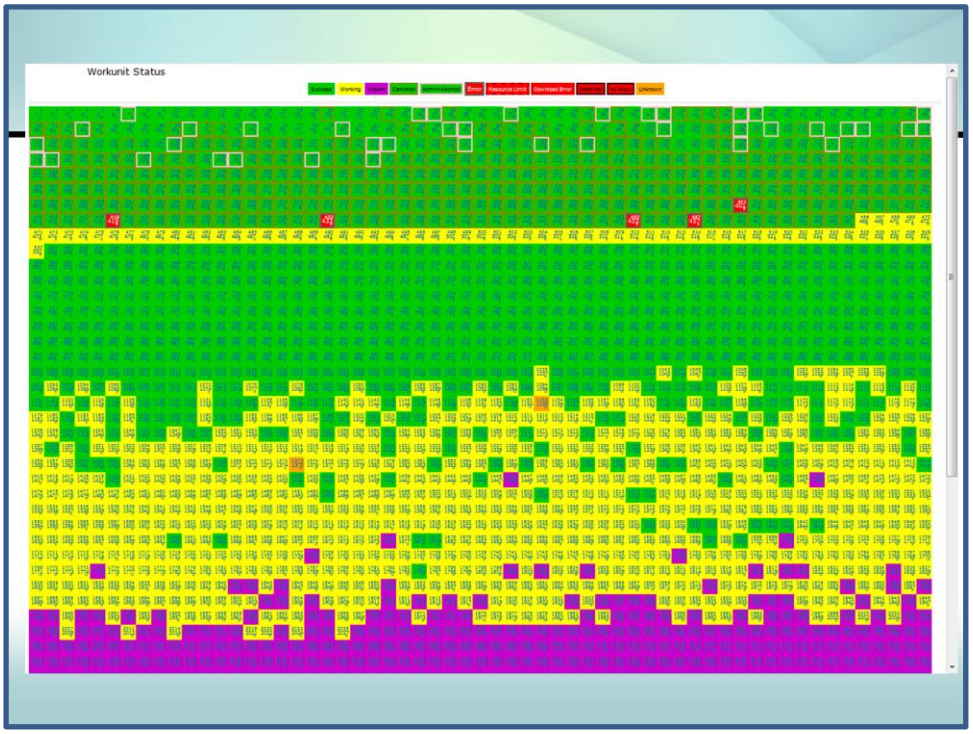
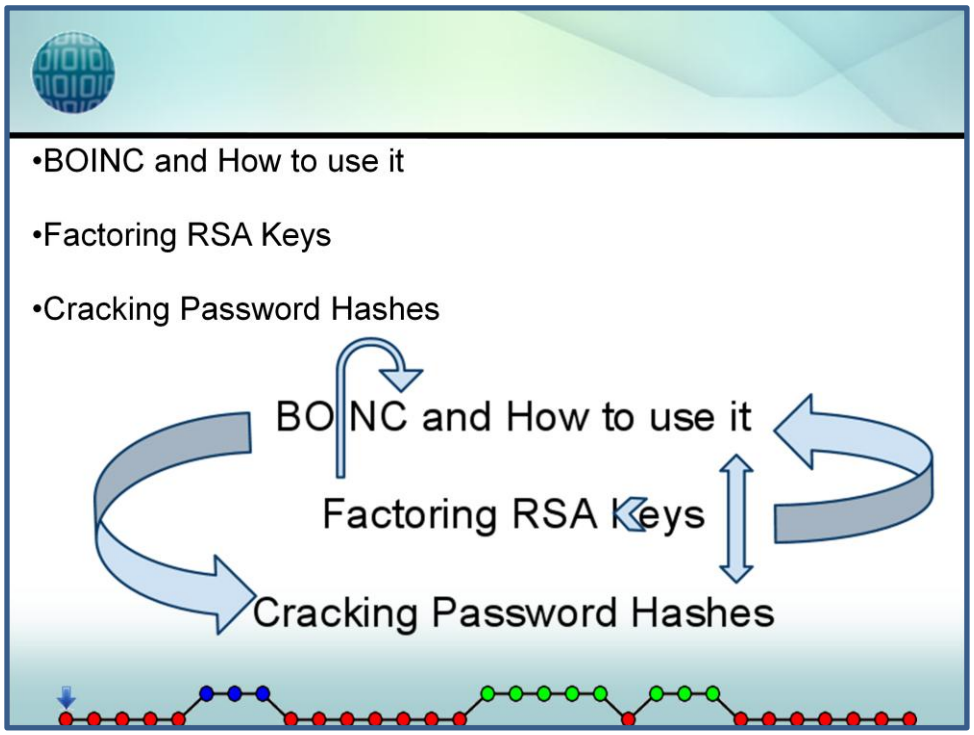
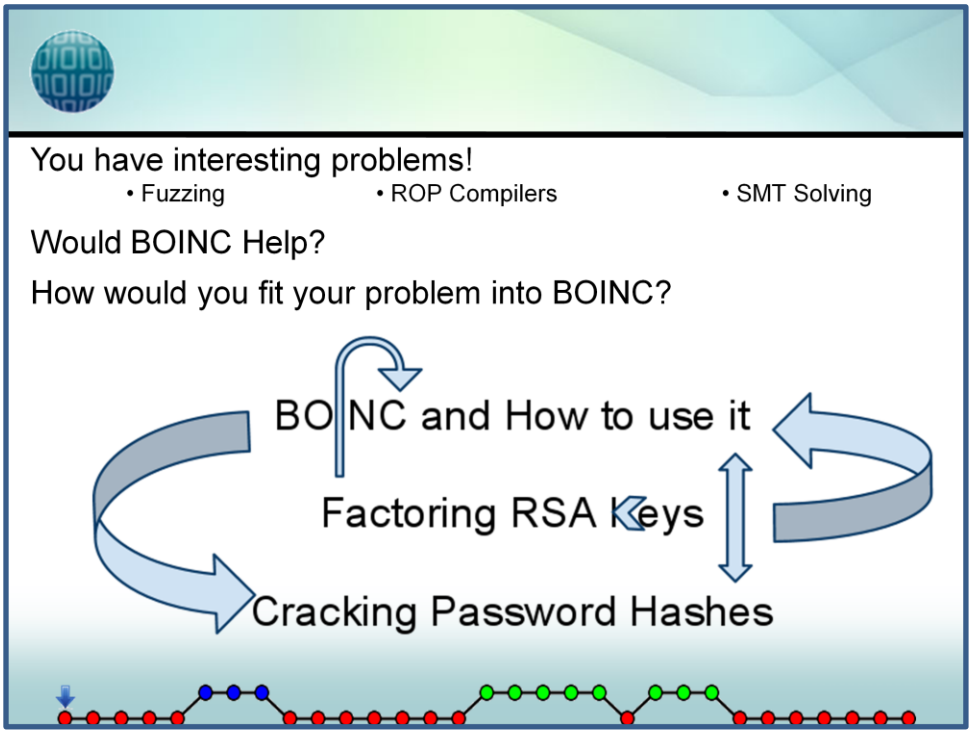

GOTHAM
DIGITAL • SCIENCE
Cloud & Control
Factoring RSA Keys & Cracking Passwords

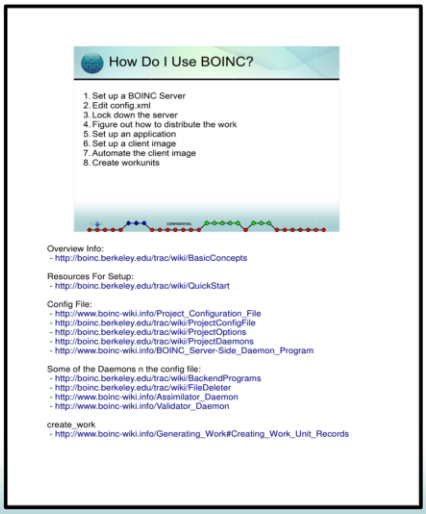


Custom BOINC Dashboard I wrote
(on github)

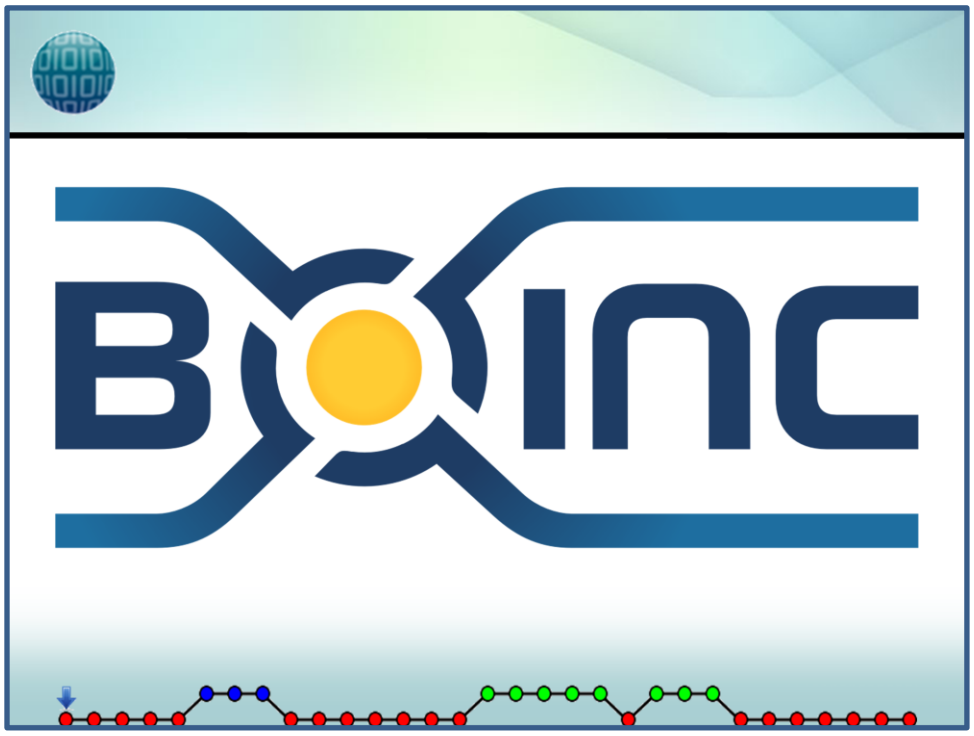

•BOINC and How to use it
•Factoring RSA Keys
•Cracking Password Hashes
BOINC and How to use it
Factoring RSA Keys
Cracking Password Hashes

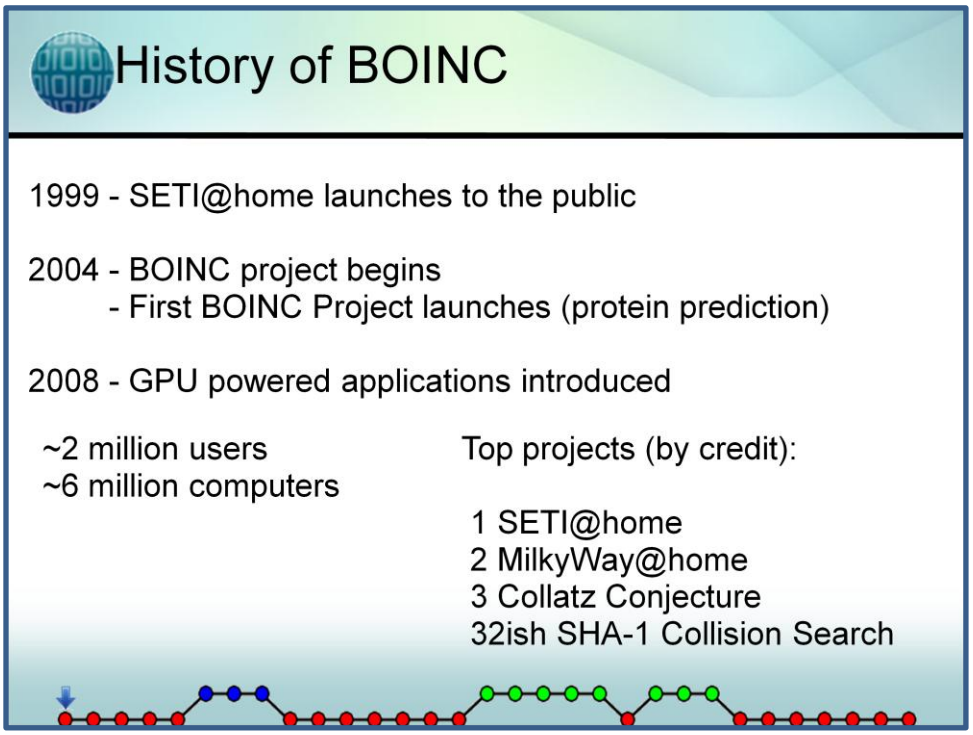

You have interesting problems!
• Fuzzing
• ROP Compilers
• SMT Solving
Would BOINC Help?
How would you fit your problem into BOINC?
BOINC and How to use it
Factoring RSA Keys
Cracking Password Hashes

Materials!

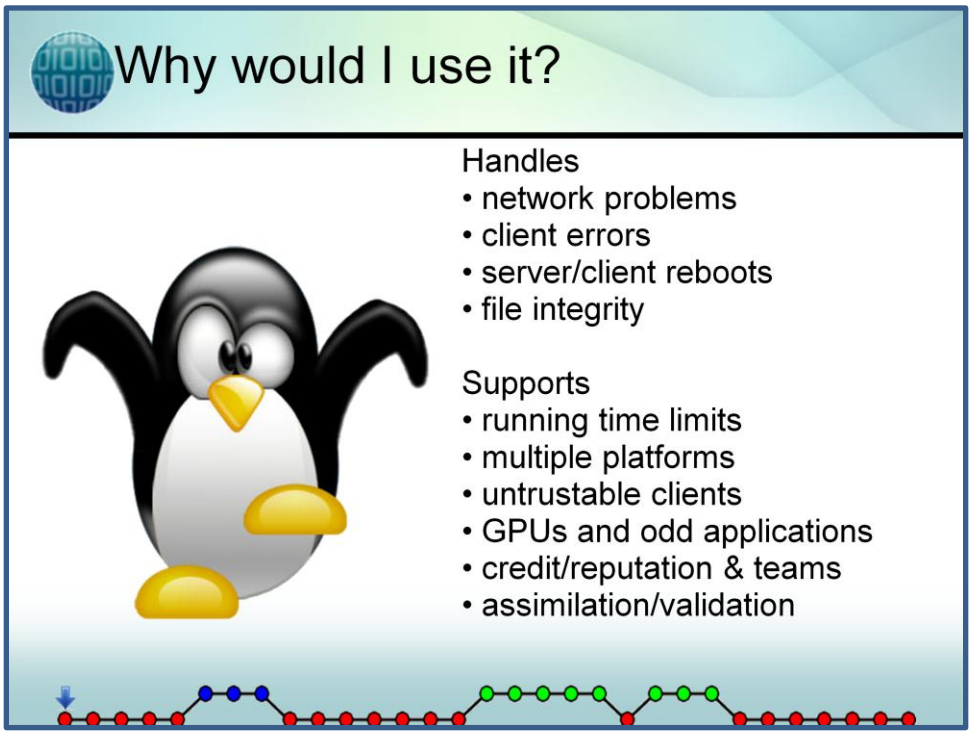


- Slides w/ references
- Sample Templates
- Scripts
- https://github.com/GDSSecurity/cloud-and-control


BOINC

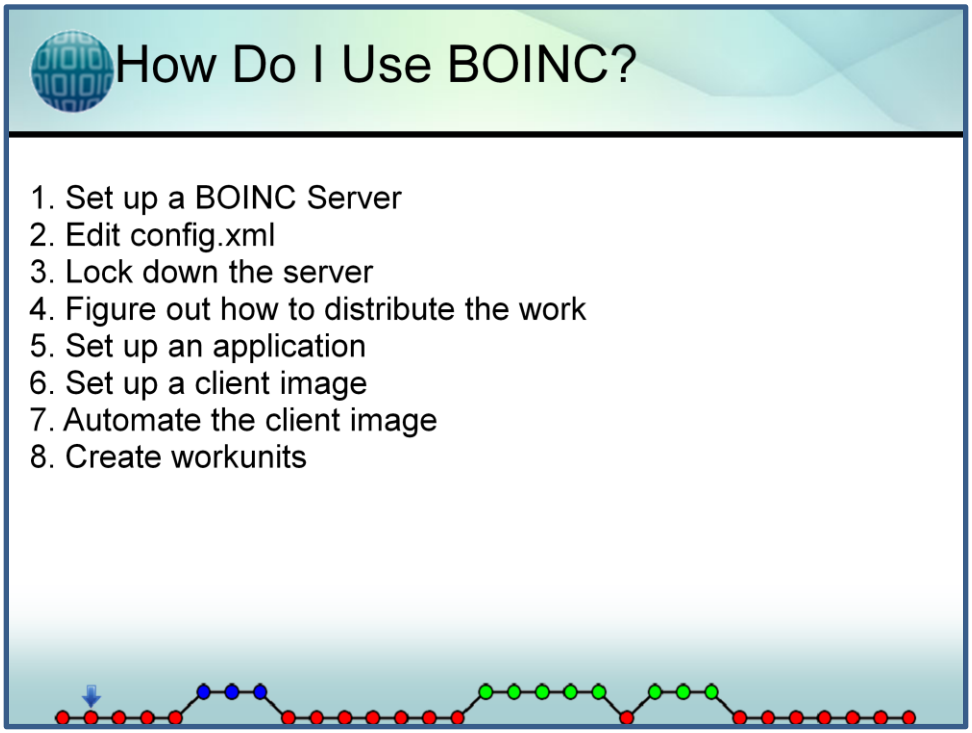


SHA-1 Collision Search
- http://www.isgtw.org/?pid=1000711



Platforms BOINC Runs on

- http://boinc.berkeley.edu/trac/wiki/BoincPlatforms



Overview Info:

- http://boinc.berkeley.edu/trac/wiki/BasicConcepts

Resources For Setup:

- http://boinc.berkeley.edu/trac/wiki/QuickStart

Config File:

- http://www.boinc-wiki.info/Project_Configuration_File
- http://boinc.berkeley.edu/trac/wiki/ProjectConfigFile
- http://boinc.berkeley.edu/trac/wiki/ProjectOptions
- http://boinc.berkeley.edu/trac/wiki/ProjectDaemons
- http://www.boinc-wiki.info/BOINC_Server-Side_Daemon_Program

Some of the Daemons n the config file:

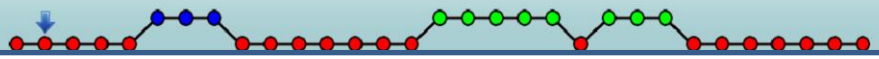
- http://boinc.berkeley.edu/trac/wiki/BackendPrograms
- http://boinc.berkeley.edu/trac/wiki/FileDeleter
- http://www.boinc-wiki.info/Assimilator_Daemon
- http://www.boinc-wiki.info/Validator_Daemon

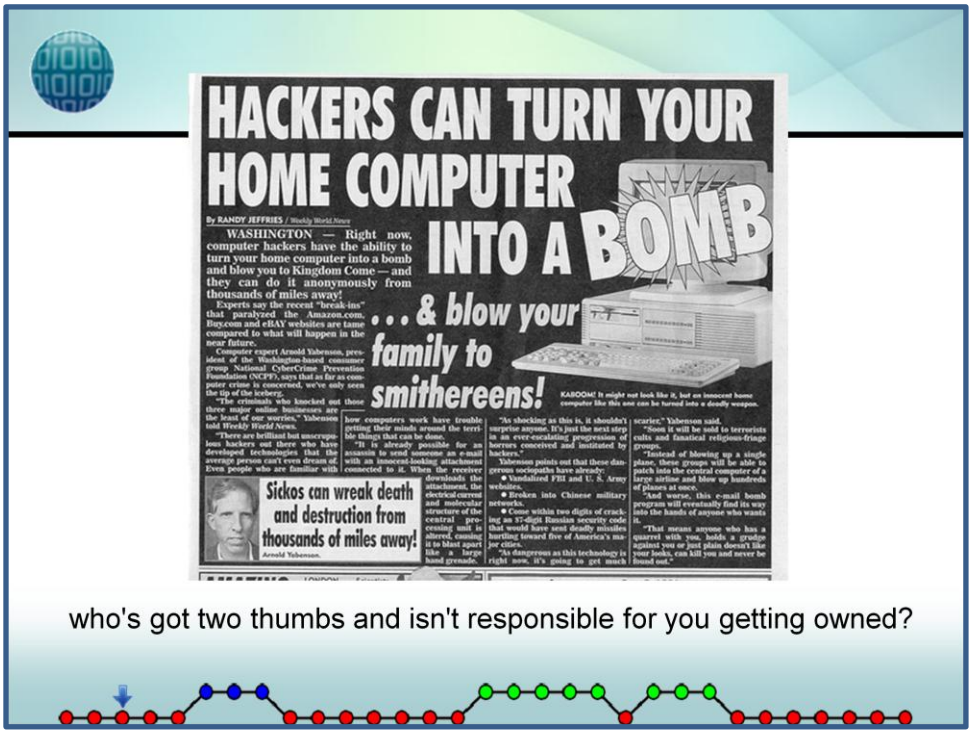
create_work

- http://www.boinc-wiki.info/Generating_Work#Creating_Work_Unit_Records

Is it hard?

1. Set up a BOINC Server - Easy
2. Edit config.xml - Easy
3. Lock down the server - Should be easy
4. Figure out how to distribute the work - Could be tricky
5. Set up an application - Trial and Error
6. Set up a client image - Easy
7. Automate the client image - Easy
8. Create workunits - Potentially annoying



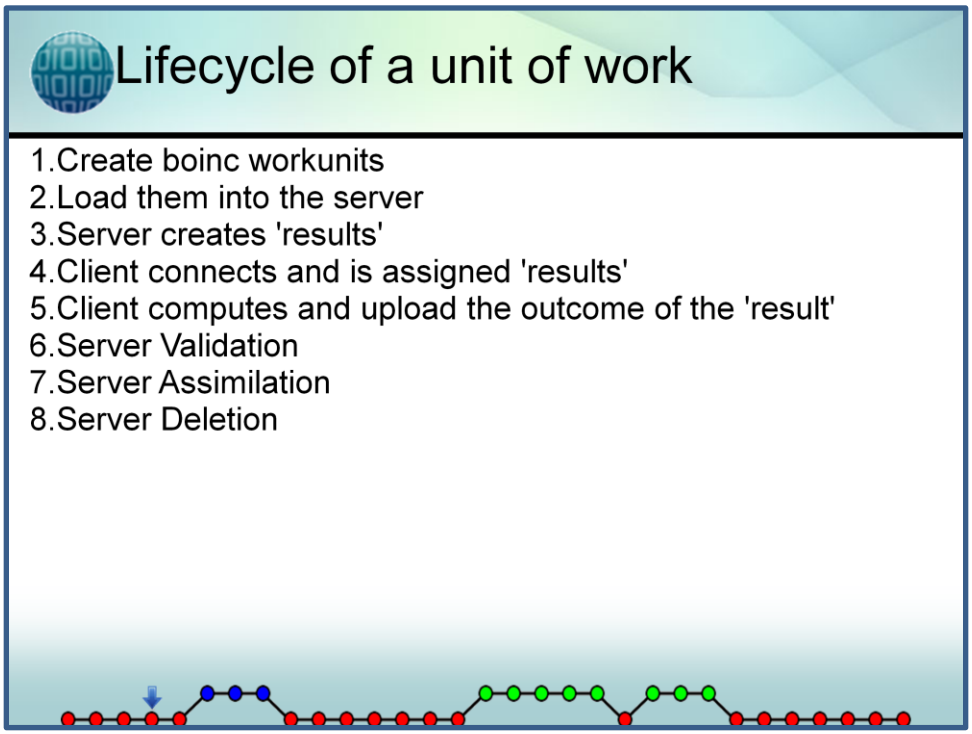


Threat Model Doc

- http://boinc.berkeley.edu/trac/wiki/SecurityIssues

In a large-scale volunteer project, BOINC strongly recommends proper code signing practices.

- http://boinc.berkeley.edu/trac/wiki/CodeSigning



More Explanations

- http://boinc.berkeley.edu/trac/wiki/JobReplication
- http://www.boinc-wiki.info/Result
- http://www.boinc-wiki.info/Redundancy_and_Errors

Tables in the DB

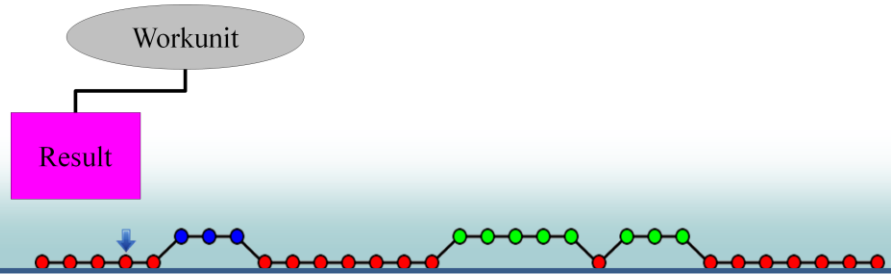
- http://www.boinc-wiki.info/Work_Unit_Record
- http://www.boinc-wiki.info/Result_Record

Work Distribution

- http://boinc.berkeley.edu/trac/wiki/WorkDistribution
- http://www.boinc-wiki.info/Work_Distribution

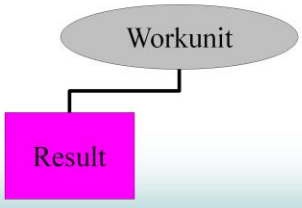
Lifecycle of a unit of work

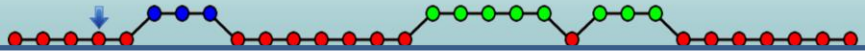
1. Create boinc workunits
2. Load them into the server
3. Server creates 'results'
4. Client connects and is assigned 'results'
5. Client computes and upload the outcome of the 'result'
6. Server Validation
7. Server Assimilation
8. Server Deletion



Lifecycle of a unit of work

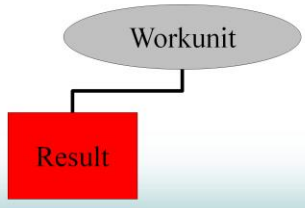
1. Create boinc workunits
2. Load them into the server
3. Server creates 'results'
4. Client connects and is assigned 'results'
5. Client computes and upload the outcome of the 'result'
6. Server Validation
7. Server Assimilation
8. Server Deletion

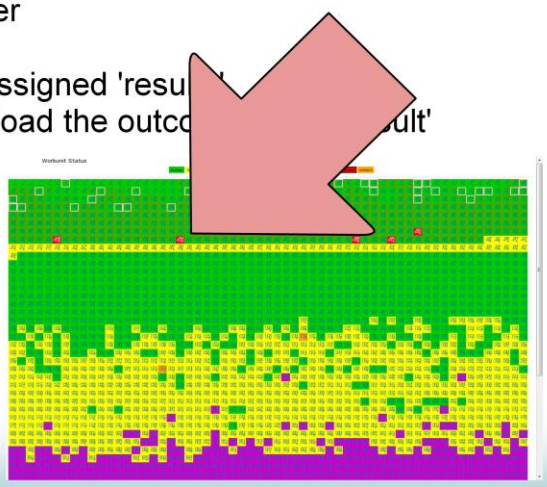


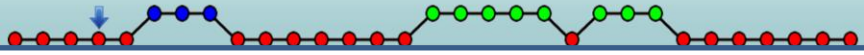


Lifecycle of a unit of work

1. Create boinc workunits
2. Load them into the server
3. Server creates 'results'
4. Client connects and is assigned 'resu
5. Client computes and upload the outc ult'
6. Server Validation
7. Server Assimilation
8. Server Deletion

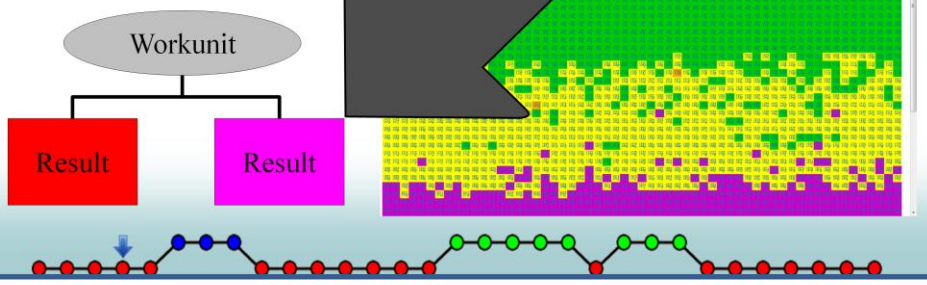






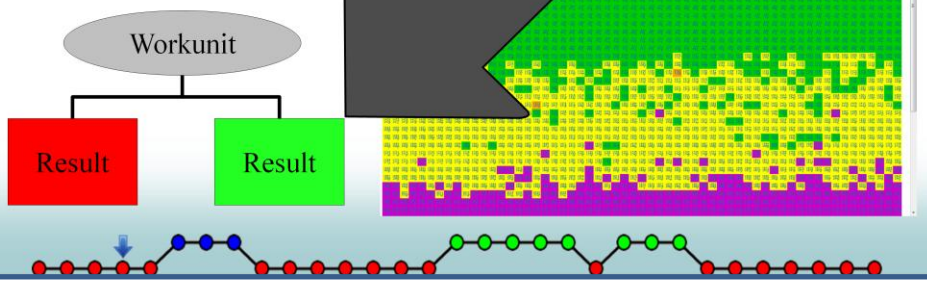
Lifecycle of a unit of work

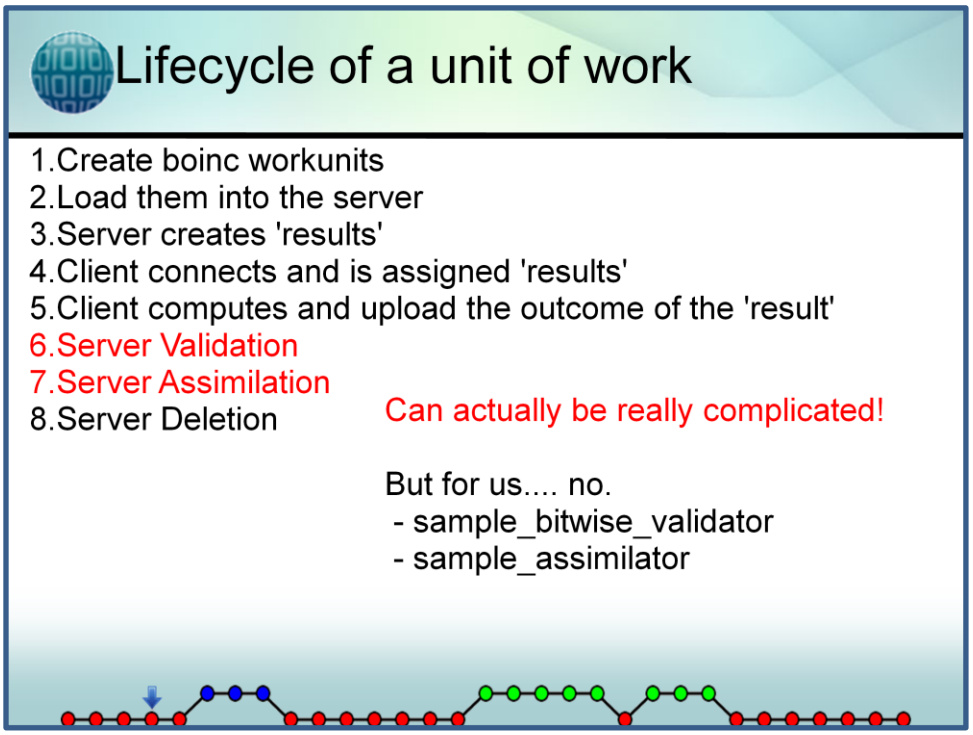
1. Create boinc workunits
2. Load them into the server
3. Server creates 'results'
4. Client connects and is assigned 'results'
5. Client computes and upload the outcome of the 'result'
6. Server Validation
7. Server Assimilation
8. Server Deletion



Lifecycle of a unit of work

1. Create boinc workunits
2. Load them into the server
3. Server creates 'results'
4. Client connects and is assigned 'results'
5. Client computes and upload the outcome of the 'result'
6. Server Validation
7. Server Assimilation
8. Server Deletion





Validator Info:

- http://boinc.berkeley.edu/trac/wiki/ValidationSummary
- http://boinc.berkeley.edu/trac/wiki/ValidationIntro
- http://www.boinc-wiki.info/Validator_Daemon
- http://www.boinc-wiki.info/Result_Validation

Writing Your Own Validator:

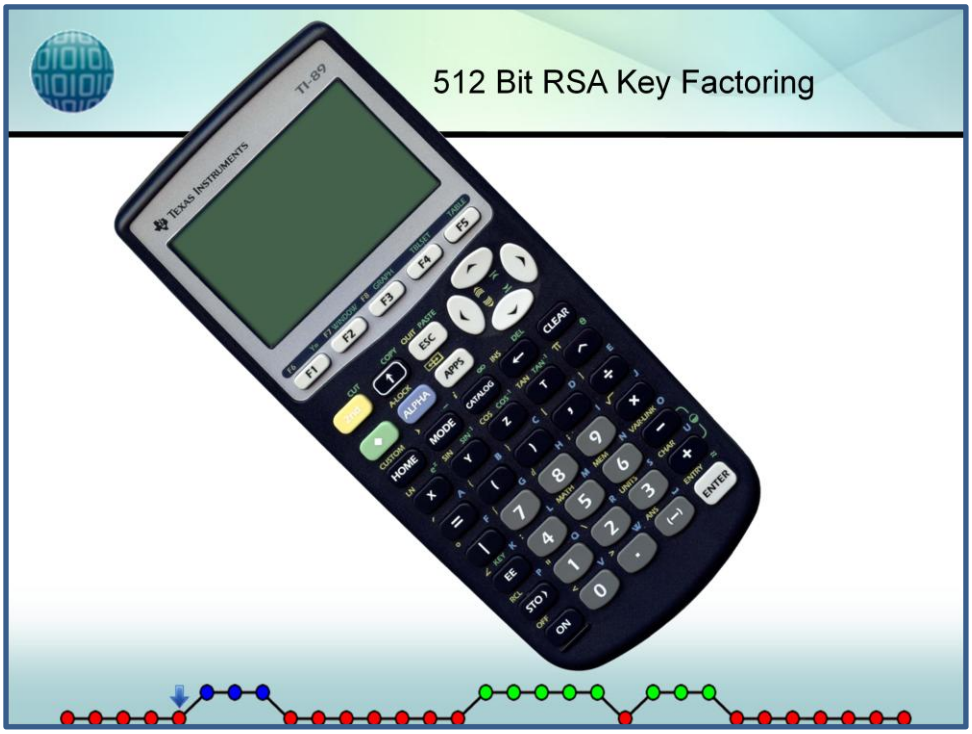
- http://boinc.berkeley.edu/trac/wiki/ValidationSimple

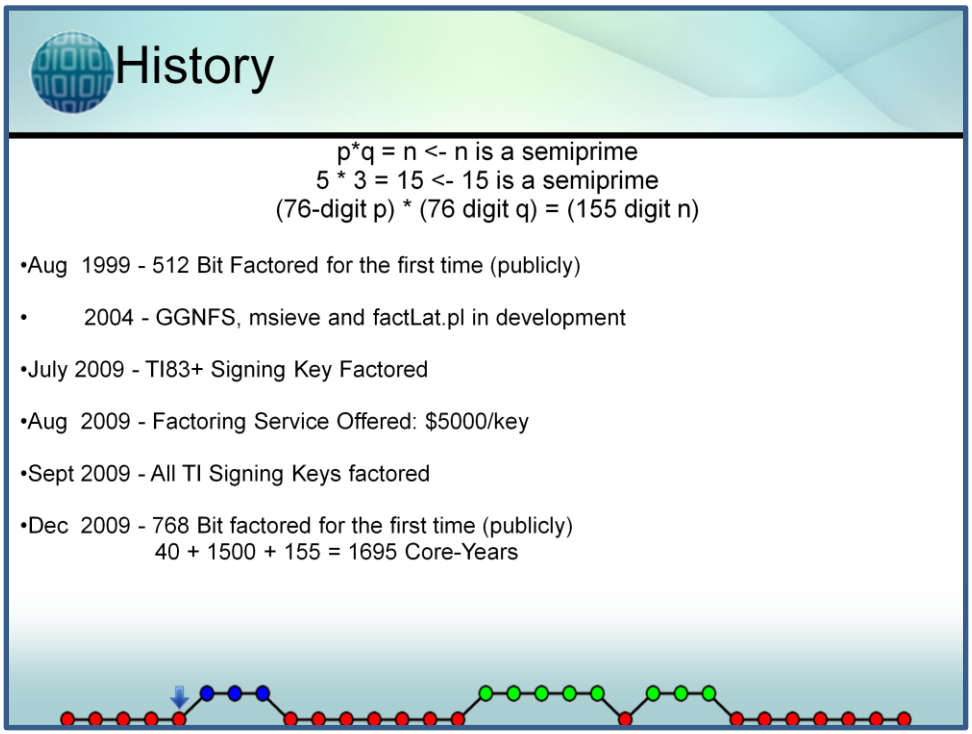
Assimilator Info

- http://boinc.berkeley.edu/trac/wiki/AssimilateIntro
- http://www.boinc-wiki.info/Result_Assimilation
- http://www.boinc-wiki.info/Assimilator_Daemon

Deletion Info

- http://www.boinc-wiki.info/Server-Side_File_Deletion
- http://www.boinc-wiki.info/Database_Purging_Utility


512 Bit RSA Key Factoring
TI-89
TEXAS INSTRUMENTS



http://forum.disk.net/security-services/10-factoring-rsa-512-service.html#post25

http://en.wikipedia.org/wiki/Texas_Instruments_signing_key_controversy
http://www.eff.org/press/archives/2009/10/13

Factoring a 768-bit semiprime took nonpublic tools. For discussion, see:
- http://www.mersenneforum.org/showthread.php?t=12958
- http://mersenneforum.org/showthread.php?t=15754

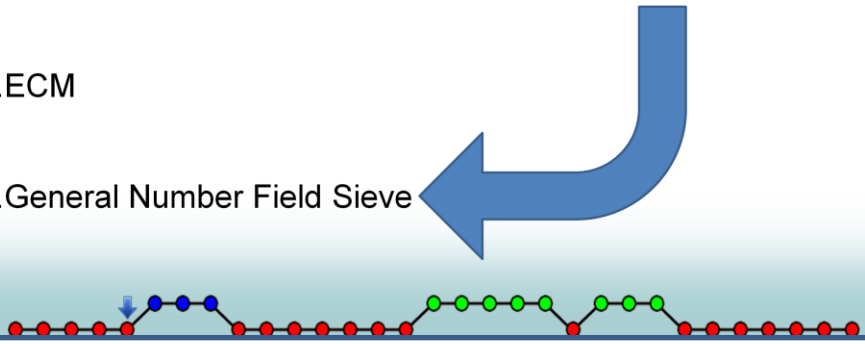
How Do I Factor

1. Trial Division?
   - Is it divisible by 2? 3? 5? 7? 11? 13

2. Pollard Rho

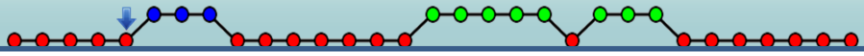
3. ECM

4. General Number Field Sieve



How Do I Factor - GNFS

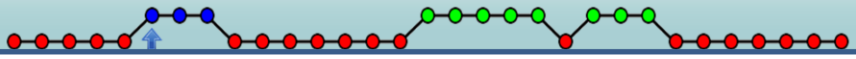
1. Polynomial Selection
2. Sieving
3. Combine

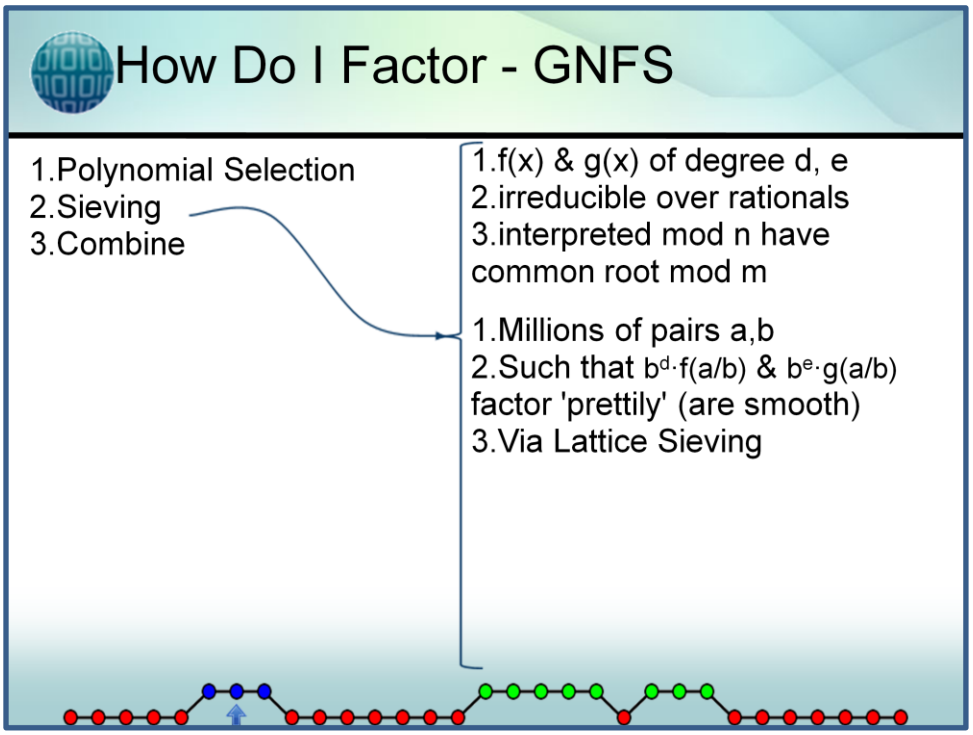


How Do I Factor - GNFS

1. Polynomial Selection
2. Sieving
3. Combine

1. f(x) & g(x) of degree d, e
2. irreducible over rationals
3. interpreted mod n have common root mod m





Some more on this:

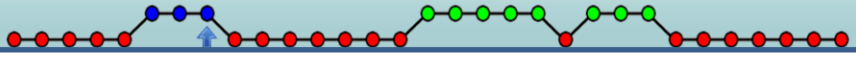
http://mersenneforum.org/showthread.php?t=15796

How Do I Factor - GNFS

1. Polynomial Selection
2. Sieving
3. Combine

1. f(x) & g(x) of degree d, e
2. irreducible over rationals
3. interpreted mod n have common root mod m

1. Millions of pairs a,b
2. Such that $b^d \cdot f(a/b)$ & $b^e \cdot g(a/b)$ factor 'prettily' (are smooth)
3. Via Lattice Sieving



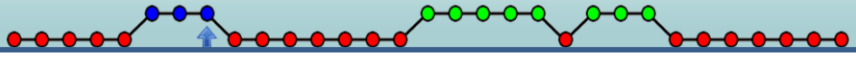
How Do I Factor - GNFS

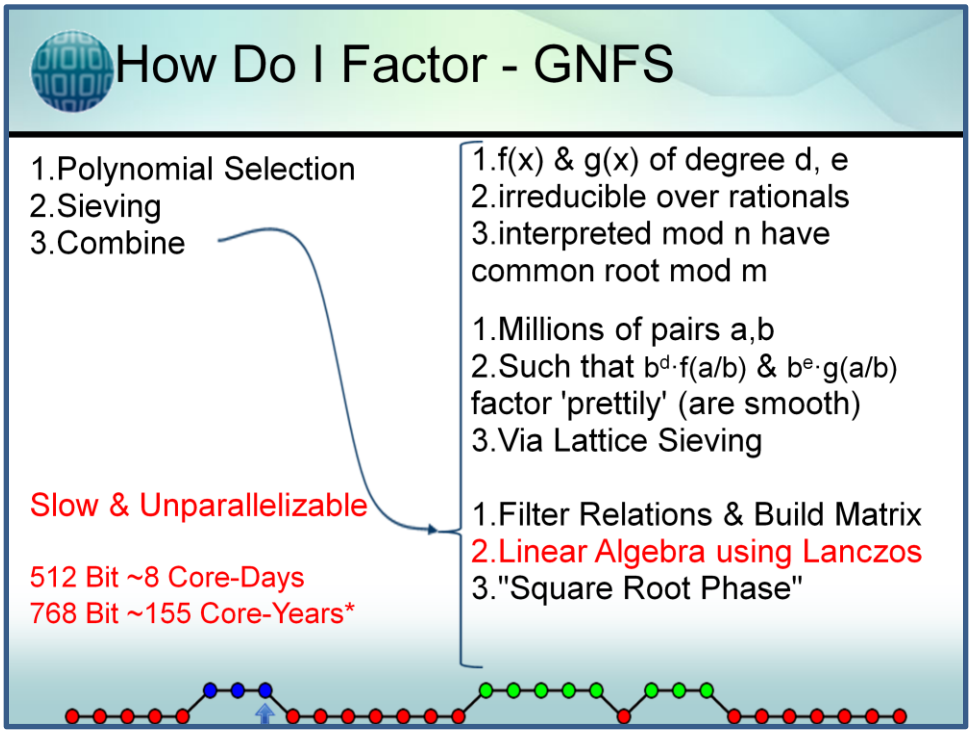
1. Polynomial Selection
2. Sieving
3. Combine

1. f(x) & g(x) of degree d, e
2. irreducible over rationals
3. interpreted mod n have common root mod m

1. Millions of pairs a,b
2. Such that $b^d \cdot f(a/b)$ & $b^e \cdot g(a/b)$ factor 'prettily' (are smooth)
3. Via Lattice Sieving

1. Filter Relations & Build Matrix
2. Linear Algebra using Lanczos
3. "Square Root Phase"



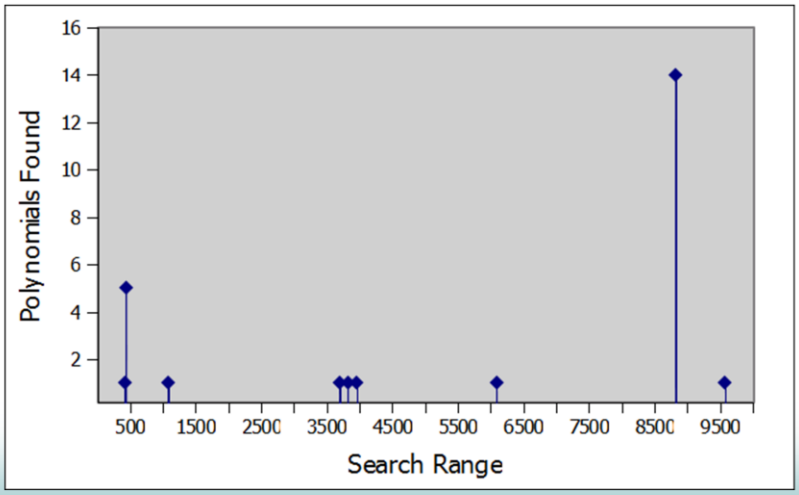


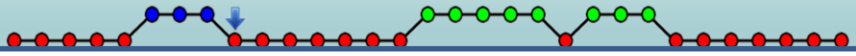
Why is it unparrellizable?
http://www.mersenneforum.org/showthread.php?t=15361

* is because the 768 bit semiprime used Block Weildmann as opposed to msieve's block lanczos algorithm.
http://www.mersenneforum.org/showthread.php?t=12958

How Do I Factor

1.Polynomial Selection





msieve by jasonp

Beautiful C Code
All Factoring Algorithms
- Trial Division
- Phollard Rho
- ECM
- GNFS

Actively Developed & Maintained
Active Support Channel
Active Community

(& happy ending)

Polynomial Selection

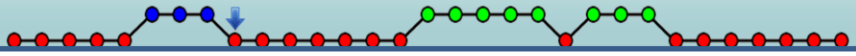
1.f(x) & g(x) of degree d, e
2.irreducible over rationals
3.interpreted mod n have common root mod m

Sieving

1.Millions of pairs a,b
2.Such that $b^{d} \cdot f(a/b)$ & $b^{e} \cdot g(a/b)$ factor 'prettily' (are smooth)
3.Via Lattice Sieving

Combine

1.Filter Relations & Build Matrix
2.Linear Algebra using Lanczos
3."Square Root Phase"



msieve by jasonp

jasonp?

Polynomial Selection

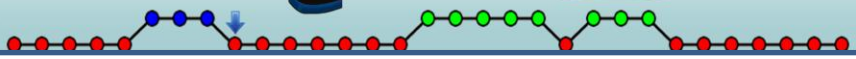
1. f(x) & g(x) of degree d, e
2. irreducible over rationals
3. interpreted mod n have common root mod m

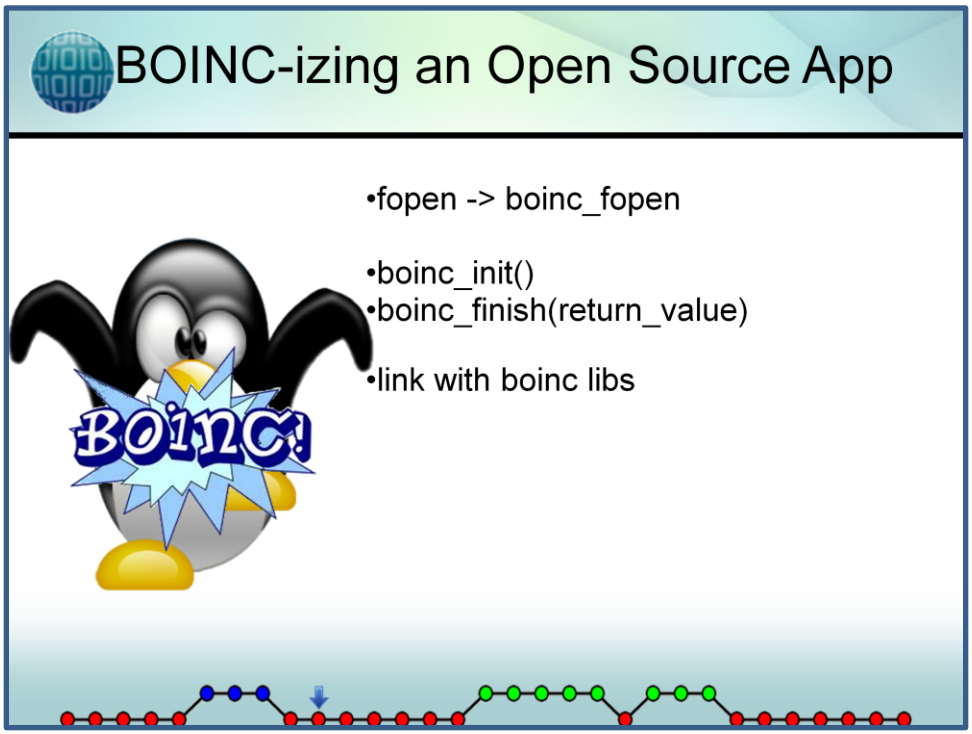
Sieving

1. Millions of pairs a,b
2. Such that $b^d \cdot f(a/b)$ & $b^e \cdot g(a/b)$ factor 'prettily' (are smooth)
3. Via Lattice Sieving

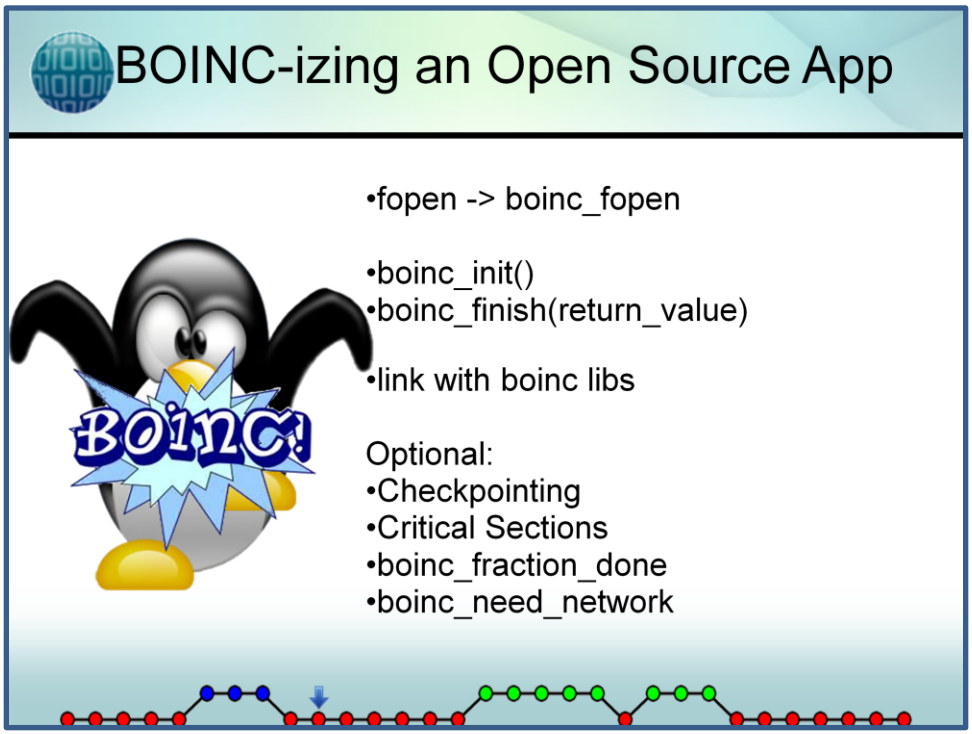
Combine

1. Filter Relations & Build Matrix
2. Linear Algebra using Lanczos
3. "Square Root Phase"





boinc_init
- http://boinc.berkeley.edu/trac/wiki/OptionsApi

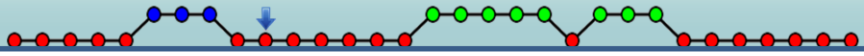


Writing your own application ground-up for BOINC is outside the scope of this talk, as is porting an application 'fully' to BOINC to take advantage of all its options. But you can start here:

http://www.boinc-wiki.info/BOINC_Development#Introduction_to_Developing_a_BOINC_Application

Rewiring msieve into a BOINC Application

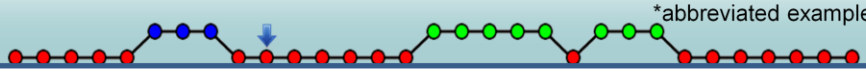
```
@@ -2852,7 +2891,33 @@
+#ifdef HAVE_BOINC
 int main(int argc, char **argv)
+{
+ int newArgc, ret;
+ char** newArgv;
+ myboincstart(&newArgc, &newArgv, argv[0]);
+ ret = sieve_main(newArgc, newArgv);
+ boinc_finish(ret);
+ return ret;
+}
+
+
+int sieve_main(int argc, char **argv)
+#else
+int main(int argc, char **argv)
+#endif
 {
```



Rewiring msieve into a BOINC Application

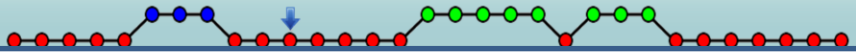
```
void myboincstart(int* argc, char *** argv, char* name)
{
    char in[500], out[500];
    boinc_init();
    boinc_resolve_filename("input_data", in, 500);
    boinc_resolve_filename("output_data", out, 500);

    *argc = 0;
    argv = new char*[7];
    argv[(*argc)++] = name;
    argv[(*argc)++] = "-i";
    argv[(*argc)++] = in;
    argv[(*argc)++] = "-nf";
    argv[(*argc)++] = out;
    argv[(*argc)++] = "-np";
    argv[(*argc)++] = "\0";
}
```

*abbreviated example



BOINC-izing an Open Source App

- fopen -> boinc_fopen
- boinc_init()
- boinc_finish(return_value)
- link with boinc libs



BOINC-izing an Open Source App

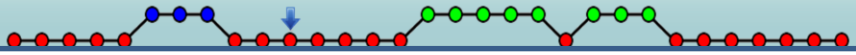
- fopen -> boinc_fopen

- boinc_init()
- boinc_finish(return_value)

- boinc_resolve_filename

```
fopen("logfile", "w")

boinc_resolve_filename("logfile", buffer);
boinc_fopen(buffer, "w")
// buffer -> workunit12345_0_1
```

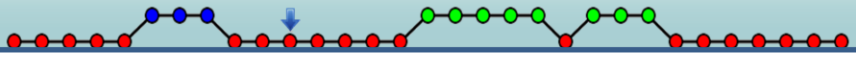
- link with boinc libs



Application Templates

Input

```
<file_info>
    <number>0</number>
     [ <sticky /> ]
     [ <nodelete /> ]
</file_info>
<workunit>
  <file_ref>
    <file_number>0</file_number>
    <open_name>rsakey</open_name>
    [ <copy_file/> ]
  </file_ref>

  <target_nresults>1</target_nresults>
</workunit>
```



Info

- http://boinc.berkeley.edu/trac/wiki/BoincFiles
- http://boinc.berkeley.edu/trac/wiki/BasicApi#filenames
- http://boinc.berkeley.edu/trac/wiki/JobSubmission
- http://www.boinc-wiki.info/Files_and_File_References

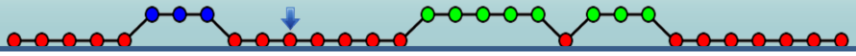
Application Templates

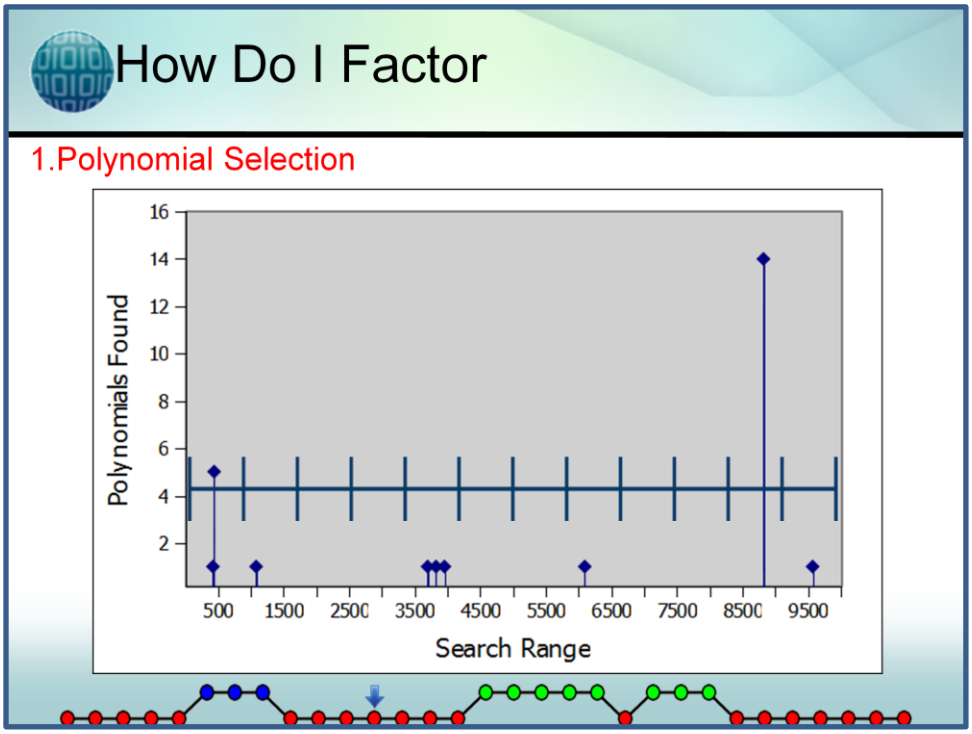
Input

```
<file_info>
    <number>0</number>
     [ <sticky /> ]
     [ <nodelete /> ]
</file_info>
<workunit>
  <file_ref>
    <file_number>0</file_number>
    <open_name>rsakey</open_name>
    [ <copy_file/> ]
  </file_ref>

  <target_nresults>1</target_nresults>
</workunit>
```

Output

```
<file_info>
  <name><OUTFILE_0/></name>
  <generated_locally/>
  <upload_when_present/>
  <url><UPLOAD_URL/></url>
</file_info>
<result>
 <file_ref>
  <file_name><OUTFILE_0/>
             </file_name>
 <open_name>logfile</open_name>
 [ <copy_file>0|1</copy_file> ]
 [ <optional>0|1</optional> ]
 </file_ref>
</result>
```





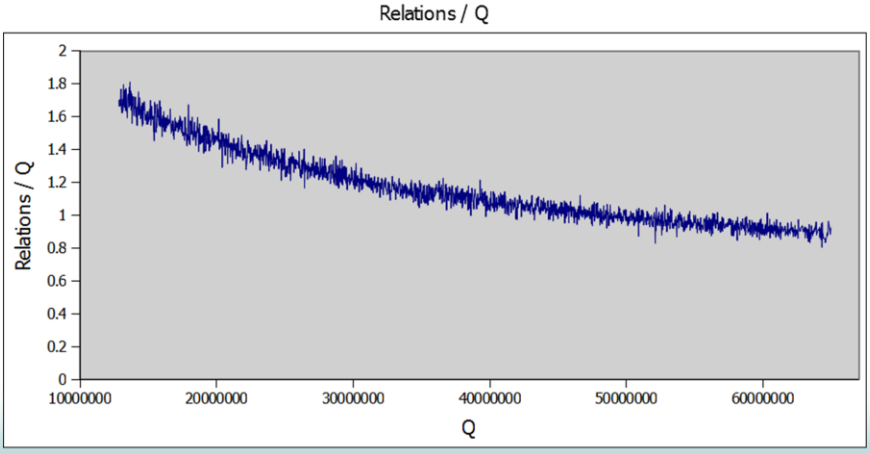
Msieve does have the ability to do GPU polynomial selection – but it's not important to get that running. It's probably more cost effective to do the CPU polynomial selection. Logic says problems should run much faster on the GPU, but instruction set-level optimizing and algorithms don't necessarily translate well to the GPU – it's not magic pixie dust.

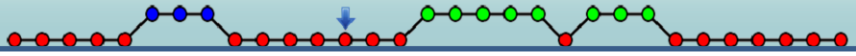
See http://mersenneforum.org/showthread.php?t=15725

How Do I Factor

1.Polynomial Selection

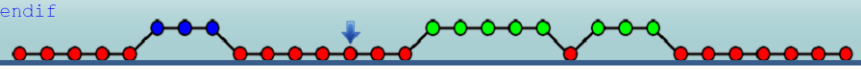
2.Sieving





Sieving with GGNFS in BOINC

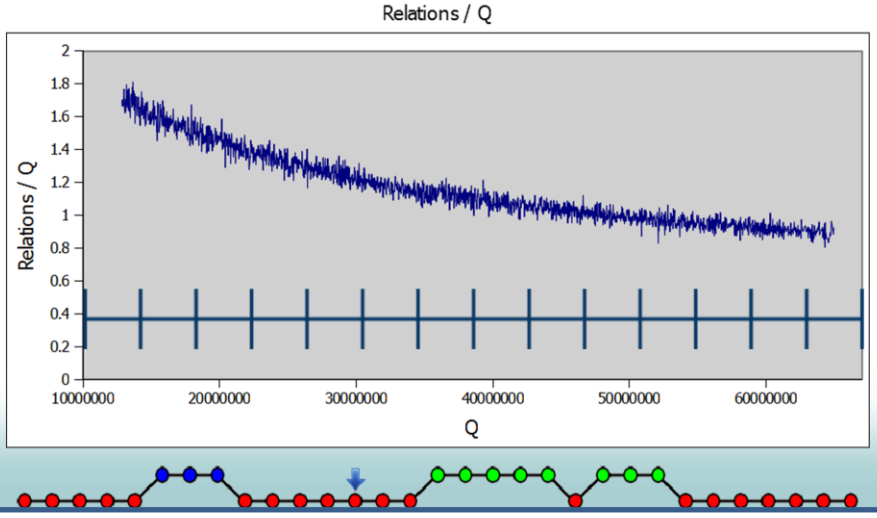
```
+#ifdef HAVE_BOINC
+int boincstart(int argc_init, char **argv) {
+  boinc_init();
+  boinc_resolve_filename("input_data", path_in, sizeof(path_in) );
+  boinc_resolve_filename("output_data", path_out,sizeof(path_out));
+  argv[argc_init++]="-R";
+  argv[argc_init++]="-a";
+  argv[argc_init++]="-o";
+  argv[argc_init++]=path_out;
+  argv[argc_init++]=path_in;
+  return argc_init;
+}
int main(int argc, char **argv) {
+       int app_argc, retcode;
+       char* app_argv[ARGVCOUNT];
+       app_argv[0] = argv[0];
+       app_argc= boincstart(1,app_argv);
+       retcode= main_lasieve(app_argc,app_argv);
+       boinc_finish(retcode);
+       return retcode;
+}
+int main_lasieve(int argc, char **argv)
+#else
+int main(int argc, char **argv)
+#endif
```



How Do I Factor

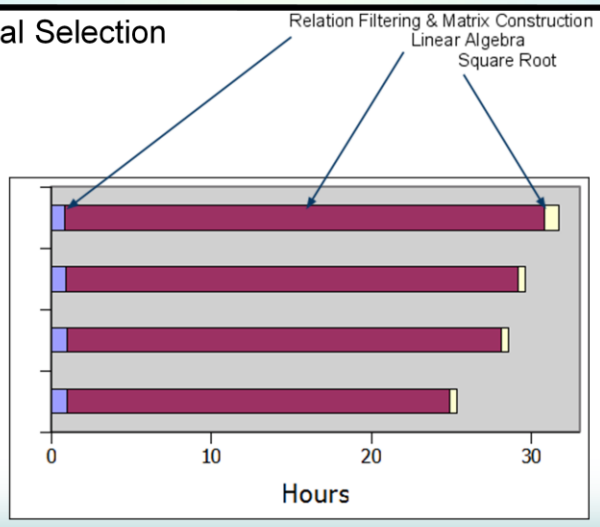
1.Polynomial Selection

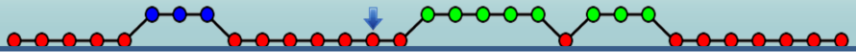
2.Sieving

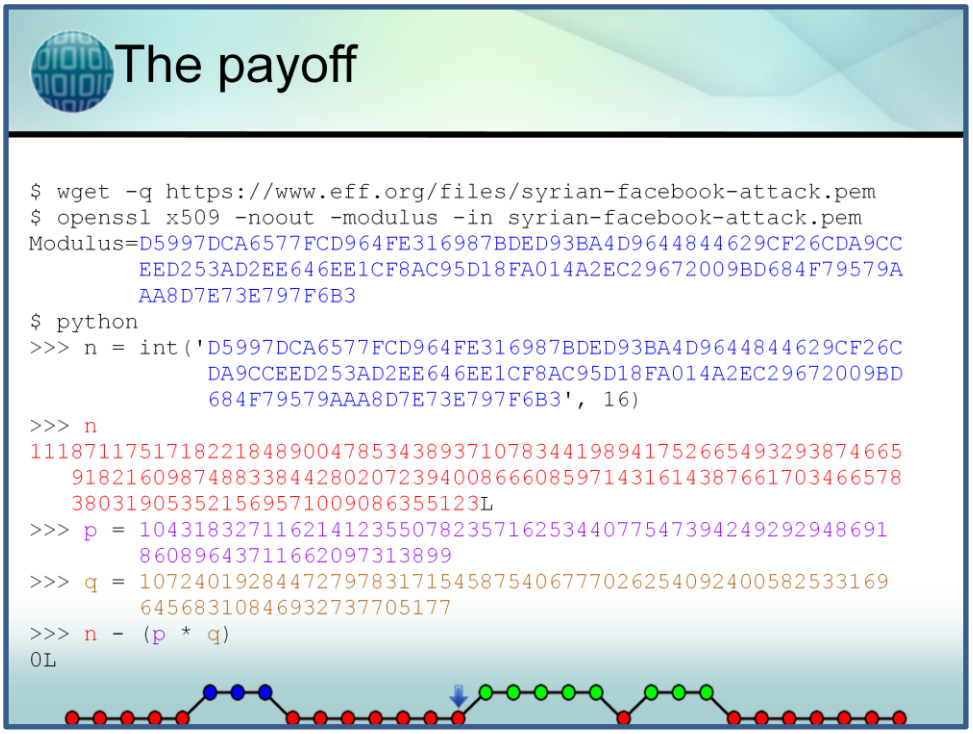


How Do I Factor

1. Polynomial Selection
2. Sieving
3. Combine

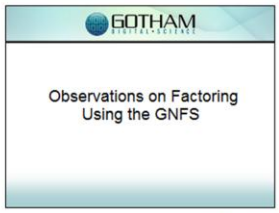






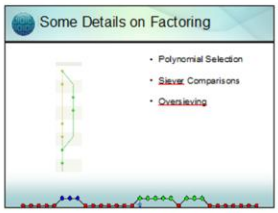
https://www.eff.org/deeplinks/2011/05/syrian-man-middle-against-facebook

Factoring Details

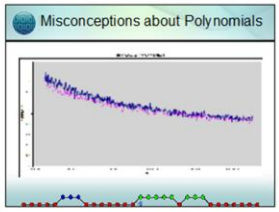


Cloud & Control

2

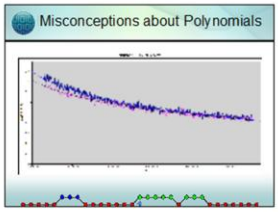


Some Details on Factoring



Misconceptions about Polynomials

4



Slide 4

Moved to their own slide deck for time/relevance

Available on github.

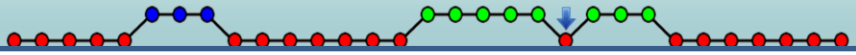
(Sorry!)

So Far

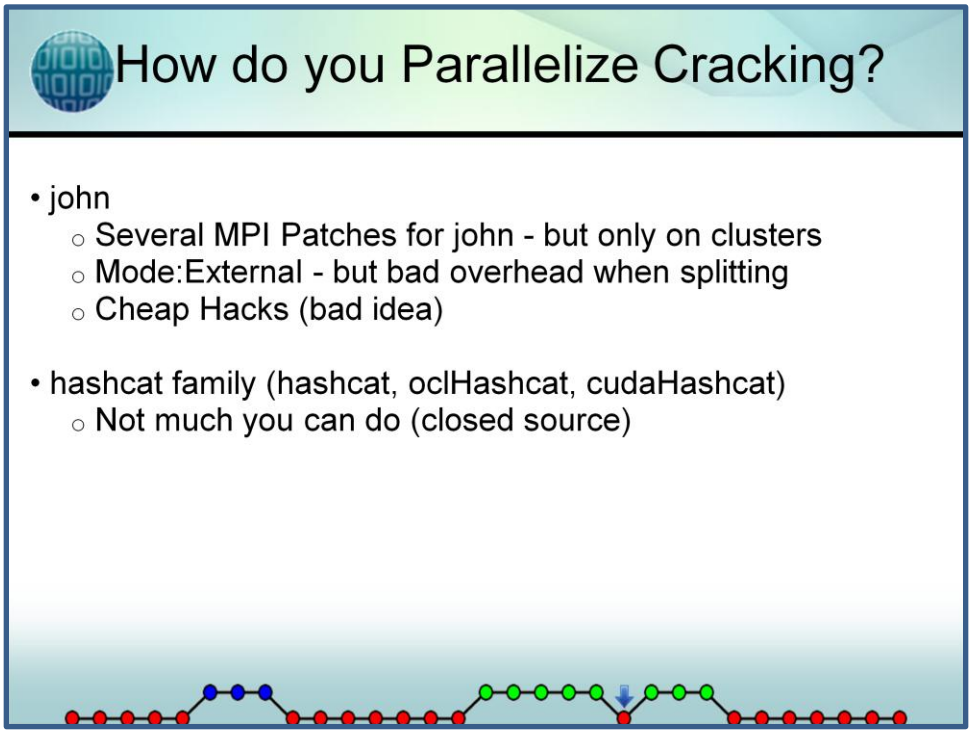
- BOINC
  - Why and How
  - Applications - Open source ⇨ BOINC Application
- Factoring RSA

Next

- BOINC
  - Close Source Applications
  - GPU Applications
- Hands-off Cracking Passwords

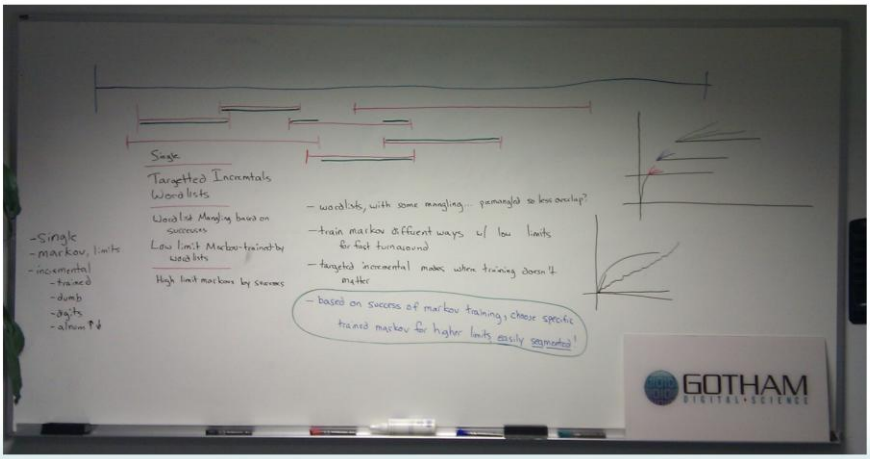


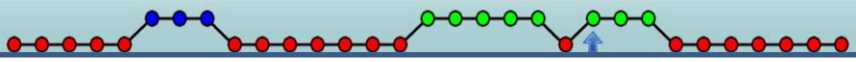
Cracking



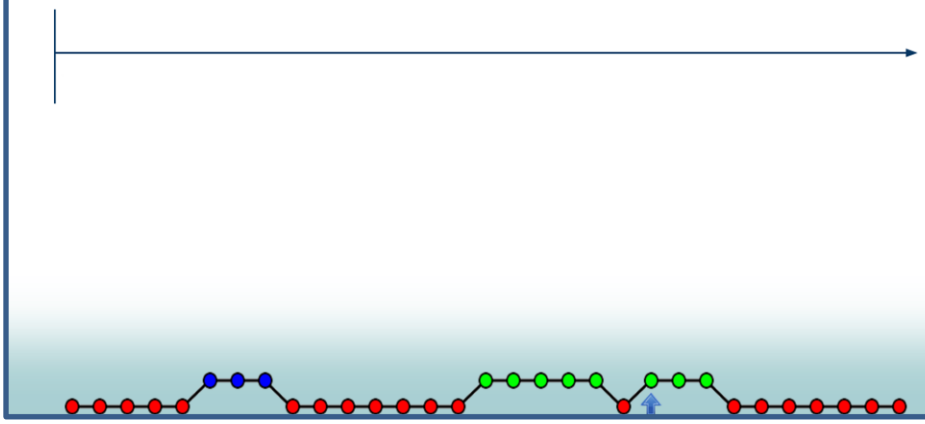
- http://openwall.info/wiki/john/parallelization

Enter the Magic

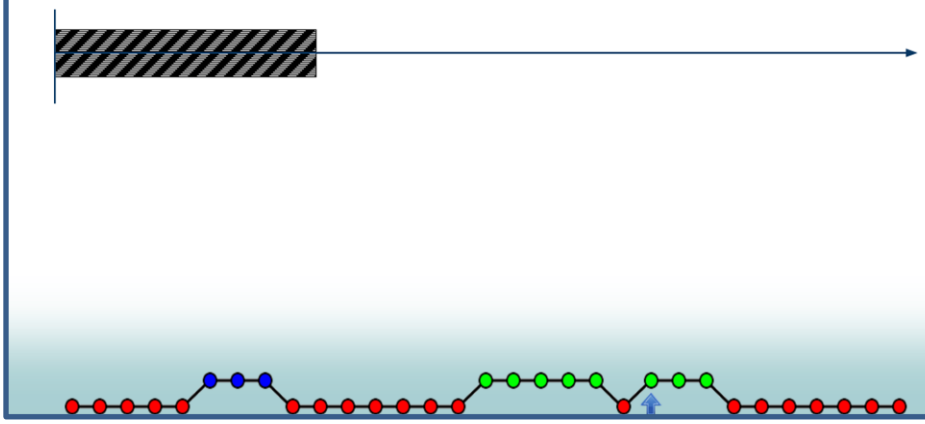




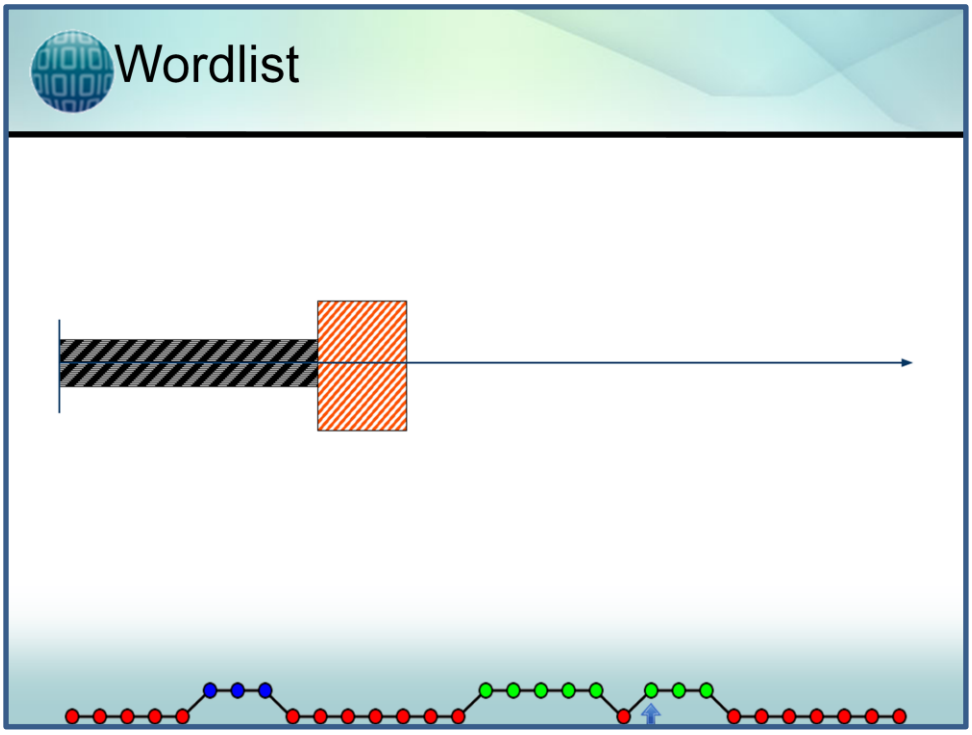
All Possible Passwords



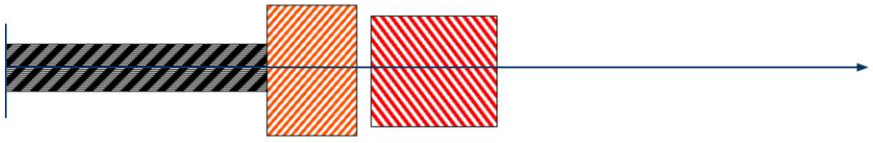
Brute Force

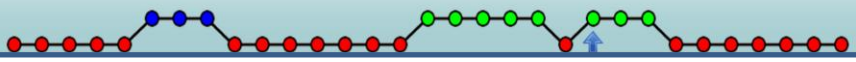


Wordlist

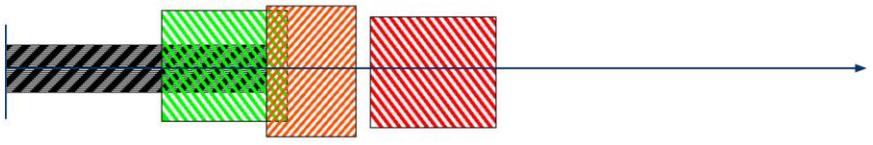


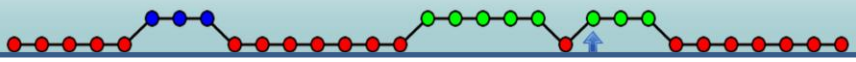
Two Wordlists!



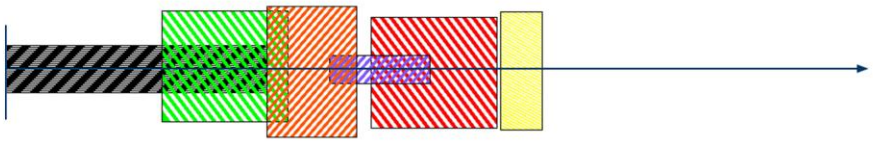


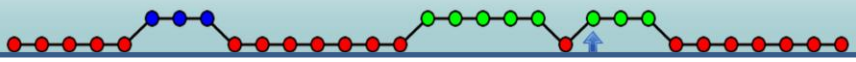
Let me try this....



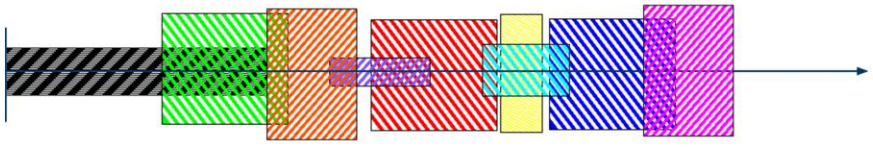


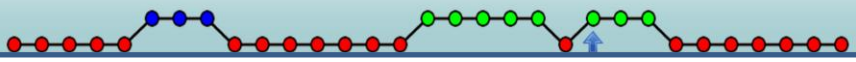
Foreign Wordlists!



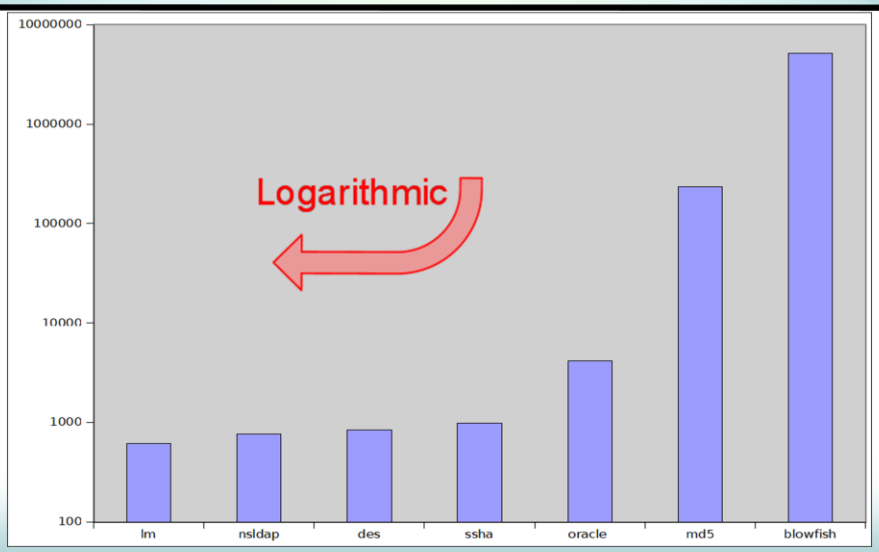


Kitchen Sink!



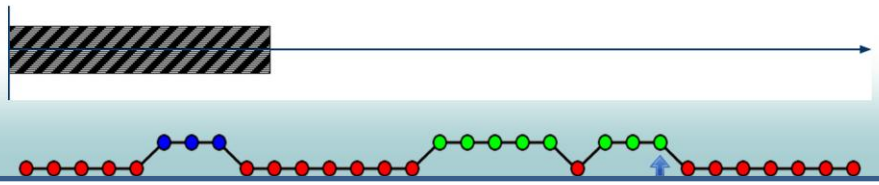


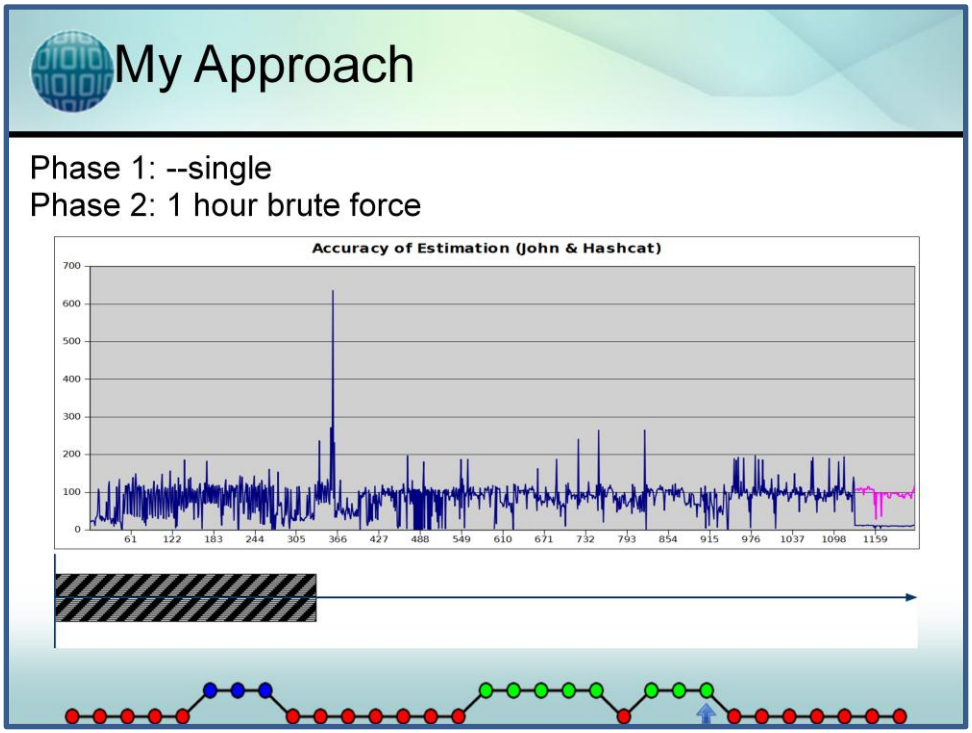
Not all hashes are created equal



My Approach

Phase 1: --single
Phase 2: 1 hour brute force



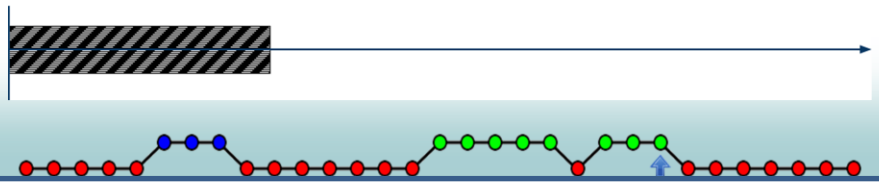


The purple is a constant-adjusted factor. I was off initially, but just multiplied it all, and it comes out nicely.

My Approach

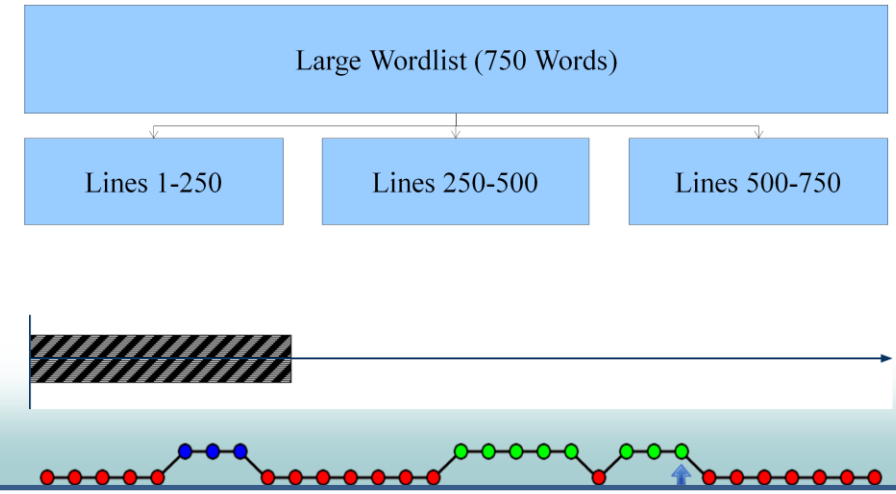
Phase 1: --single
Phase 2: 1 hour brute force

Large Wordlist (750 Words)



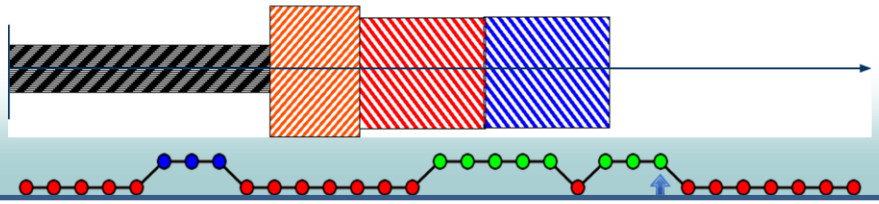
My Approach

Phase 1: --single
Phase 2: 1 hour brute force



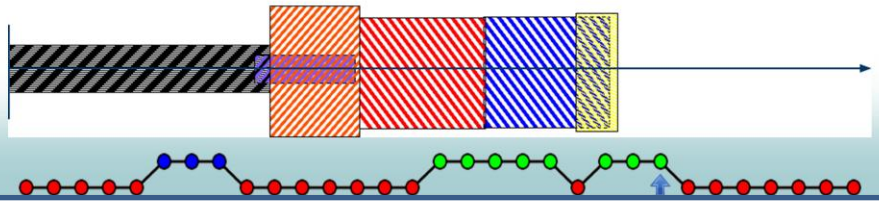
My Approach

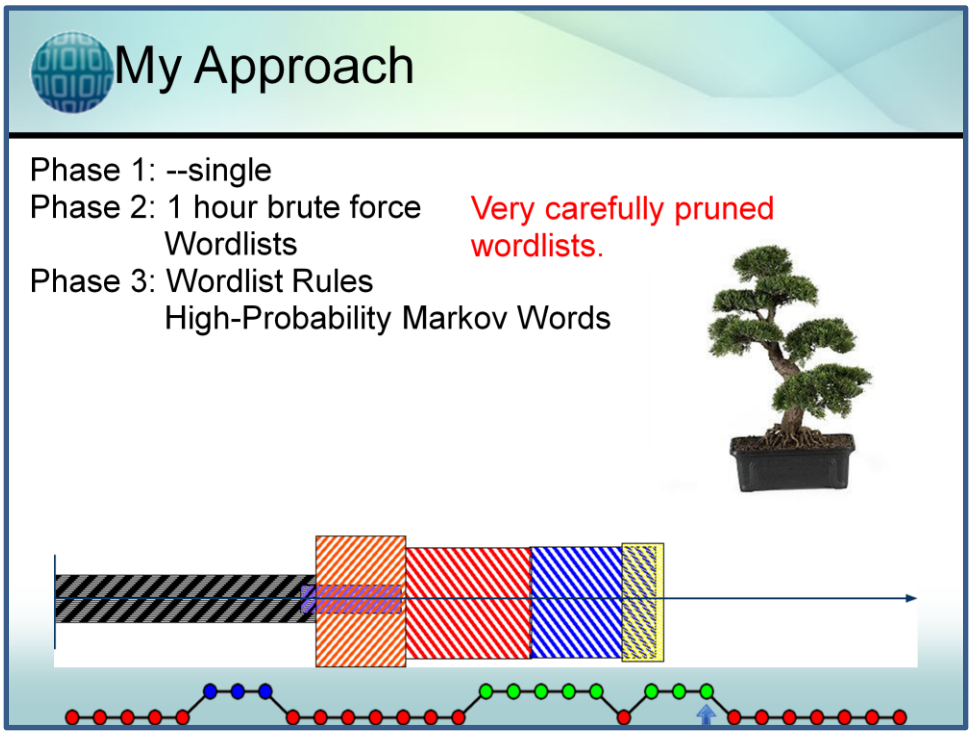
Phase 1: --single
Phase 2: 1 hour brute force
Wordlists



My Approach

Phase 1: --single
Phase 2: 1 hour brute force
Wordlists
Phase 3: Wordlist Rules
High-Probability Markov Words

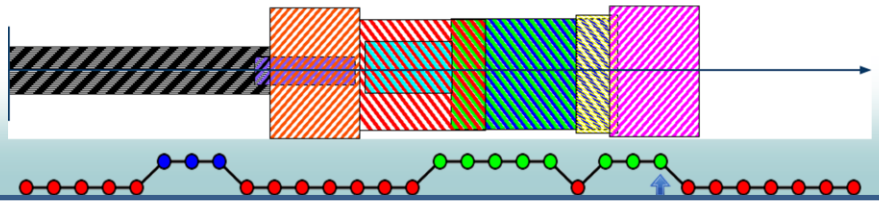


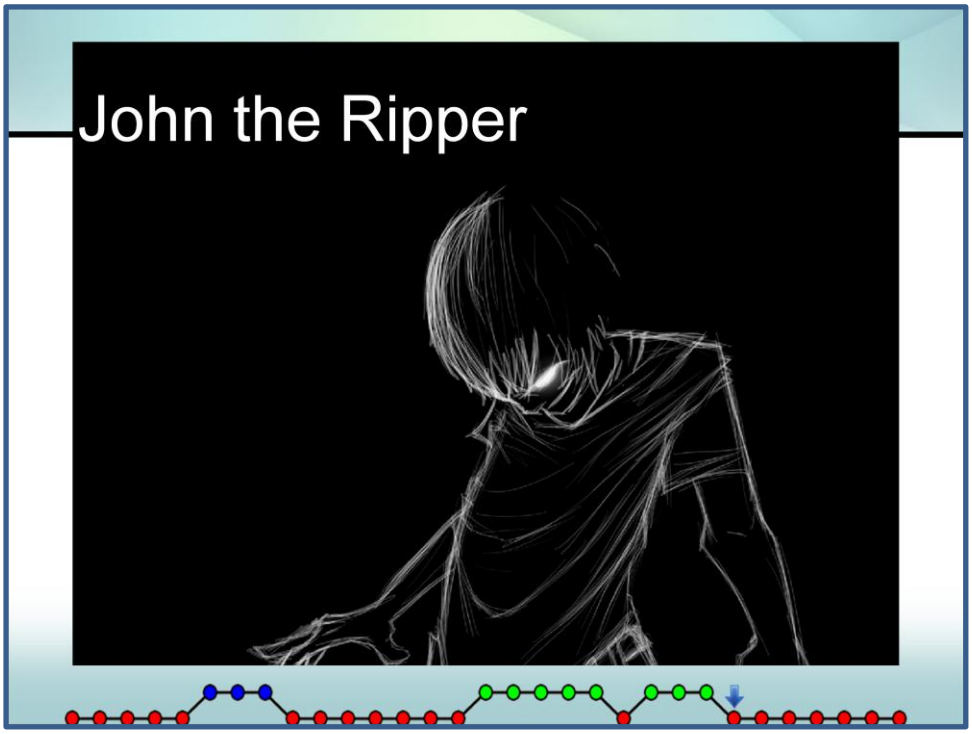


All wordlists, including the markow lists, were de-duped against each other, and the generated-mangling rules were also de-duped from the original lists.

My Approach

Phase 1: --single
Phase 2: 1 hour brute force
Wordlists
Phase 3: Wordlist Rules
High-Probability Markov Words
Phase 4: Phase 3 Markovs + Rules
Low-Probability Markov Words
Phase 5: Phase 4 Markovs + Rules



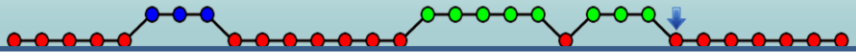


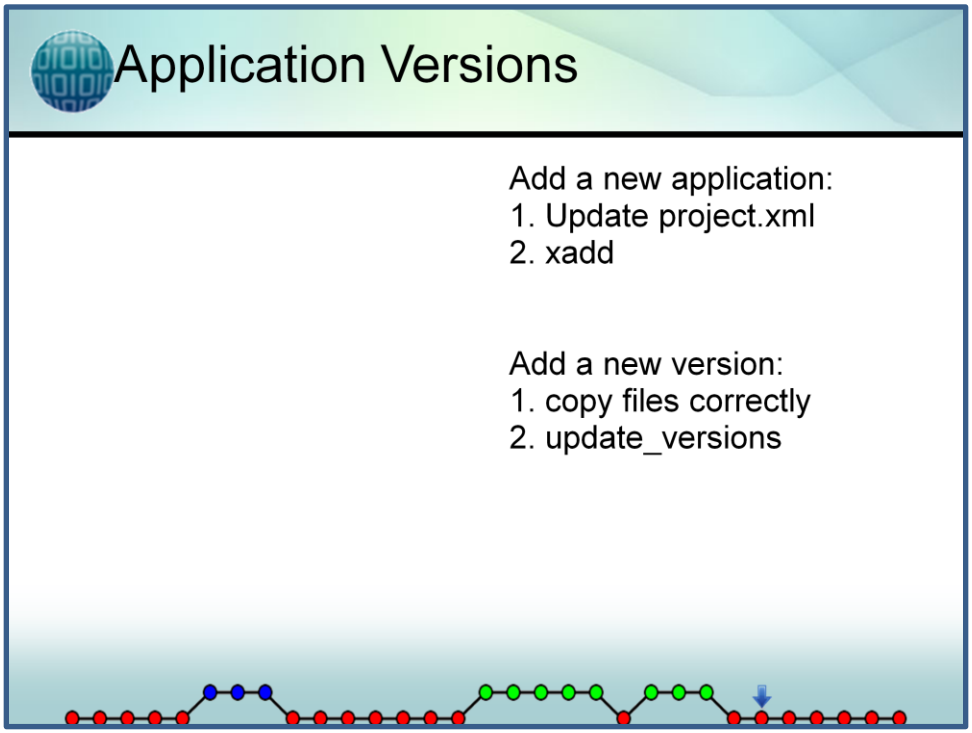
* not actually john the ripper's logo, but a google image result for it, that I think looks awesome.

Rewiring John into a BOINC App

```
+int main(int argc, char **argv) {
+       int status = boinc_init();
+       boinc_resolve_filename("john.conf", confFile, sizeof(confFile) );
+       boinc_resolve_filename("passwordlist", passlist, sizeof(passlist) );
+
+       int i, newArgc = 2, hasWordlist = 0;
+       for(i=1; i < argc; i++) {
+           newArgc++;
+           hasWordlist = strstr(argv[i], "<<WORDLIST>>") ? i : hasWordlist; }
+       if(hasWordlist) {
+          boinc_resolve_filename("wordlist", wordlistName, 512 );
+          snprintf(wordlistParameter, 612, "--wordlist=%s", wordlistName); }
+
+       newArgv[i=0] = argv[0];
+       for (i++; i<argc; i++) newArgv[i] = i == hasWordlist ? wordlistParameter : argv[i];
+       newArgv[i] = passlist;
+       int ret = john_main(newArgc, newArgv);
+       boinc_finish(ret);
+       return ret;
+}
+int john_main(int argc, char **argv)
+#else
int main(int argc, char **argv)
+#endif
```

*heavily abbreviated and trimmed





Info
- Project.xml http://www.boinc-wiki.info/Project_XML_Document
- http://boinc.berkeley.edu/trac/wiki/XaddTool
- http://boinc.berkeley.edu/trac/wiki/AppVersion
- http://boinc.berkeley.edu/trac/wiki/AppVersionNew
- http://boinc.berkeley.edu/trac/wiki/UpdateVersions
- http://www.boinc-wiki.info/Applications_and_Application_Versions

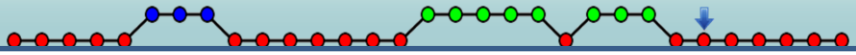
• Lots of links off this article
- http://www.boinc-wiki.info/Adding_Application_Versions

Application Versions

apps/
  name/
    name_version.minor_platform[.ext]

Add a new application:
1. Update project.xml
2. xadd

Add a new version:
1. copy files correctly
2. update_versions

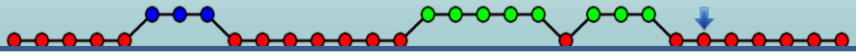


Application Versions

```
apps/
  name/
    name_version.minor_platform[.ext]

  msieve/
    msieve_148.1_linux
```

Add a new application:
1. Update project.xml
2. xadd

Add a new version:
1. copy files correctly
2. update_versions

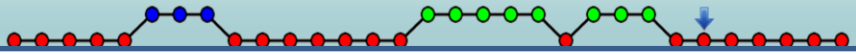


Application Versions

```
apps/
  name/
    name_version.minor_platform[.ext]

  msieve/
    msieve_148.1_linux


  newapp/
    newapp_1.0_linux/
      newapp_1.0_linux
      resourcefile.dat
      somethingelse.db
```

Add a new application:
1. Update project.xml
2. xadd

Add a new version:
1. copy files correctly
2. update_versions

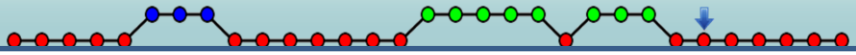


Application Versions

```
apps/
  name/
    name_version.minor_platform[.ext]

  msieve/
    msieve_148.1_linux


  newapp/
    newapp_1.0_linux/
      newapp_1.0_linux
      resourcefile.dat
      somethingelse.db

    new   _1.1_   ux/
      s   fol   r/
       uff.db
```

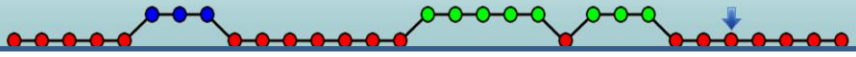
Add a new application:
1. Update project.xml
2. xadd

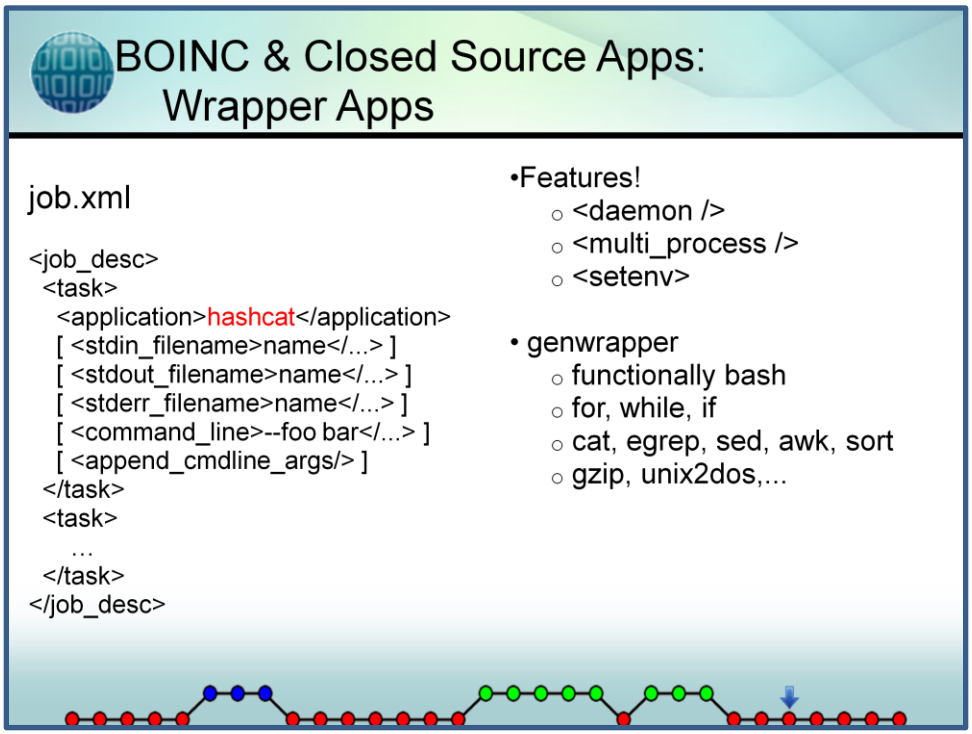
Add a new version:
1. copy files correctly
2. update_versions



Hashcat

hashcat, oclhashcat, oclhashcat+, oclhashcat-lite





Info
- http://boinc.berkeley.edu/trac/wiki/WrapperApp

GenWrapper
- http://genwrapper.sourceforge.net/
- http://sourceforge.net/apps/trac/genwrapper/wiki/manual

Files are immutable, so if you create job.xml – you can never have another job.xml! But the file must be accessible as "job.xml"! Annoying!

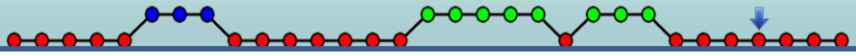
So we use the logical=physical trick:
http://boinc.berkeley.edu/trac/wiki/UpdateVersions#extrainfo

I recommend the format:
job.xml=job.xml-1.0-appname
See sample-server/README

App Plans & GPU Stuff

apps/
  name/
    name_version.minor_platform[.ext]

  msieve/
    msieve_148.1_linux

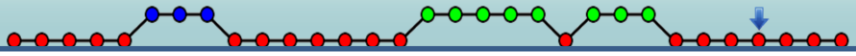


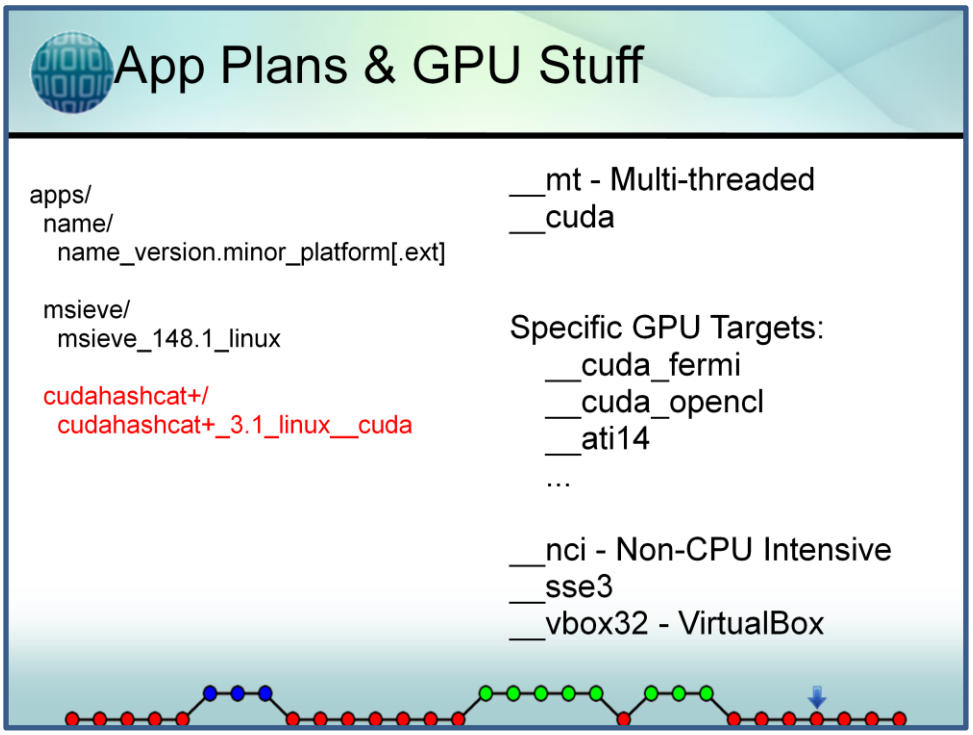
App Plans & GPU Stuff

apps/
  name/
    name_version.minor_platform[.ext]

  msieve/
    msieve_148.1_linux

  cudahashcat+/
    cudahashcat+_3.1_linux__cuda



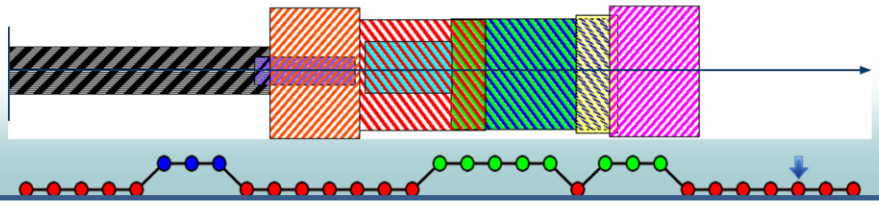


App Plans

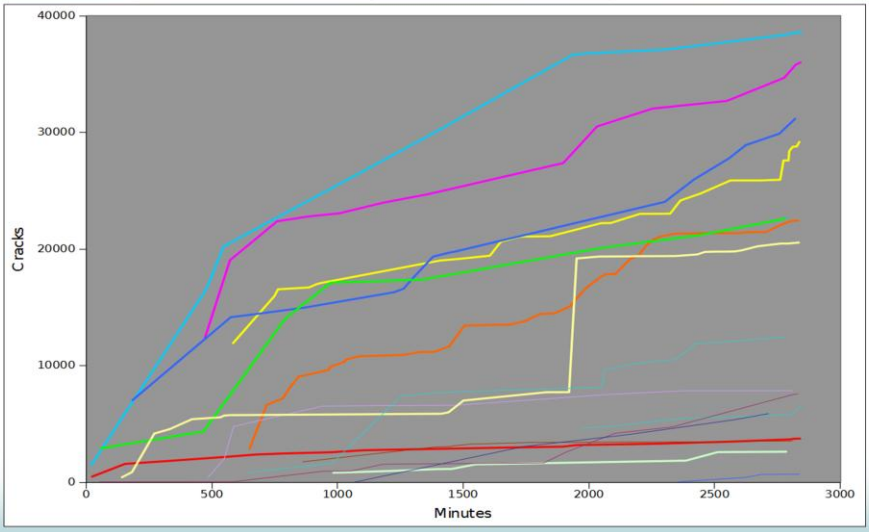
- http://boinc.berkeley.edu/trac/wiki/AppPlan
- GPU: http://boinc.berkeley.edu/trac/wiki/AppCoprocessor
- GPU: http://boinc.berkeley.edu/trac/wiki/GPUApp
- CUDA: http://boinc.berkeley.edu/trac/wiki/CudaApps
- MPI: http://boinc.berkeley.edu/trac/wiki/MpiApps
- http://boinc.berkeley.edu/trac/wiki/AppMultiThread
- Vbox: http://boinc.berkeley.edu/trac/wiki/VirtualBox
- NCI: http://boinc.berkeley.edu/trac/wiki/NonCpuIntensive

My Approach

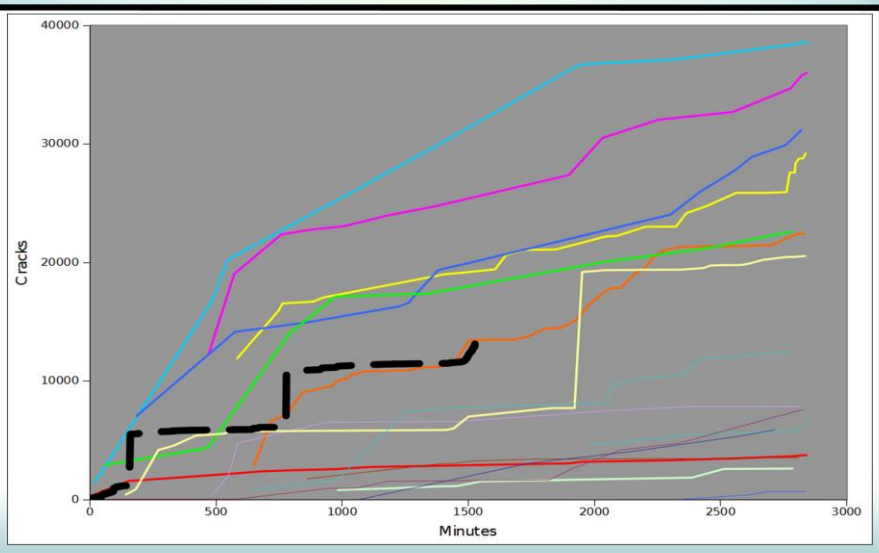
Phase 1: --single
Phase 2: 1 hour incremental
Wordlists
Phase 3: Wordlist Rules
High-Probability Markov Words
Phase 4: Phase 3 Markovs + Rules
Low-Probability Markov Words
Phase 5: Phase 4 Markovs + Rules

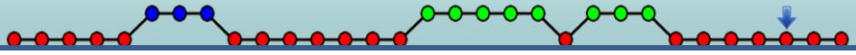


Benchmarking: 2010 Defcon Korelogic Crack Me If You Can Contest

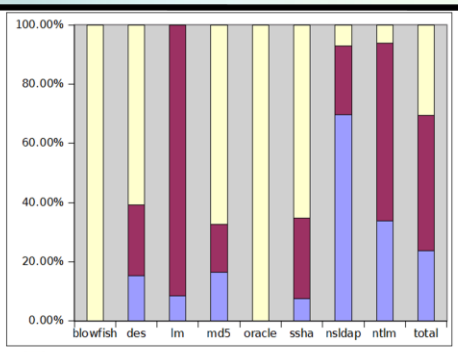


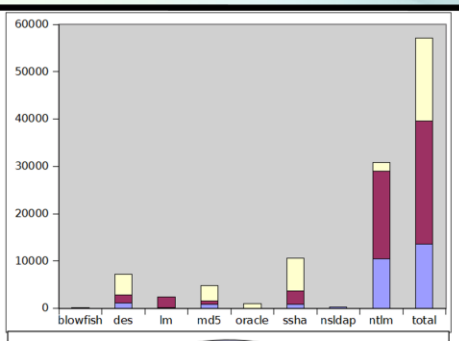
Abject Failure.

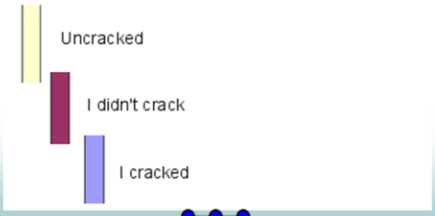


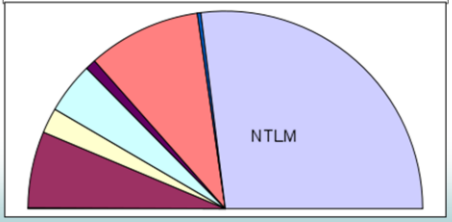


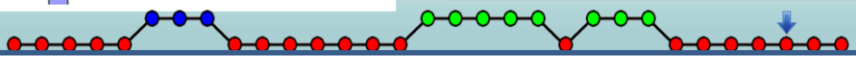
Failure by Hash Type









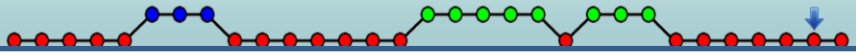


Lessons Learned

- Iterative Cracking

  new pattern -> maskprocessor -> rules -> cracks
  /\ \/
  new plains <- random rules <- new dic
- Automatic mangle rules creation
- Observations from cracked passwords

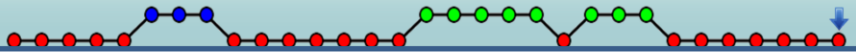
- Cracked password lists
- Actually Crack LM



Back to BOINC

1.Set up a BOINC Server
2.Edit config.xml
3.Lock down the server
4.Set up a client image
5.Set up an application
6.Automate the client image
7.???
8.Profit!

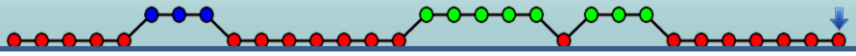
1. Patch source
   or
1. Write job.xml
2. Write input & output templates
3. update_versions
4. Create test workunits
5. Test
6. Repeat 1-6 as needed



Alternatives to BOINC

Password Cracking Only
- Browser Based using Javascript / AJAX / Web Workers
- Durandal http://durandal-project.org/
- Rick Redman of Korelogic's tool

General Architecture
- Amazon Elastic Beanstalk (Java-only)
- Amazon SQS (Write your own wrapper and uploader)
- Bash Scripts/tentakel/multixterm/cssh
- Write your own?
  Will that take more or less than time than configuring BOINC?
  I think more.



Obligatory Ending Slide

Questions?

Thanks:
- GDS
- NYSec
- MersenneForum & jasonp

Tom Ritter
http://ritter.vg
(encrypted mail preferred)

Big Ups To:
- jasonp
- BOINC Devs

http://www.gdssecurity.com/ Hiring: NYC & London
https://github.com/GDSSecurity/cloud-and-control