

CrypTool Number Field Sieve Extensions

Theodore Winograd

May 12 2008 / ECE746

What is CrypTool?

What is CrypTool?

- Educational software that aims to assist in learning about cryptography
- Provides tools for encryption, decryption, MACs, signatures, etc...
- Provides tools for performing cryptanalysis

CrypTool Needs

CrypTool lacks cryptanalysis tools for RSA.

Extending CrypTool to support NFS has many benefits:

- Students can see first hand how the security of RSA depends on the difficulty of factoring large numbers
- Students can see how NFS parameters affect factoring time
- Students can feasibly break small RSA keys generated in CrypTool

What is NFS?

NFS is the fastest known algorithm for factoring large numbers.
It is composed of five steps:

- Polynomial selection
- Sieving step
- Mini-factoring step
- Matrix step
- Square root step

Polynomial Selection

In polynomial selection, we strive to produce a polynomial, f such that:

$$f(x) \equiv 0 \pmod{n} \quad (1)$$

Murphy and Kleinjung have provided some heuristics. Some common features are:

- 4th or 5th degree polynomials
- The leading coefficient is 1
- Trial sieving is required

Sieving

- There are two families of sieving operations
 - Lattice
 - Line
- Sieving must have factor bases provided
 - *Rational Factor Base* - prime integers up to a certain bound
 - *Algebraic Factor Base* - roots of $f(x)$ modulo p for prime integers up to a certain bound

Line Sieving

- There are two primary parameters:
 - A such that $-A < a < A$
 - B_0 and B_1 such that $B_0 < b < B_1$
- Each relation (a, b) is saved if:
 - $a + bm$ is smooth over the RFB
 - $a + b\theta$ is smooth over the AFB
- Smoothness?
 - *All primes occurring in the unique factorization are members of the specified set*

Mini-Factoring

- We must factor numbers to produce the matrix.
- Each column represents an (a, b) pair
 - First entry: the sign of $a + bm$
 - Factorization of $a + bm$ over RFB
 - Factorization of the “norm” of $a + b\theta$ over AFB

Matrix and Square Root Steps

- Matrix step:
 - Find a dependency among binary vectors corresponding to (a, b) pairs
 - Dependencies imply $a + bm$ pairs that result in a square in \mathbb{Z}
 - Usually via Lanczos or Wiedemann algorithms
- Square Root Step:
 - $\sqrt{\prod_{(a,b)}(a + bm)}$ and $\sqrt{\prod_{(a,b)}(a + b\theta)}$
 - Congruent squares method:
 - If $x^2 \equiv y^2 \pmod{n}$
 - p, q are found via $\gcd(n, x \pm y)$

What work has been done?

In 2007, I evaluated available NFS implementations:

- Chris Monico: GNU General Number Field Sieve (GGNFS)
- Per Leslie Jensen: Pleslie's General Number Field Sieve (pGNFS)
- Chris Card: factor-by-gnfs
- Jason Papadopoulos: msieve

Msieve is the most efficient—and freely licensed implementation.

GGNFS is a close second—and recommended by the Msieve author and users groups.

Msieve

Msieve is developed by Jason Papadopoulos and a group of volunteers. It is available in the public domain (no license restrictions).

- Latest release: April 14, 2008
- Used by NFSNet to perform the final steps of NFS
- Together with GGNFS, has factored a 518-bit number

GGNFS

Originally developed by Chris Monico.

Up until a few months ago, GGNFS was abandonware. Jasonp (from msieve) has restarted the project with the following goals:

- Integrate msieve and GGNFS
- Remove much of msieve's NFS code
- Replace GGNFS matrix code with msieve's

Features

Msieve:

- Murphy's α approximation for polynomial selection
- Bucket sort line sieve algorithm

GGNFS:

- Murphy's and Franke/Kleinjung polynomial selection
- Classical line sieve
- Lattice sieve

Both:

- Lanczos Matrix step

Build Environment

- CrypTool is written in C++ for MS Visual C++ .NET 2003 and Perl
- Msieve is cross-platform (MS VC++ 2005)
- GGNFS is written for UNIX-like systems, but supports MS VC++
- Early testing was performed on ENIGMA

Software Requirements

This project will require some additional software:

- CrypTool
 - Visual C++ 2003
- Msieve
 - Visual C++ 2005
- GGNFS
 - GMP 4.1.4
 - Visual C++ 2003, 2005, 2008
- CVS
- MAGMA

NFS Parameters

General Parameters	
n	The number to factor
$f(x)$	The polynomial NFS will use
m	A value of x at which $f(x) = 0$
P_{max}	The largest prime to be used in each factor base
RFB	The size of the rational factor base
RFB_{max}	The largest value of the rational factor base
AFB	The size of the algebraic factor base
AFB_{max}	The largest value of the algebraic factor base
QCB	The size of the quadratic character base (optional)

NFS Parameters Cont'd...

Polynomial Selection Parameters	
α	The maximum allowed value of Murphy's α approximation in selecting the polynomial
d	The degree of the polynomial (usually 3 or 5)
Line Sieving Parameters	
A	For line sieving, the sieving interval such that $-A < a < A$

GGNFS Parameters

Table: Polynomial Selection Parameters

n	The number to factor
d	The desired degree of the polynomial
j_0 and j_1	Murphy's sieving-for-root properties
$maxs1$	Threshold for the initial polynomial test
$maxskew$	The maximum acceptable skew for the polynomial
$lc1$	The maximum relative size of the leading coefficient
lcp	The number of primes to choose from for leading coefficient divisors
$leave$	The number of bits of the leading coefficient that will be random

GGNFS Parameters Cont'd...

Table: Factor Base Parameters

n	The number to factor
f	The polynomial chosen for NFS
m	The value at which $f(m) = 0 \pmod n$
rl	The maximum size of the rational factor base
al	The maximum size of the algebraic factor base
mpr	The max large rational prime for large-prime variation
mpa	The max large algebraic prime for large-prime variation
p	The number of large rational and algebraic primes

GGNFS Parameters Cont'd...

Table: Sieving Parameters

n	The number to factor
f	The polynomial chosen for NFS
m	The value at which $f(m) = 0 \pmod n$
$(r a)lambda$	How far from perfect sieve values to look for good relations.
q_0	The initial value of q for the lattice sieve
$qintsiz$	The q range size for the lattice sieve
a_0	The initial value of a for the line sieve
a_1	The final value of a for the line sieve
b_0	The initial value of b for the line sieve
b_1	The final value of b for the line sieve

GGNFS Parameters Cont'd...

Table: Matrix Parameters

<i>minff</i>	The minimum number of FFs
<i>maxrelsinf</i>	The maximum relation-set weight
<i>wt</i>	The weight factor determining the sparseness of the matrix.

GGNFS Parameters Cont'd...

Table: Square Root Parameters

<i>depnum</i>	The specific dependency to try from the matrix solution
<i>knowndiv</i>	The product of known small divisors of n

Intermediate Files

Msieve and GGNFS have two sets of outputs:

- Polynomial
- Sieving

This provides several advantages:

- Re-use of polynomials
- Re-use of existing relations
- Memory benefits

Previous Work

- Izu, et al.: CAIRN - FPGA Implementation of the Sieving step
- Shamir: TWIRL - Hybrid Implementation of NFS
- Gaj, et al.: "Implementing the Elliptic Curve Method of Factoring in Reconfigurable Hardware" - Mini-factoring
- Anand: "Factoring of Large Numbers using Number Field Sieve - The Matrix Step"
- CWI and Lenstra - successful factorization of several RSA challenge numbers
- Open Source projects (GGNFS, Msieve, et al.)

Implementation

This project was divided into two parts:

- Implementation - integrate GGNFS and CrypTool
- Experiments - measure the effects of NFS parameters

Implementation

Msieve was originally chosen as the NFS implementation to merge with CrypTool.

- Supports Win32 compilation out-of-the-box
- Implemented as a library that CrypTool could use

Msieve integration would require several steps:

- Convert Msieve to VC++ 2003
- Add CrypTool interface

Modified Implementation Plan

As shown in results, msieve integration was unsuccessful. To use GGNFS instead, the following steps would be required:

- Modify the GGNFS VC++ 2003 project file to compile and run
- Build GMP 4.1.4 in Win32
- Replace CrypTool's GMP implementation with GMP 4.1.4
- Allow CrypTool and GGNFS to link
- Add CrypTool interface
- Add GGNFS code to CrypTool

Experiments

There are a number of things we can test with a NFS implementation:

- How does the choice of polynomial selection parameters affect the amount of time spent on polynomial selection step versus other steps?
- How do *AFB* and *RFB* affect the amount of time spent on each step?
- How does each parameter affect the total amount of time?
- How well does this implementation handle Special NFS numbers?

Experiments Cont'd...

All experiments were run on Enigma.

- Owned and operated by GMU
- Quad Xeon 2.8 GHz
- 4 GB RAM
- Several hundred GB of hard disk space

Experiments Cont'd...

All experiments followed the following steps (for 40-bit and 80-bit)

- Generate multiple polynomials for each value of n with different parameters
- Generate multiple sieving runs for each polynomial with different parameters
- Generate multiple matrices for each sieving run with different parameters

The results will be tabulated and compared to outline their effect on performance.

Msieve

Msieve introduced the following requirements:

- CrypTool *must* be compiled using VS2003
- msieve relies on capabilities that exist only in VS2005
- Unexpected parameter values cause msieve to behave erratically
- msieve would not successfully factor smaller numbers (<98 bits)

Attempts to address these issues proved fruitless:

- Port msieve to VS2005 (partially successful)
- Test msieve on Enigma (unsuccessful)

GGNFS

GGNFS integration proved more successful than Msieve:

- Successfully downloaded, installed and compiled GGNFS
- Successfully modified CrypTool to generate GGNFS input files
- Polynomial selection supported
- Factor base generation supported
- Line sieve supported
- To do:
 - Lattice sieving
 - Relations processing (mini-factoring)
 - Matrix generation
 - Matrix solution
 - Square root step

GGNFS

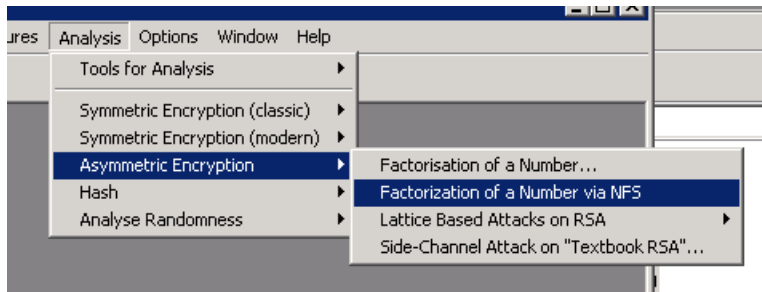
Has the switch to GGNFS been successful?

- CrypTool compiles and runs with the updated GMP version
- Project files were successfully modified to build and run
- CrypTool links with ggnfs.lib

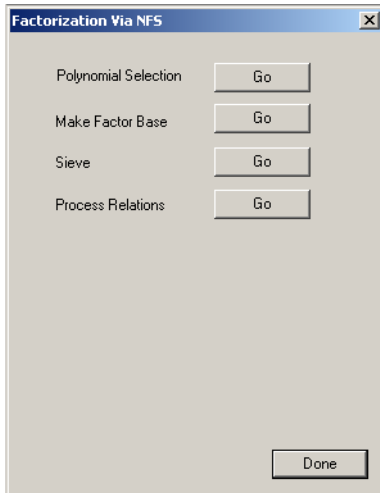
Name	Size	Type	Date Modified
last_spq0	1 KB	LAST_SPQ0 File	4/6/2008 12:39 PM
all.poly	44 KB	POLY File	4/5/2008 6:02 PM
autogplot.sh	2 KB	SH File	3/27/2008 6:33 PM
best.poly	1 KB	POLY File	4/5/2008 7:43 PM
default.fb	436 KB	FB File	4/1/2008 11:49 PM
def-rm-params.txt	4 KB	Text Document	3/27/2008 6:33 PM
def-par.txt	5 KB	Text Document	3/27/2008 6:33 PM
ggnfs.log	5 KB	Text Document	4/6/2008 1:29 PM
ggnfs-lasieve112e.exe	548 KB	Application	4/2/2008 2:23 PM
ggnfs-lasieve113e.exe	548 KB	Application	4/2/2008 2:23 PM
ggnfs-lasieve114e.exe	548 KB	Application	4/2/2008 2:23 PM
makefb.exe	760 KB	Application	4/2/2008 2:23 PM
matbuild.exe	848 KB	Application	4/2/2008 2:23 PM
matprune.exe	672 KB	Application	4/2/2008 2:23 PM
matsolve.exe	684 KB	Application	4/2/2008 2:23 PM
poS1m0b.exe	488 KB	Application	4/2/2008 2:23 PM
poS1m0n.exe	500 KB	Application	4/2/2008 2:23 PM
poS1opt.exe	436 KB	Application	4/2/2008 2:23 PM
polyselect.exe	792 KB	Application	4/13/2008 5:59 PM
procris.exe	796 KB	Application	4/2/2008 2:23 PM
sieve.exe	772 KB	Application	4/2/2008 2:23 PM
spairs.out	77,480 KB	OUT File	4/6/2008 12:43 PM
sqrt.exe	796 KB	Application	4/2/2008 2:23 PM
tst50.fb	439 KB	FB File	4/5/2008 6:07 PM
tst50.job	1 KB	Task Object	4/5/2008 6:19 PM
tst50.job.afb.0	124 KB	0 File	4/5/2008 6:20 PM
tst50.polel	1 KB	POLSEL File	4/5/2008 7:27 PM

CrypTool Modifications

CrypTool has been successfully modified:



CrypTool Modifications



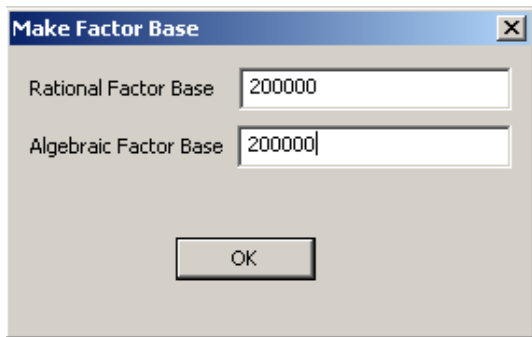
CrypTool Modifications

NFS Polynomial Selection [X]

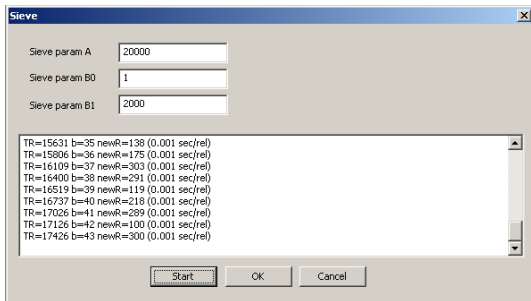
The number to factor	<input type="text" value="583803909215926328117241823630434271"/>
The desired degree of the polynomial	<input type="text" value="4"/>
The maximum skew of the polynomial	<input type="text" value="1500"/>
Murphy's root property j0	<input type="text" value="200"/>
Murphy's root property j1	<input type="text" value="12"/>

Starting polynomial selection...

CrypTool Modifications



CrypTool Modifications



GGNFS vs Msieve

The msieve library failed on every test except those run last semester.

- Ran tests on msieve 1.34
- Ran tests on msieve in Windows, Linux

This is most likely because msieve was designed and tested for large numbers. GGNFS is more robust in this regard.

Demo

Demo

Experimental Results

Default time for 40-digits is 30 minutes. With modifications:

- $n = 11897449189438347511562773578961590477869$
- RFB max: 200000
- AFB max: 250000
- $-20000 < a < 20000$
- $1 < b < 200$
- Factored in 30 seconds

80-digit Results

Table: Results of a 1500 skew polynomial

A_0	A_1	B_0	B_1	<i>Time</i>	<i>Results</i>
$-2 * 10^6$	$2 * 10^6$	1	2000	4m51s	2233/71537
$-2 * 10^6$	$2 * 10^6$	1	20000	33m43s	3949/71537
$-2 * 10^6$	$2 * 10^6$	1	100000	182m15	5748/71537

80-digit Results

Table: Results of a 1625 skew polynomial

A_0	A_1	B_0	B_1	<i>Time</i>	<i>Relts</i>
$-2 * 10^6$	$2 * 10^6$	1	2000	5m47s	2935/71297
$-2 * 10^6$	$2 * 10^6$	1	20000	35m0s	5805/71297
$-2 * 10^6$	$2 * 10^6$	1	100000	185m56s	8829/71297

80-digit results

Table: Results of altering the value of a vs b

A_0	A_1	B_0	B_1	<i>Time</i>	<i>RelS</i>
$-4 * 10^6$	$4 * 10^6$	1	2000	18m0s	5062/71297
$-6 * 10^6$	$6 * 10^6$	1	2000	19m30s	6862/71297
$-8 * 10^6$	$8 * 10^6$	1	2000	26m14s	8529/71297
$-10 * 10^6$	$10 * 10^6$	1	2000	43m21s	10103/71297
$-2 * 10^7$	$2 * 10^7$	1	2000	41m17s	17017/71297
$-4 * 10^7$	$4 * 10^7$	1	4000	138m54s	57503/71297
$-4 * 10^7$	$4 * 10^7$	1	8000	253m97s	71297/71297
$-6 * 10^7$	$6 * 10^7$	1	2000	103m53s	41891/71297
$-8 * 10^7$	$8 * 10^7$	1	2000	130m2s	52969/71297

Future Work

- Improve GGNFS mini-factoring
- Include additional algorithms
- 250-digit experiments
- CrypTool enhancements:
 - Provide users with a range of numbers to choose from
 - Provide users with an estimate of the number of relations necessary
 - Provide users with default values appropriate for the size of the number chosen
 - Integrate NFS with the other factoring algorithms

Questions?

Questions?