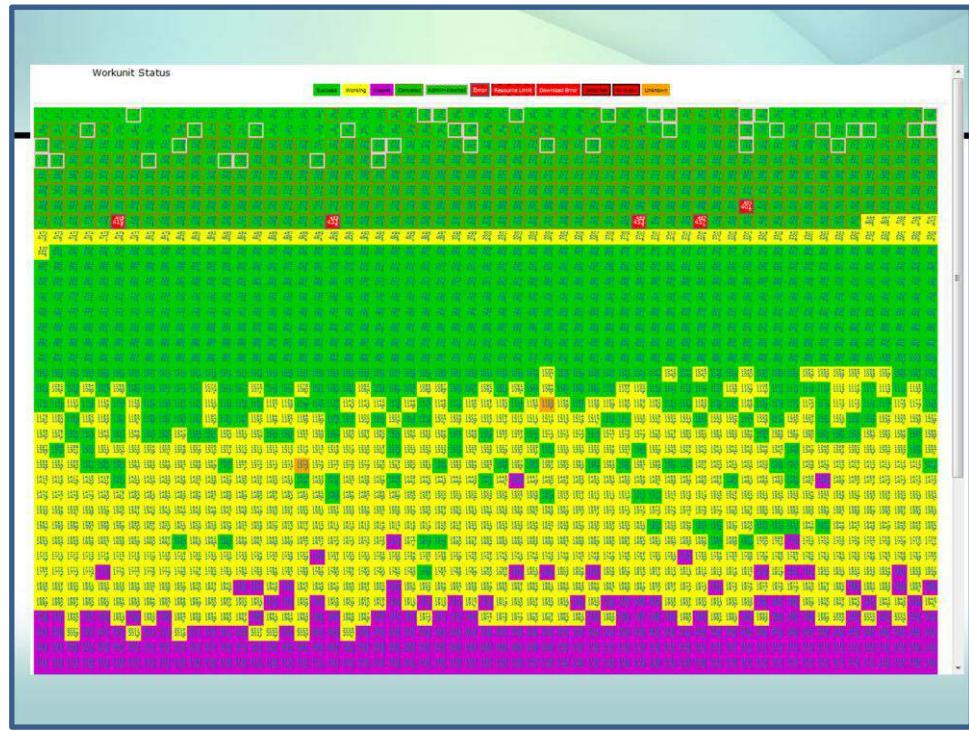




Cloud & Control

Factoring RSA Keys & Cracking Passwords

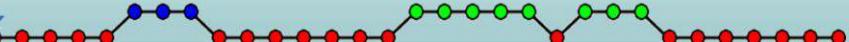
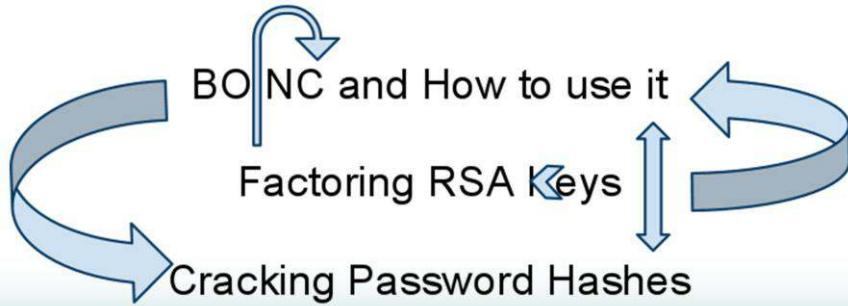




Custom BOINC Dashboard I wrote
(on github)



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



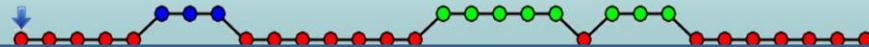
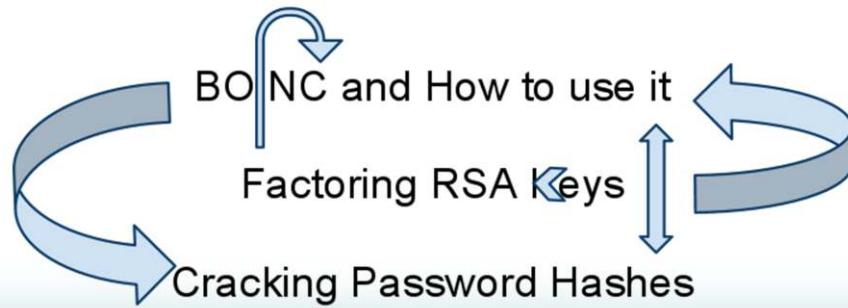


You have interesting problems!

- Fuzzing
- ROP Compilers
- SMT Solving

Would BOINC Help?

How would you fit your problem into BOINC?



Materials!

How Do I Use BOINC?

1. Set up a BOINC Server

2. Edit config.xml
3. Lock down the server
4. Find a host to distribute the work
5. Set up an application
6. Set up a client image
7. Add a client image
8. Create workunits

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 options in 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_Records

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







History of BOINC

1999 - SETI@home launches to the public

2004 - BOINC project begins
- First BOINC Project launches (protein prediction)

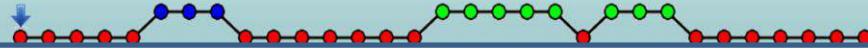
2008 - GPU powered applications introduced

~2 million users

~6 million computers

Top projects (by credit):

- 1 SETI@home
- 2 MilkyWay@home
- 3 Collatz Conjecture
- 32ish SHA-1 Collision Search

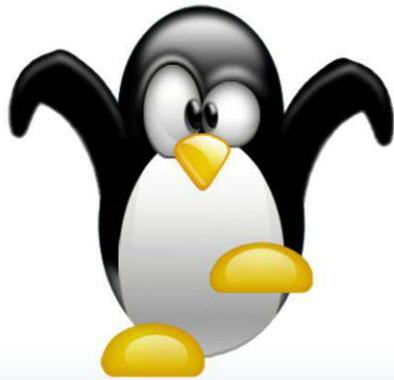


SHA-1 Collision Search

- <http://www.isgtw.org/?pid=1000711>



Why would I use it?



Handles

- network problems
- client errors
- server/client reboots
- file integrity

Supports

- running time limits
- multiple platforms
- untrustable clients
- GPUs and odd applications
- credit/reputation & teams
- assimilation/validation

Platforms BOINC Runs on

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



How Do I Use BOINC?

1. Set up a BOINC Server
2. Edit config.xml
3. Lock down the server
4. Figure out how to distribute the work
5. Set up an application
6. Set up a client image
7. Automate the client image
8. Create workunits



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

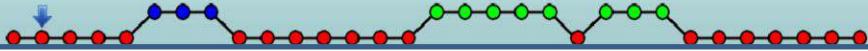
create_work

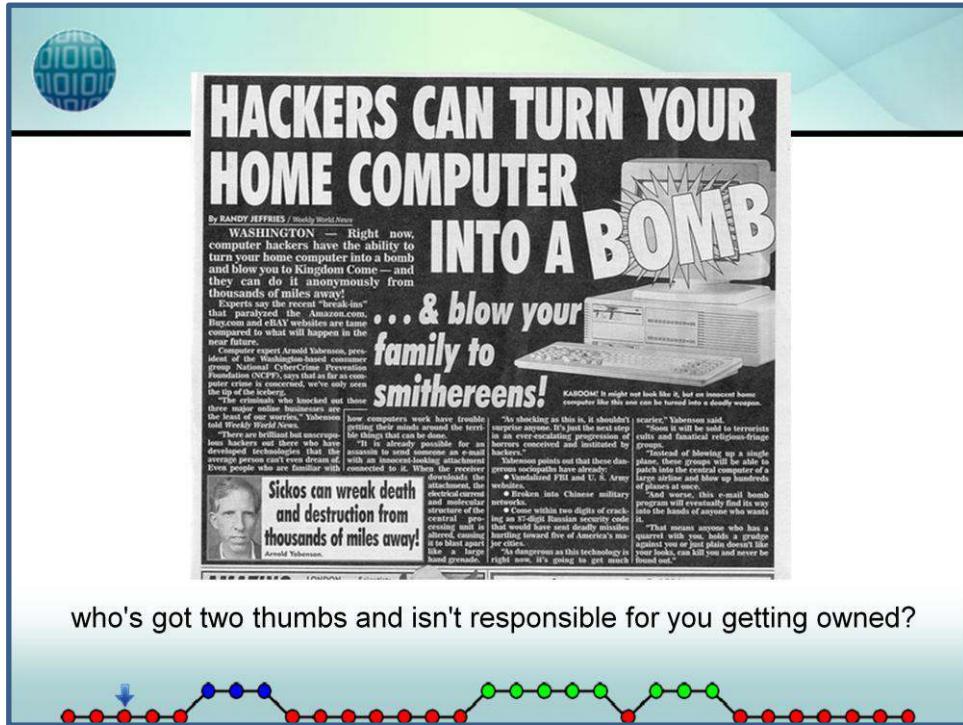
- http://www.boinc-wiki.info/Generating_Work#Creating_Work_Unit_Records



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>



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



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

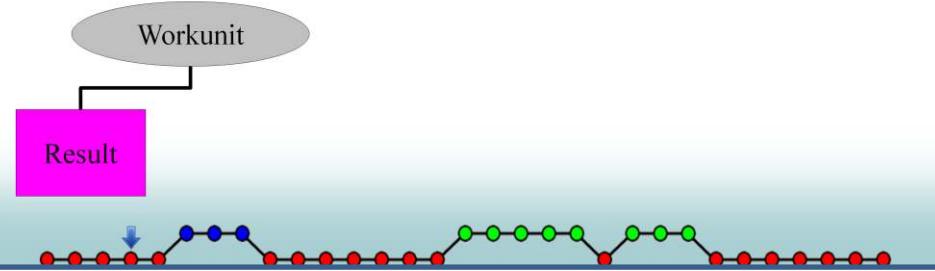
Work Distribution

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



Lifecycle of a unit of work

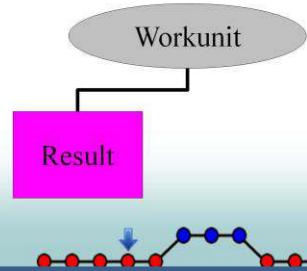
- 1.Create boinc workunits
- 2.Load them into the server
- 3.Server creates 'results'
- 4.Client connects and is assigned 'results'
- 5.Client computes and upload the outcome of the 'result'
- 6.Server Validation
- 7.Server Assimilation
- 8.Server Deletion





Lifecycle of a unit of work

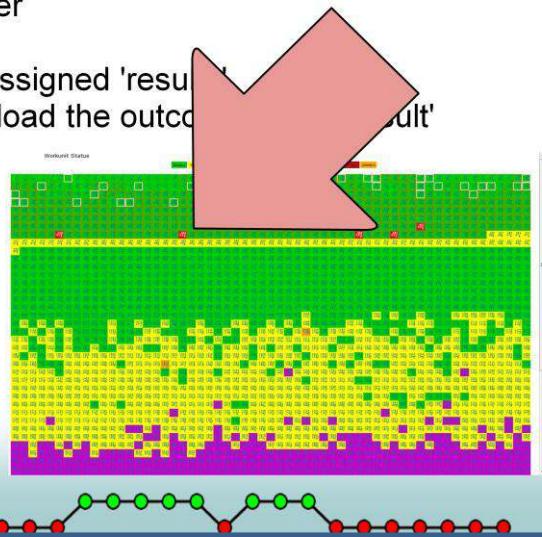
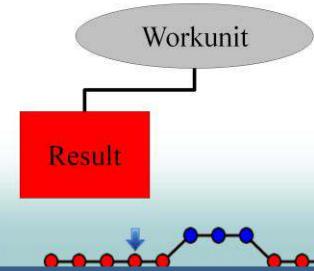
- 1.Create boinc workunits
- 2.Load them into the server
- 3.Server creates 'results'
- 4.Client connects and is assigned 'results'
- 5.Client computes and upload the outcome of the 'result'
- 6.Server Validation
- 7.Server Assimilation
- 8.Server Deletion





Lifecycle of a unit of work

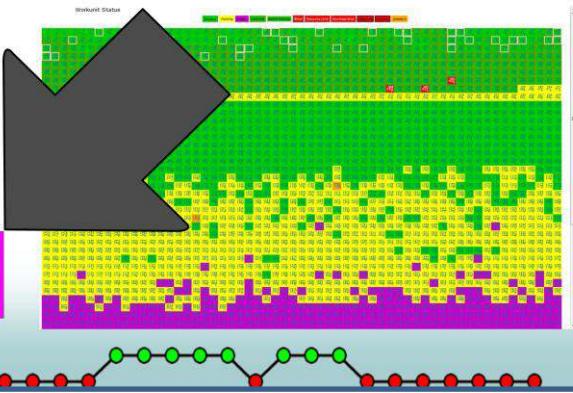
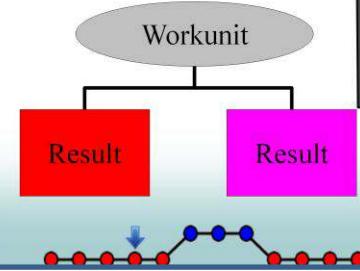
- 1.Create boinc workunits
- 2.Load them into the server
- 3.Server creates 'results'
- 4.Client connects and is assigned 'result'
- 5.Client computes and upload the outcome 'result'
- 6.Server Validation
- 7.Server Assimilation
- 8.Server Deletion





Lifecycle of a unit of work

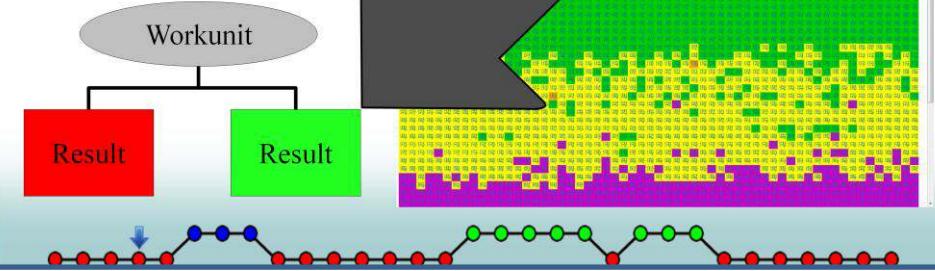
- 1.Create boinc workunits
- 2.Load them into the server
- 3.Server creates 'results'
- 4.Client connects and is assigned 'results'
- 5.Client computes and upload the outcome of the 'result'
- 6.Server Validation
- 7.Server Assimilation
- 8.Server Deletion





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 'results'
- 5.Client computes and upload the outcome of the 'result'
- 6.Server Validation
- 7.Server Assimilation
- 8.Server Deletion





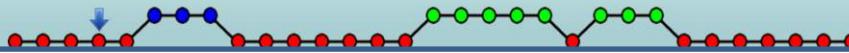
Lifecycle of a unit of work

- 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

Can actually be really complicated!

But for us.... no.

- sample_bitwise_validator
- sample_assimilator



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:

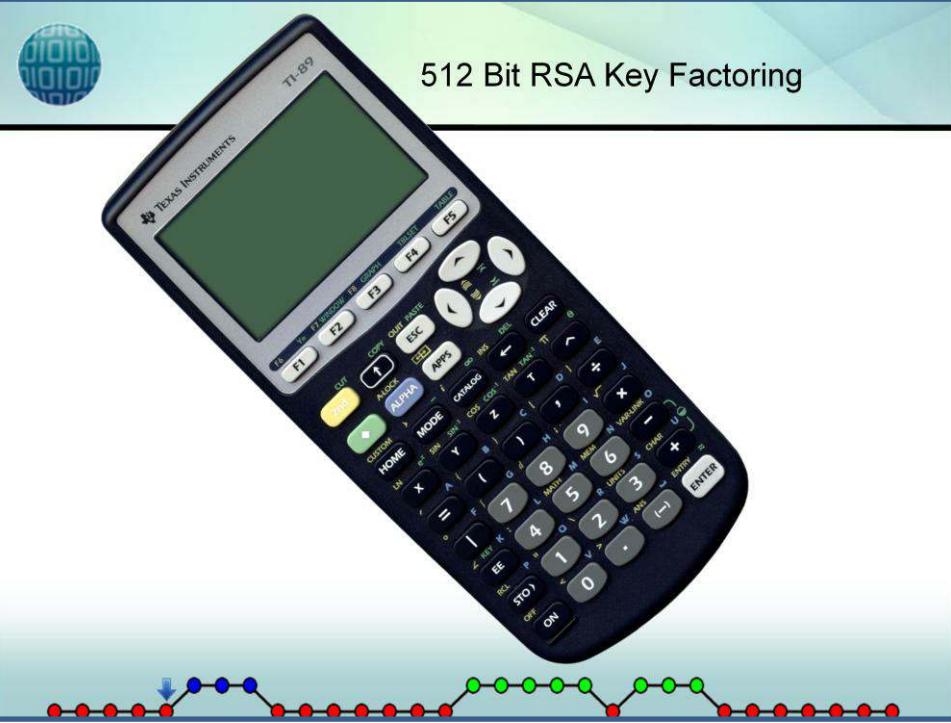
- <http://boinc.berkeley.edu/trac/wiki/ValidationSimple>

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





History

$p * q = n \leftarrow n$ is a semiprime
 $5 * 3 = 15 \leftarrow 15$ is a semiprime
(76-digit p) * (76 digit q) = (155 digit n)

- Aug 1999 - 512 Bit Factored for the first time (publicly)
- 2004 - GGNFS, msieve and factLat.pl in development
- July 2009 - TI83+ Signing Key Factored
- Aug 2009 - Factoring Service Offered: \$5000/key
- Sept 2009 - All TI Signing Keys factored
- Dec 2009 - 768 Bit factored for the first time (publicly)
 $40 + 1500 + 155 = 1695$ Core-Years



<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

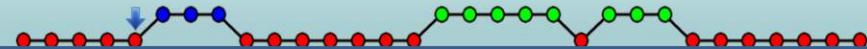
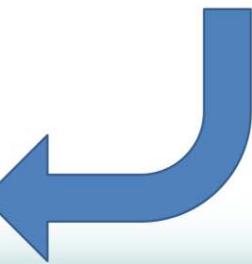
1.Trial Division?

- Is it divisible by 2? 3? 5? 7? 11? 13

2.Pollard Rho

3. ECM

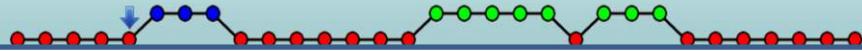
4.General Number Field Sieve





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





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



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





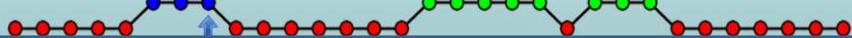
How Do I Factor - GNFS

- 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"





How Do I Factor - GNFS

- 1.Polynomial Selection
- 2.Sieving
- 3.Combine

Slow & Unparallelizable

512 Bit ~8 Core-Days
768 Bit ~155 Core-Years*

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 unparallelizable?

<http://www.mersenneforum.org/showthread.php?t=15361>

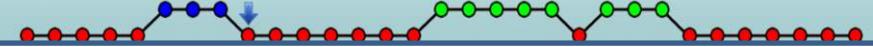
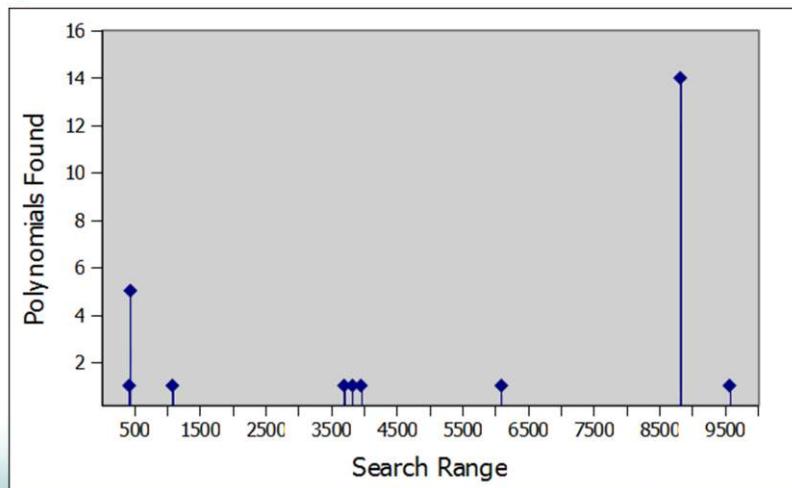
* is because the 768 bit semiprime used Block Weildmann as opposed to msieve's block lanczos algorithm.

<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)

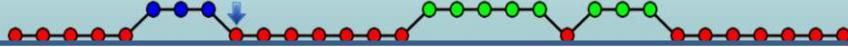
- 3.Via Lattice Sieving

Combine

- 1.Filter Relations & Build Matrix

- 2.Linear Algebra using Lanczos

- 3."Square Root Phase"





msieve by jasonp

jasonp?



Polynomial Selection

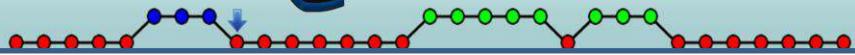
1. $f(x) \& g(x)$ of degree d, e
2. irreducible over rationals
3. interpreted mod n have common root mod m

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"



10

BOINC-ing an Open Source App

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



boinc_init

- <http://boinc.berkeley.edu/trac/wiki/OptionsApi>

BOINC-ing an Open Source App

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

Optional:

 - Checkpointing
 - Critical Sections
 - `boinc_fraction_done`
 - `boinc_need_network`



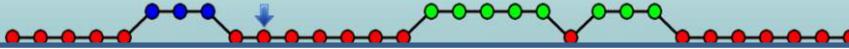
Writing your own application ground-up for BOINC is outside the scope of this talk, as is porting an application 'fully' to BOINC to take advantage of all its options. But you can start here:

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



Rewiring msieve into a BOINC Application

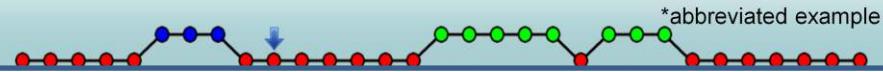
```
@@ -2852,7 +2891,33 @@
+#ifdef HAVE_BOINC
int main(int argc, char **argv)
{
+ int newArgc, ret;
+ char** newArgv;
+ myboincstart(&newArgc, &newArgv, argv[0]);
+ ret = sieve_main(newArgc, newArgv);
+ boinc_finish(ret);
+ return ret;
}
+
+
+int sieve_main(int argc, char **argv)
+#else
+int main(int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

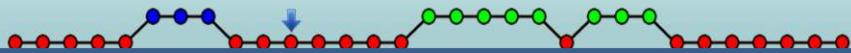
    *argc = 0;
    argv = new char*[7];
    argv[(*argc)++] = name;
    argv[(*argc)++] = "-i";
    argv[(*argc)++] = in;
    argv[(*argc)++] = "-nf";
    argv[(*argc)++] = out;
    argv[(*argc)++] = "-np";
    argv[(*argc)++] = "\0";
}
```





BOINC-ing an Open Source App

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



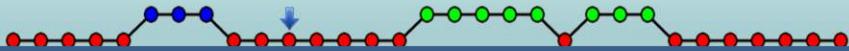
BOINC-ing an Open Source App



- `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



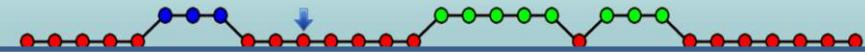
Application Templates

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>
```

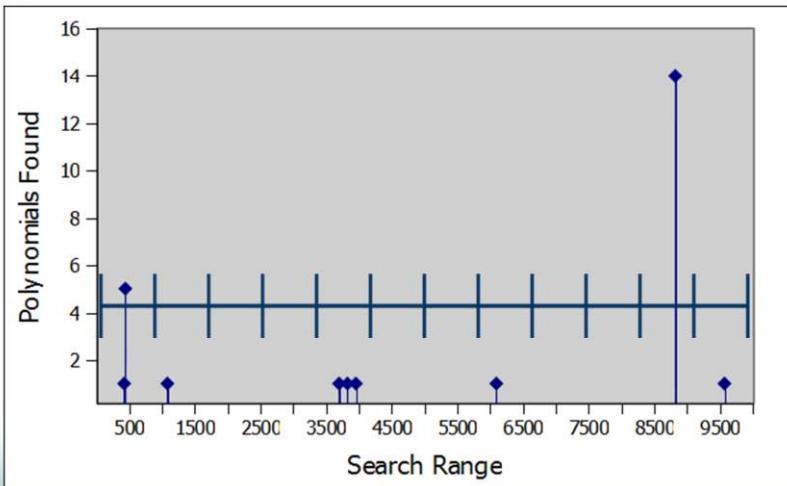
Output

```
<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>
```



How Do I Factor

1. Polynomial Selection



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.

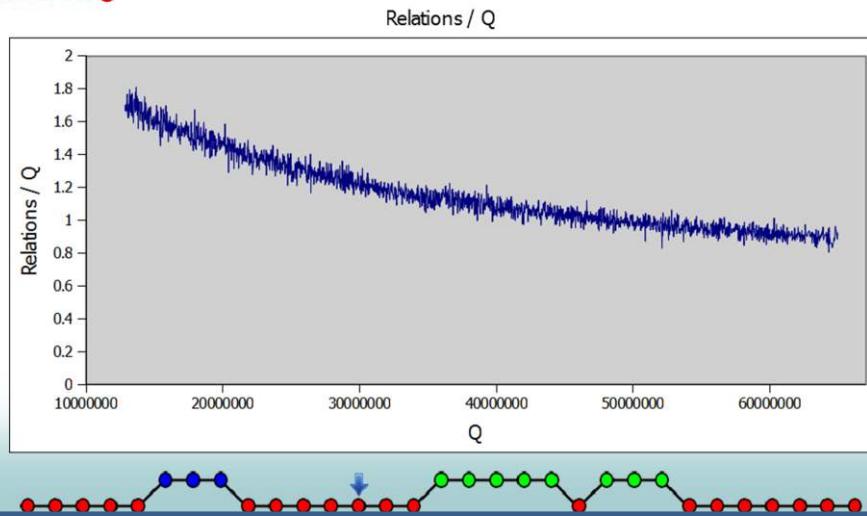
See <http://mersenneforum.org/showthread.php?t=15725>



How Do I Factor

1. Polynomial Selection

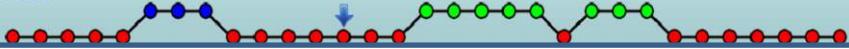
2. Sieving





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)
#endif
int main(int argc, char **argv)
#endif
```

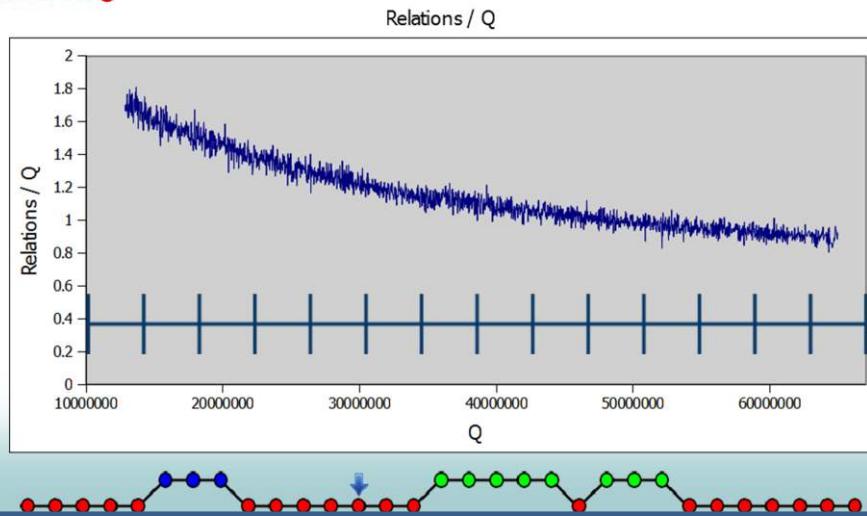




How Do I Factor

1. Polynomial Selection

2. Sieving

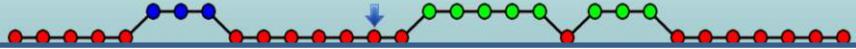
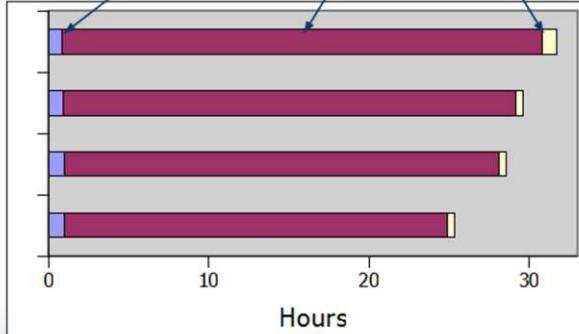




How Do I Factor

- 1.Polynomial Selection
- 2.Sieving
- 3.Combine

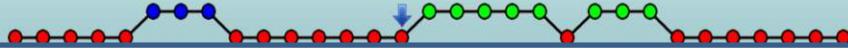
Relation Filtering & Matrix Construction
Linear Algebra
Square Root





The payoff

```
$ wget -q https://www.eff.org/files/syrian-facebook-attack.pem
$ openssl x509 -noout -modulus -in syrian-facebook-attack.pem
Modulus=D5997DCA6577FCD964FE316987BDED93BA4D9644844629CF26CDA9CC
    EED253AD2EE646EE1CF8AC95D18FA014A2EC29672009BD684F79579A
    AA8D7E73E797F6B3
$ python
>>> n = int('D5997DCA6577FCD964FE316987BDED93BA4D9644844629CF26C
        DA9CCEED253AD2EE646EE1CF8AC95D18FA014A2EC29672009BD
        684F79579AAA8D7E73E797F6B3', 16)
>>> n
1118711751718221848900478534389371078344198941752665493293874665
9182160987488338442802072394008666085971431614387661703466578
380319053521569571009086355123L
>>> p = 1043183271162141235507823571625344077547394249292948691
86089643711662097313899
>>> q = 1072401928447279783171545875406777026254092400582533169
64568310846932737705177
>>> n - (p * q)
0L
```



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

Factoring Details

1 Observations on Factoring Using the GNFS
Cloud & Control

2 Some Details on Factoring
Some Details on Factoring

3 Misconceptions about Polynomials
Misconceptions about Polynomials

4 Misconceptions about Polynomials
Slide 4

Moved to their own slide deck for time/relevance

Available on github.

(Sorry!)

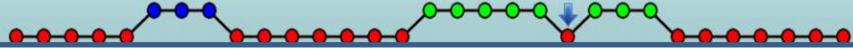


So Far

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

Next

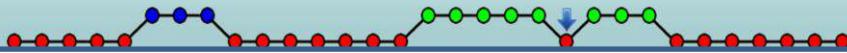
- BOINC
 - Close Source Applications
 - GPU Applications
- Hands-off Cracking Passwords





Cracking

```
xshdofysw784cnquikngcoh43unbkogjpb1kfdmgoifdgnm  
nsdiyuofuycbrihmhodrtus5vhlfdgmuoyhgiovdmugosho  
lb43gjungotfsmcohsu4ohnuihti5vtrtugreuggbik  
uy1gsfbuug4tq5fjuughf169hgugvegttrvcs418hc7  
atv7icqirhc1acschr17denctgibc48  
zhgd8oiz8s3cyn87fzs74chbz1u64yhawfihoc  
ngr13wchgikchqkserv731jh48t47sdift  
78c5yu1nbguzrheurgicgr  
guea623vab2xixnuam33khgthgkdsuh  
zxifyud1u6739dhcrdsik  
hbgk64389b2bikcjmuhiu6zsui3kj8oyd  
07eh7i1v74xf7icjghcf  
aiggy3dyud2uaayezs37y  
iekyhfc3o8wbrelkjmfsl  
nc798hs0cuo9jughgniom  
jocx7w9465fgisuyfsciam  
jkshgsuuisglassfyidfeoa  
iubcoac3yhrrocniouity  
90hvwpaoity7iy475yt1wo  
iusauitizsgucjyuitisgt4phncu4is  
egsauigf648cnglog24891gx4e  
javbwoonity878ncirusdfisuavm...  
395y892eggrefiusghgewichngfwqucyenbytwiths  
3397675n1vrgs1ggrincgeflemkxg1csgugtriuuuuu  
1so587hsigf46fgtynotaygonchfwnaaiginviathote  
78wauZasha9sB9uuv9nuvew98n9nnnuu93uukivwags
```





How do you Parallelize Cracking?

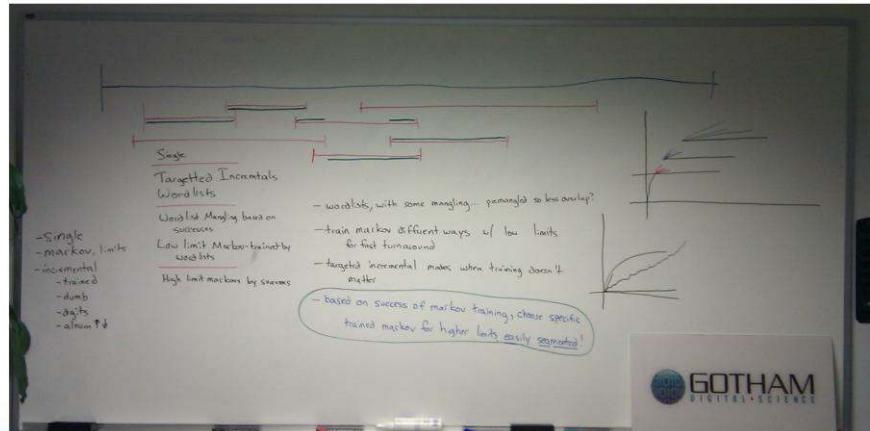
- john
 - Several MPI Patches for john - but only on clusters
 - Mode:External - but bad overhead when splitting
 - Cheap Hacks (bad idea)
- hashcat family (hashcat, oclHashcat, cudaHashcat)
 - Not much you can do (closed source)



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



Enter the Magic





All Possible Passwords





Brute Force



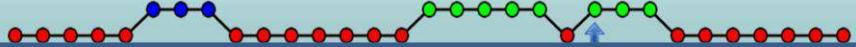


Wordlist



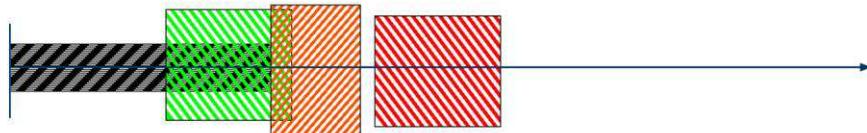


Two Wordlists!



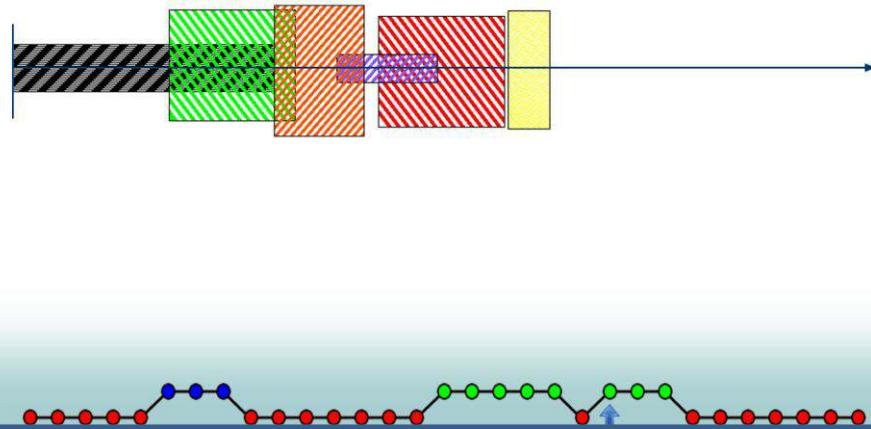


Let me try this....



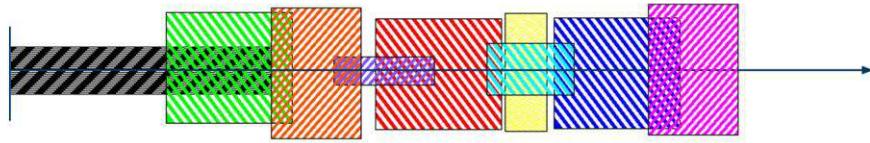


Foreign Wordlists!



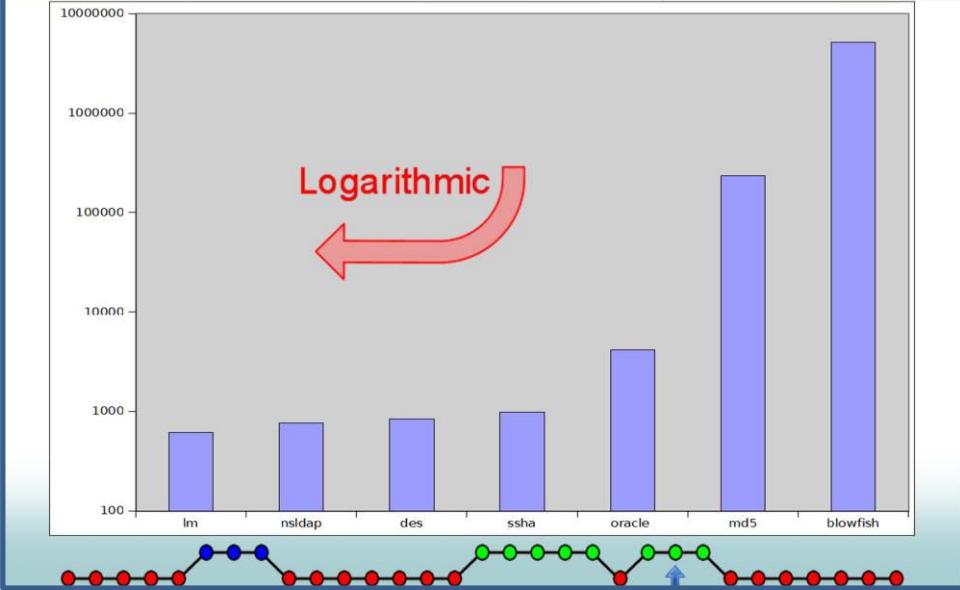


Kitchen Sink!





Not all hashes are created equal

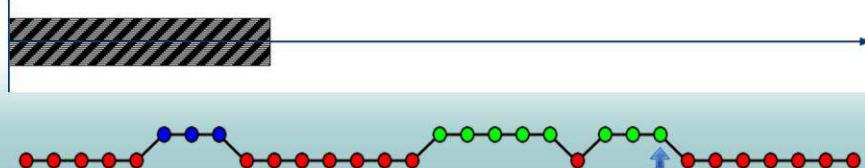




My Approach

Phase 1: --single

Phase 2: 1 hour brute force

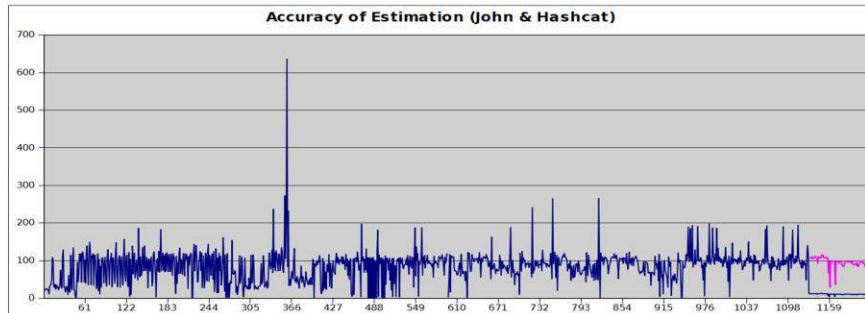




My Approach

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.

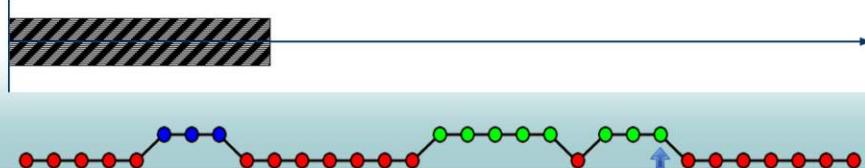


My Approach

Phase 1: --single

Phase 2: 1 hour brute force

Large Wordlist (750 Words)





My Approach

Phase 1: --single

Phase 2: 1 hour brute force

Large Wordlist (750 Words)

Lines 1-250

Lines 250-500

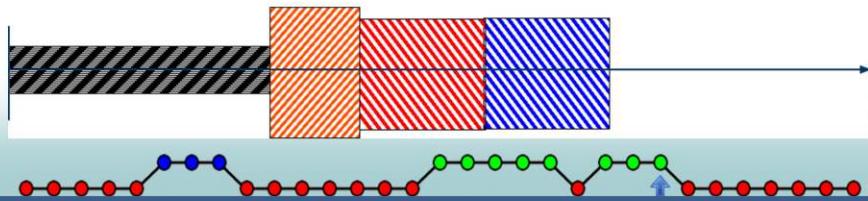
Lines 500-750





My Approach

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



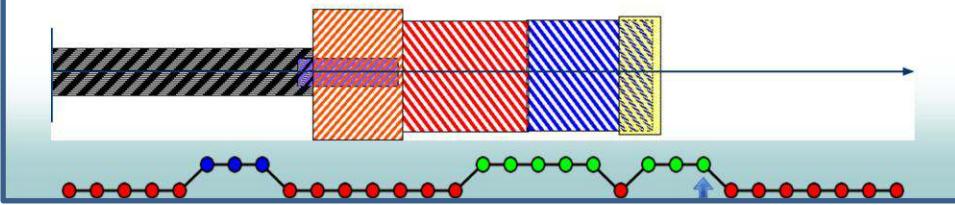


My Approach

Phase 1: --single

Phase 2: 1 hour brute force
Wordlists

Phase 3: Wordlist Rules
High-Probability Markov Words





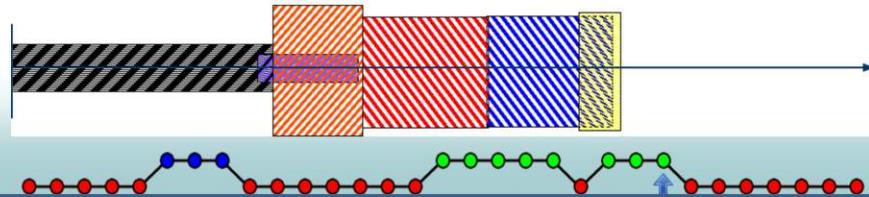
My Approach

Phase 1: --single

Phase 2: 1 hour brute force
Wordlists

Very carefully pruned
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

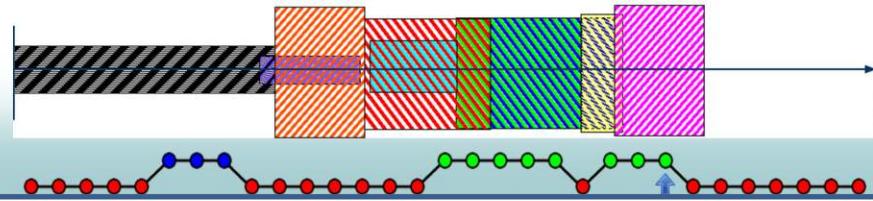
Phase 1: --single

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



John the Ripper



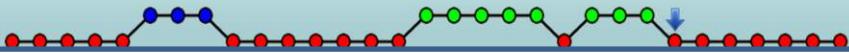
* not actually john the ripper's logo, but a google image result for it, that I think looks awesome.



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





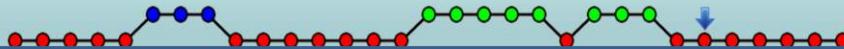
Application Versions

Add a new application:

1. Update project.xml
2. xadd

Add a new version:

1. copy files correctly
2. update_versions



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

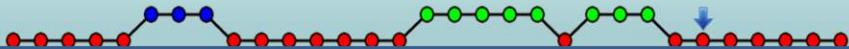
apps/
name/
name_version.minor_platform[.ext]

Add a new application:

1. Update project.xml
2. xadd

Add a new version:

1. copy files correctly
2. update_versions





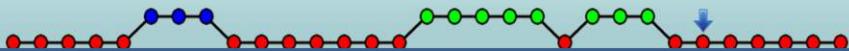
Application Versions

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
```

newapp_1.1_linux/
 subfolder/
 stuff.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





BOINC & Closed Source Apps: Wrapper Apps

job.xml

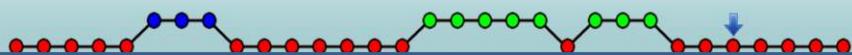
```
<job_desc>
<task>
  <application>hashcat</application>
  [ <stdin_filename>name</...> ]
  [ <stdout_filename>name</...> ]
  [ <stderr_filename>name</...> ]
  [ <command_line>--foo bar</...> ]
  [ <append_cmdline_args/> ]
</task>
<task>
  ...
</task>
</job_desc>
```

- Features!

- <daemon />
- <multi_process />
- <setenv>

- genwrapper

- functionally bash
- for, while, if
- cat, egrep, sed, awk, sort
- gzip, unix2dos,...



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

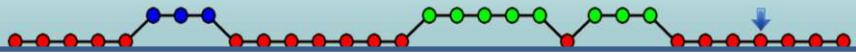
See sample-server/README



App Plans & GPU Stuff

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

```
msieve/  
  msieve_148.1_linux
```



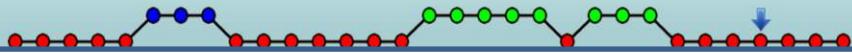


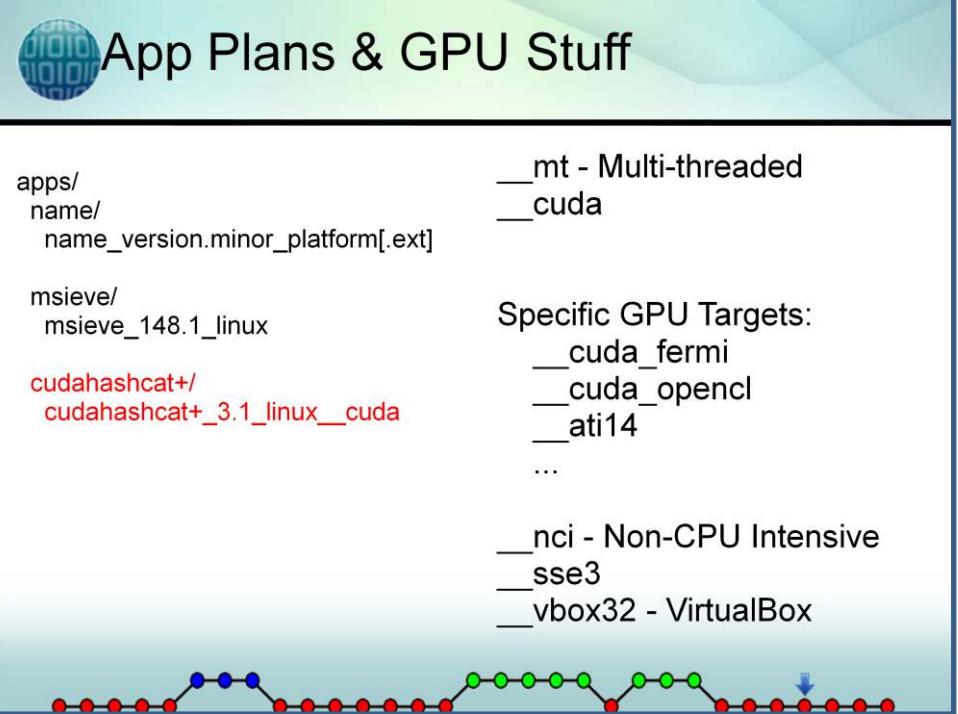
App Plans & GPU Stuff

```
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

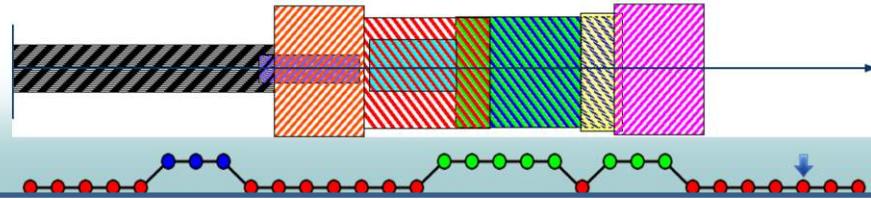
Phase 2: 1 hour incremental
Wordlists

Phase 3: Wordlist Rules

High-Probability Markov Words

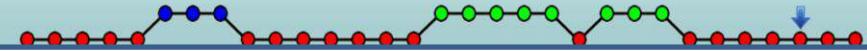
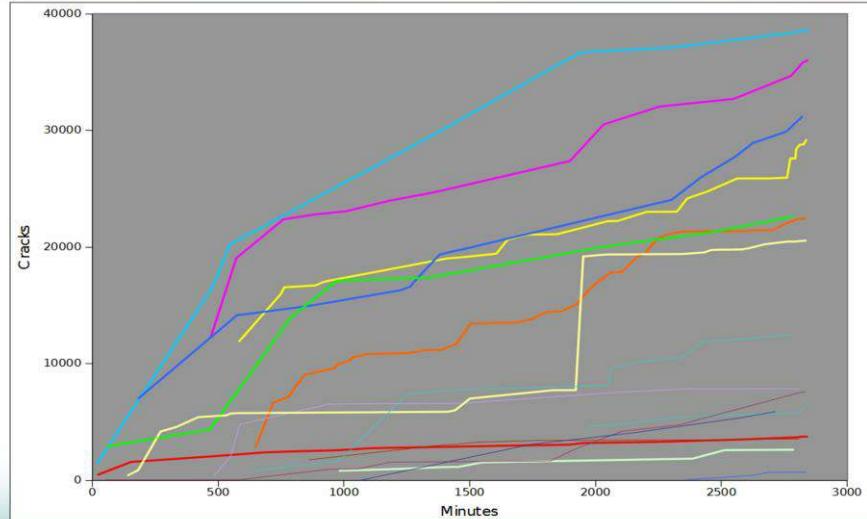
Phase 4: Phase 3 Markovs + Rules
Low-Probability Markov Words

Phase 5: Phase 4 Markovs + Rules



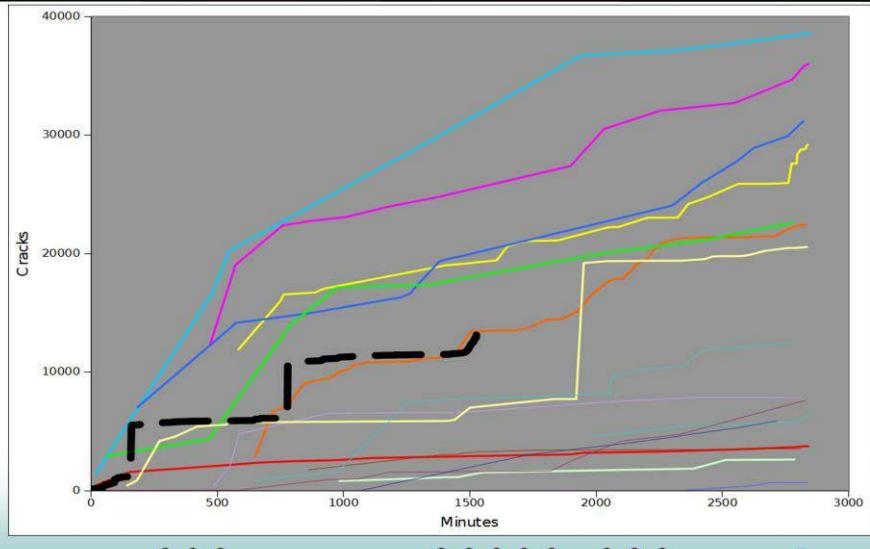


Benchmarking: 2010 Defcon Korelogic Crack Me If You Can Contest

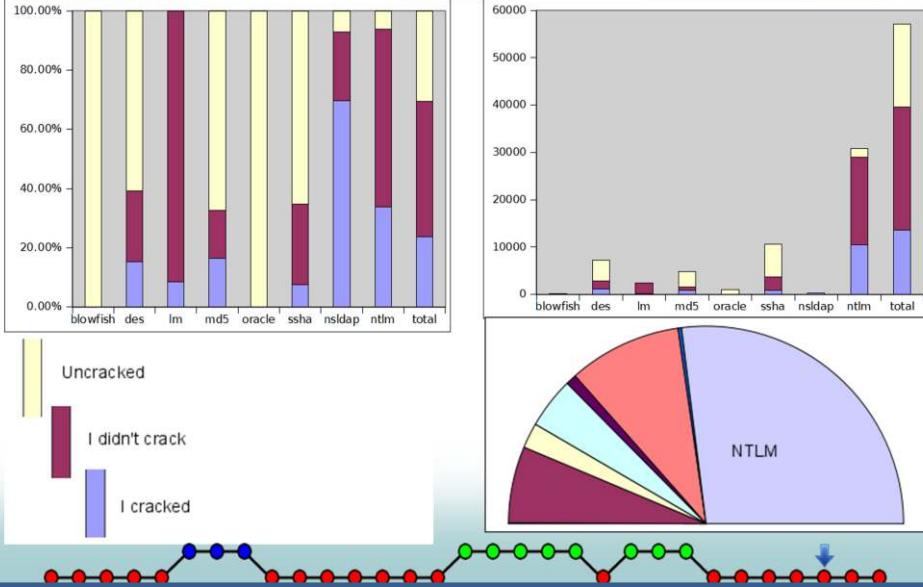




Abject Failure.



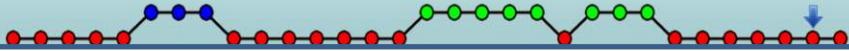
Failure by Hash Type





Lessons Learned

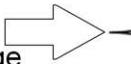
- Iterative Cracking
 - new pattern -> maskprocessor -> rules -> cracks
 - ^
 - V
 - new plains <- random rules <- new dic
- Automatic mangle rules creation
- Observations from cracked passwords
- Cracked password lists
- Actually Crack LM



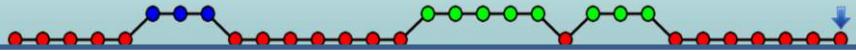


Back to BOINC

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

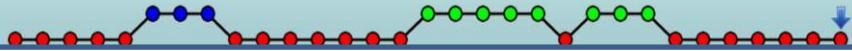
- Browser Based using Javascript / AJAX / Web Workers
 - Durandal <http://durandal-project.org/>
 - Rick Redman of Korelogic's tool

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>