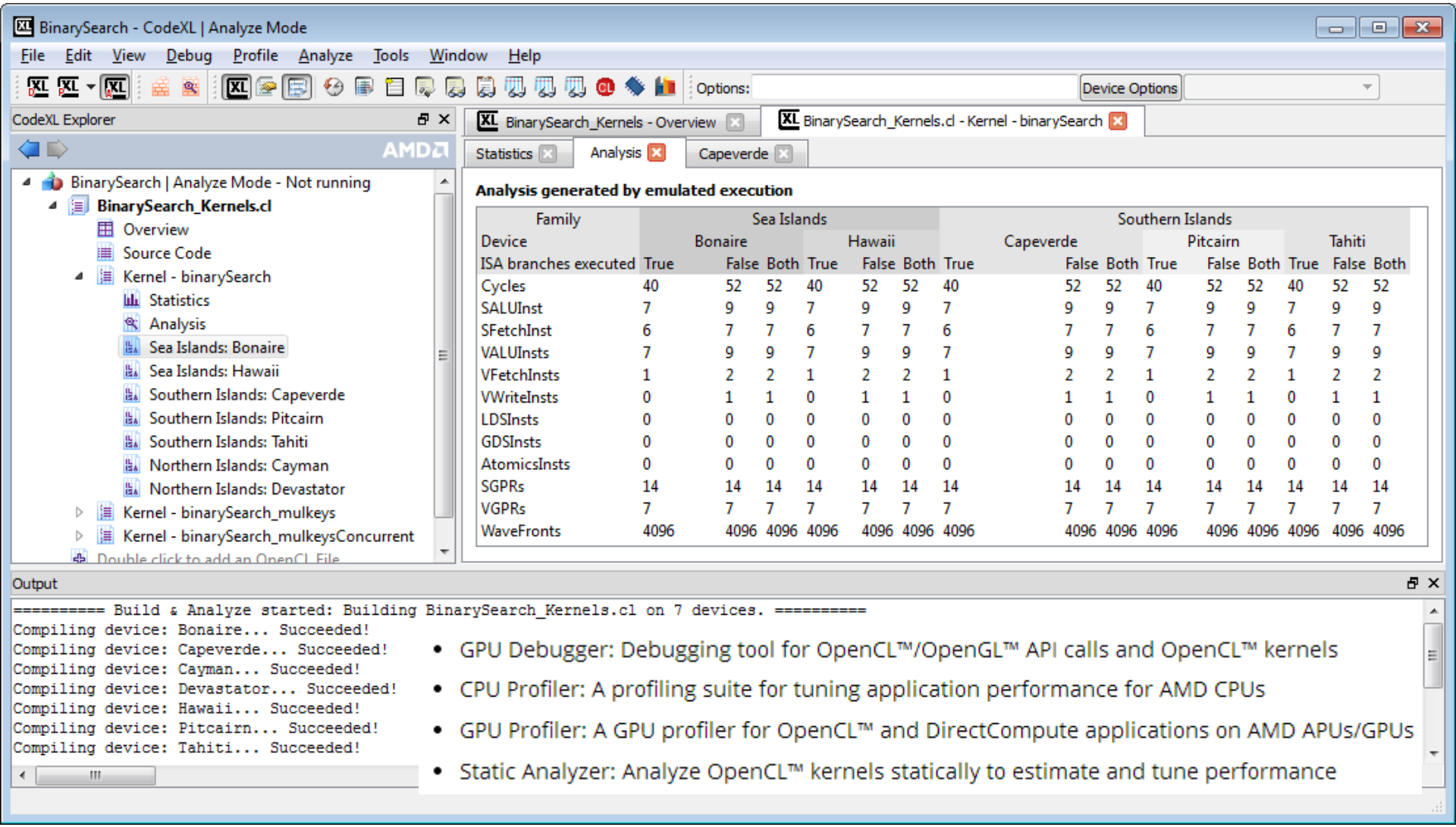
Results OK

http://mathema.tician.de/software/pyopencl

```
# Hello World in Python with OpenCL
import numpy
import pyopencl as cl

hello = "Hello World"
a = numpy.empty((len(hello),), dtype=numpy.byte)

for platform in cl.get_platforms():
    for device in platform.get_devices():
        print "Device name: " + device.name

ctx = cl.Context([device])
queue = cl.CommandQueue(ctx)
mf = cl.mem_flags
dest_buf = cl.Buffer(ctx, mf.WRITE_ONLY, a.nbytes)

prg = cl.Program(ctx, """
#pragma OPENCL EXTENSION cl_khr_byte_addressable_store : enable
__constant char hw[] = "Hello World";
__kernel void hello(__global char * out)
{
    size_t tid = get_global_id(0);
    out[tid] = hw[tid];
}
""")

try:
    prg.build()
except:
    print "Error:"
    print prg.get_build_info(ctx.devices[0],
        cl.program_build_info.LOG)
    raise

prg.hello(queue, a.shape, dest_buf)
hw = numpy.empty_like(a)
cl.enqueue_read_buffer(queue, dest_buf, hw).wait()
world =""
for res in hw:
    world = world + chr(res)
print world
```

AMD CodeXL

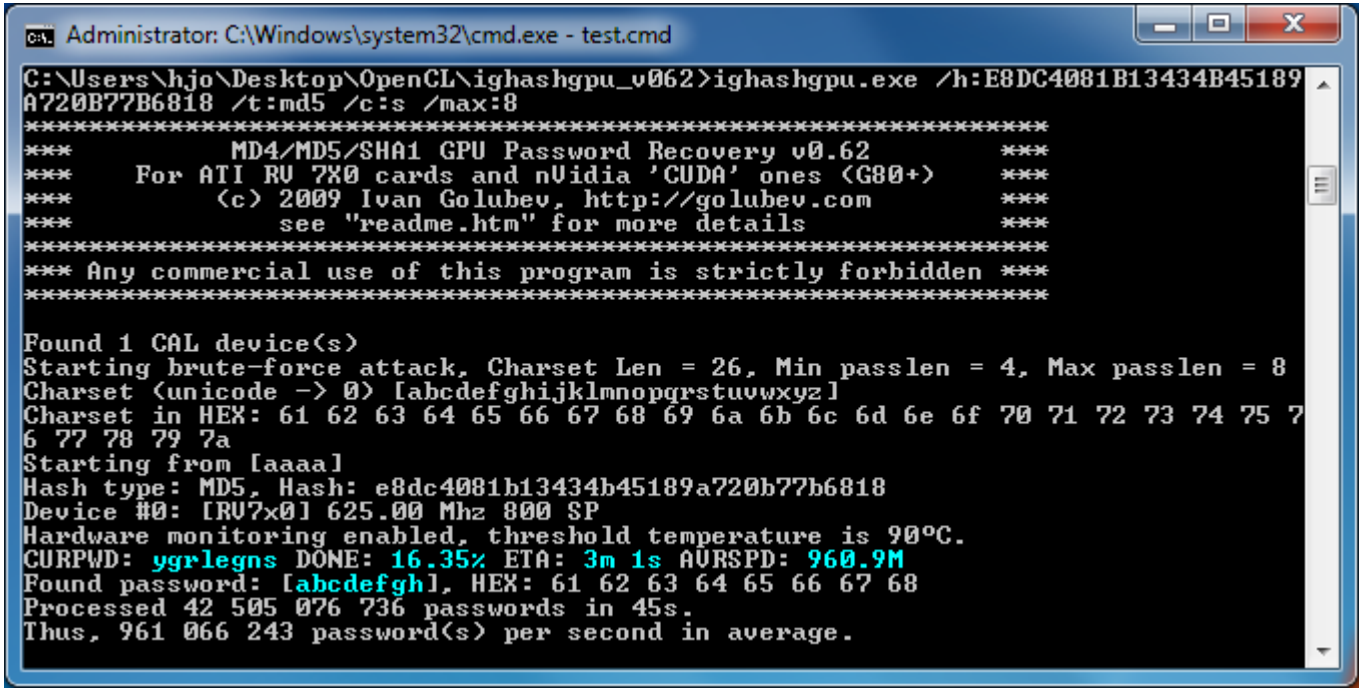
http://developer.amd.com/community/blog/2013/11/08/codexl-1-3-released/



GPGPU testing 1

- IGHASHGPU (Brook+/CUDA), recover/crack SHA1, MD5 & MD4 hashes
  - Supports salted hashes, NTLM, MySQL*, Oracle 11g, ..., etc.
  - Plain MD5, 8 chars, lowercase
  - Windows 7 x64, AMD Phenom x4 @ 2.2GHz
  - ATI 4850 - 800SP, Catalyst 9.12, Stream SDK v2.0
  - Count down time (ETA) started at almost 4 minutes

http://golubev.com
Intresting discussion
RAR GPU as well

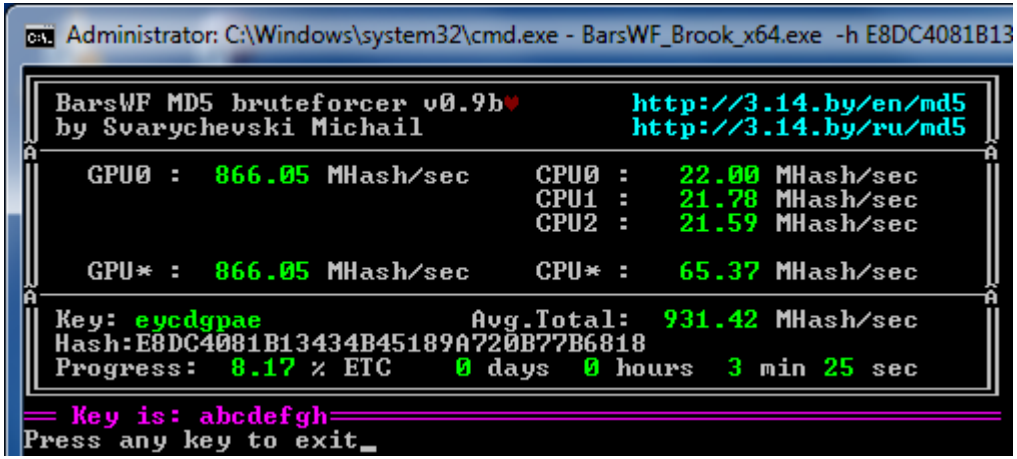


BF NT hash crack
7 char pass a-z,0-9
ETA:
Cain, 3.5h
IGHASHGPU, 1 min

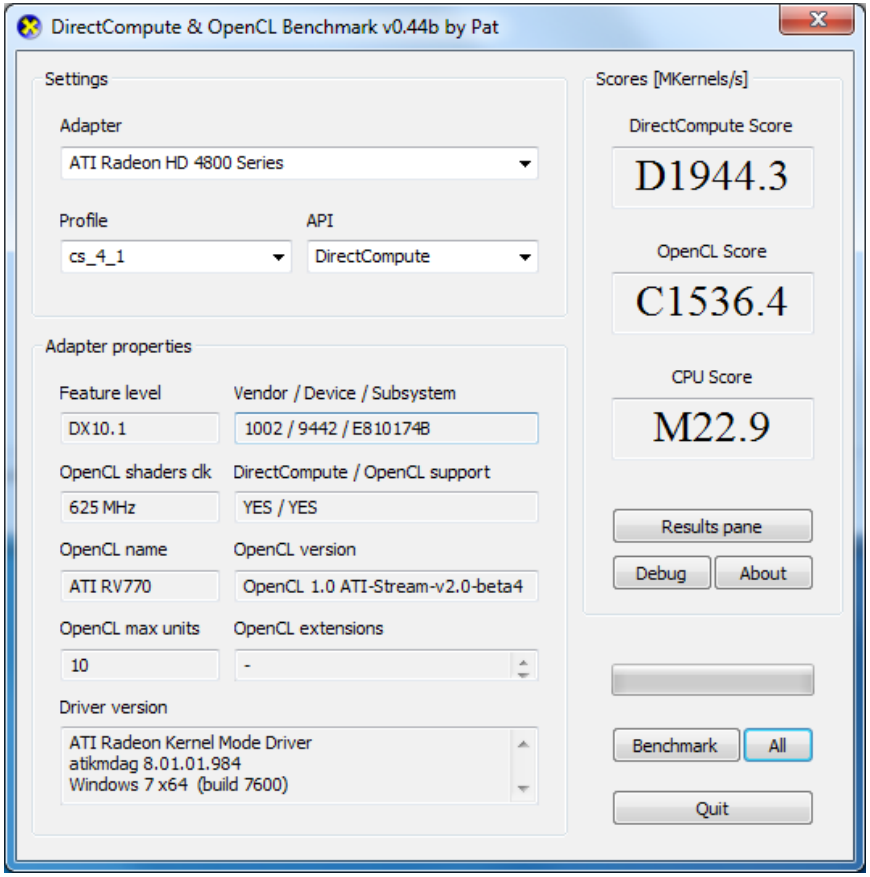
Current GPU generation is more than 10 times faster!

GPGPU testing 2

- BarsWF (Brook+/CUDA/SSE), recover/crack MD5
  - Same settings as for IGHASHGPU



DirectCompute Benchmark v0.44b



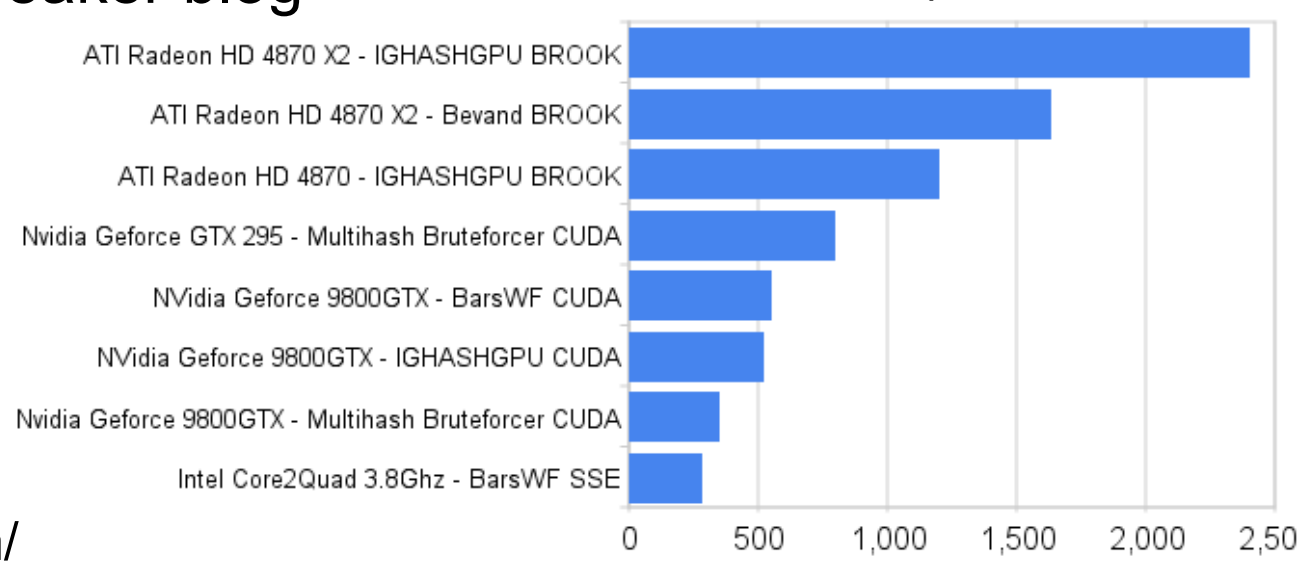
- Others
- OCLCrack (OpenCL) with source
- http://sghctoma.extra.hu/index.php?p=entry&id=11
- Multihash Bruteforcer CUDA
- http://www.cryptohaze.com/
- Extreme GPU Bruteforcer CUDA
- http://www.insidepro.com/eng/egb.shtml

MD5 Software Benchmark

- From the Code Breaker blog

Marc Bevand have some intersting articles about breaking crypto using ATI CAL IL and Sony PS3

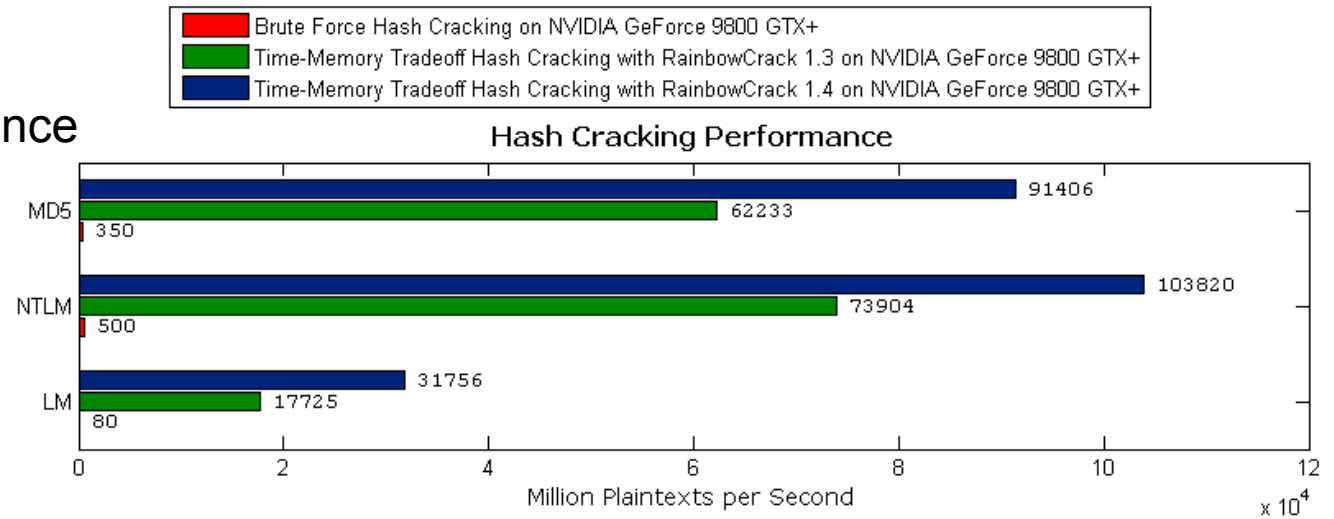
http://www.zorinaq.com/



RainbowCrack performance

Around 200 times faster than brute force with GPU

Run tables with CPU?



GPU limitations

- System to GPU bandwith is limited
  - Around 1 byte per clock tick
  - A CPU will be able to perform **A LOT** of instructions during the copy of input and output data to/from the compute kernel
- Error control handling and debugging can be difficult to perform
- GPUs like their data arranged in specific ways/formats etc.
- Not all problems/algorithms are optimal to run on a GPU
- Amdahls law
  - How much in a program can be parallelized?

http://en.wikipedia.org/wiki/Amdahl%27s_law

$$\text{speedup} = \frac{1}{\frac{P}{N} + S}$$

where P = parallel fraction, N = number of processors and S = serial fraction

	speedup		
N	P = .50	P = .90	P = .99
10	1.82	5.26	9.17
100	1.98	9.17	50.25
1000	1.99	9.91	90.99
10000	1.99	9.91	99.02
100000	1.99	9.99	99.90

GPGPU references/resources

- Blogs
  - Speed Junkie - http://gpgpu-computing.blogspot.com/
  - Code Breaker - http://jchblue.blogspot.com/
  - GPU computing - http://oscarbg.blogspot.com/
- OpenCL tutorials etc.
  - http://www.macresearch.org/opencl
  - http://gpgpu.org/
  - http://en.wikipedia.org/wiki/DirectCompute
  - http://developer.amd.com
  - http://developer.nvidia.com
  - http://www.khronos.org/developers/
  - http://www.geeks3d.com/
- C# bindings/wrappers - http://sourceforge.net/projects/cloo/
- Java bindings/wrappers - http://www.jocl.org/

RussianPasswordCrackers http://www.password-crackers.com/en/

Field-programmable gate array (FPGA)

- Tableau TACC1441 Hardware Accelerator $3900
  - http://www.tableau.com/
  - AccessData PRTK support - TACC_Install.pdf
  - http://www.digitalintelligence.com/products/rack-a-tacc/
- Bruce Schneier - Secure Passwords Keep You Safer
  - http://www.schneier.com/essay-148.html
- NSA (at) Home
  - Breaks 800 hashes concurrently
  - http://nsa.unaligned.org/

Rack-A-TACK 2U module, $20000

Rack-A-TACC™

Performance Data