

CrypTool Number Field Sieve Extensions

Theodore Winograd

I. INTRODUCTION AND MOTIVATION

CrypTool [9] is educational software that aims to assist in learning about cryptography. CrypTool provides tools for performing encryption, decryption, generating message authentication codes (MACs), digital signatures, and many other cryptographic operations using both modern and historical ciphers. In addition, CrypTool provides some mechanisms for performing cryptanalysis. While CrypTool does provide some tools for attacking RSA (one of the most popular cryptographic algorithms in use), it does not provide support for the most fundamental attack against the algorithm: factorization. This project will improve the CrypTool software by providing support for the Number Field Sieve (NFS), the fastest known algorithm for factoring large numbers.

By extending CrypTool to support NFS, students will be able to see first-hand how the security of RSA depends on the difficulty of factoring large numbers. Similarly, students will also be able to examine how the different NFS parameters can drastically effect the amount of time necessary to factor a large number—anywhere between 15 minutes and two hours for a 50-digit number. Similarly, because students can generate RSA keys of any size in CrypTool, they will be able to feasibly perform attacks against small (less than 100-digit) RSA keys.

NFS is a complex algorithm and it is beyond the scope of this project specification to delve into the details. There are five major steps of NFS: [3]

- Polynomial selection
- Sieving
- Mini-factoring
- Linear algebra
- Square root

Each step of the NFS algorithm has a number of potential algorithms to choose from. In the polynomial step, two common algorithms include Murphy's [17] and Kleinjung's [15] algorithms. In the sieving step, the two primary families of algorithms are the line sieve and the lattice sieve as described by Franke. The mini-factoring step can take advantage of many of the factoring algorithms available, including elliptic curve methods (studied by researchers at GMU) and the quadratic sieve. Finally, alternate algorithms are available for the linear algebra step as well, including the Wiedemann and Lanczos algorithms. Once the first implementation of the CrypTool extension is complete, it is hoped that multiple algorithms will be implemented for each step of NFS, allowing users of CrypTool to gain a full understanding of the benefits and drawbacks of the choices available for each step of NFS.

In Fall 2007, as part of my thesis research, I evaluated a number of different readily available implementations of NFS:

- Chris Monico: GNU General Number Field Sieve (GGNFS) [16]
- Per Leslie Jensen: Pleslie's General Number Field Sieve (pGNFS) [14]
- Chris Card: factor-by-gnfs [6]
- Jason Papadopoulos: msieve [18]

From the results of my investigations, I found msieve to be the most efficient and freely licensed implementation of NFS. In addition, msieve is actively developed, with the most recent release on January 13, 2008. Msieve has been used by the NFSNet project and other researchers as one of the primary tools for factoring numbers as large as 518 bits.

Both msieve and GGNFS rely on Murphy's α approximation to generate polynomials. Both implementations provide line sieves, but msieve's line sieve uses the bucket sort algorithm outlined by Aoki and Ueda.[2] For the linear algebra step, msieve does not explicitly say on which algorithm it relies while GGNFS provides an implementation of the Lanczos algorithm.

II. LANGUAGE, PLATFORM, AND COMPILER

CrypTool is a C++ application written using Microsoft Visual C++ .NET 2003 and the Perl scripting language for the Windows Platform. There is a port of CrypTool for Linux [10] written using C++ and the QT4 graphical user interface (GUI) library. According to the CrypTool Readme, updated in July 2006, claims that Visual C++ 2005 is not supported, which may introduce some difficulties into the project as Visual C++ 2005 is the most readily available version.

Msieve and GGNFS are both written to be cross-platform but were developed on UNIX-like systems. As such, they may not easily compile in a Windows environment. Early testing has been done on the GMU Enigma machine, which is a Linux-based OS. Both GGNFS and msieve function well on this system. Because CrypTool is available both as a Linux and Windows tool, it will be possible to choose the CrypTool implementation to work with based on the feasibility of running msieve and GGNFS on Windows.

III. ADDITIONAL SOFTWARE

According to the CrypTool Readme, CrypTool requires only an Perl installation and Microsoft Visual C++ to successfully compile. To extend CrypTool with NFS, both msieve and GGNFS will be required. While msieve does not have any library requirements, GGNFS introduces an additional library

to the system: the GNU Multiprecision Library (GMP), which has been ported to the Windows platform.

To aid with development, this project will use a version control system such as subversion or concurrent versioning system (CVS). To aid with testing, this project will rely on the MAGMA computational algebra system, which provides a number of functions useful for generating test values to factor.

IV. INPUT AND OUTPUT

CrypTool input files are arbitrary binary files to be hashed, encrypted, decrypted, signed, or verified. CrypTool output files are arbitrary binary files that are the hash, encryption, decryption, signature, or verification of the input files. Due to the nature of this extension, input and output files will likely not be required. CrypTool will be modified to support a dialog box through which users can select NFS parameters and the number to be factored. For example, Table 1 shows the parameters that will be supported. In addition to these parameters, users will be able to select what type of sieve will be used in the sieving step: classical line sieve, bucket sort line sieve, or lattice sieve.

TABLE I
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)
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 $-A < a < A$

Currently, the msieve and GGNFS implementations require their own input and output files. Both msieve and GGNFS rely primarily on their own defaults for generating polynomials. These defaults can be improved upon by introducing some of the parameters defined in Table 1. Both implementations logically break the NFS algorithm up into two steps, resulting in two sets of input and output files: polynomial selection and sieving. In the first phase of the implementation, users provide n , d , and α resulting in an output of $f(x)$ and m . This output can be saved—or replaced with different values—before running the second portion of the algorithm. During the sieving phase, both implementations produce output files containing the (a, b) relations found, allowing the sieving process to be paused and restarted—or even distributed across multiple computers. This also reduces the amount of RAM necessary to factor large numbers. The final output, the factors of n , is often provided to the user as standard output rather than an output file.

If possible, msieve and GGNFS will be modified to allow CrypTool to directly provide parameters to the NFS implementations, otherwise, CrypTool will be required to create the appropriate input files. Like with the input, if possible, these tools will be modified to provide the factorization output directly to CrypTool. The NFS extension will also provide users with detailed information about how long the NFS factoring process took.

V. FUNCTION PERFORMED BY THE PROGRAM

The CrypTool program is educational software for cryptography. The function of previous versions will not be changed. Users will still be able to select a variety of cryptographic operations and run them against data of their choosing. This project will expand CrypTool's functionality to include factoring arbitrary large numbers and provide the user with the results. Future extensions could allow users to take these results and decrypt ciphertext created using the now-cracked RSA key.

As with the two chosen implementations, the NFS extension will allow users to perform the polynomial selection step separate from the rest of the algorithm. Because polynomial selection is a non-deterministic function, this will allow users to record and re-use the results of this step before running the rest of the algorithm. Similarly, users will be able to see first hand how polynomial selection can affect the rest of the algorithm. Users will also have the ability to input polynomials generated using other tools to test their effectiveness. For the rest of the algorithm, users will be able to supply the NFS parameters described in Table 1 to monitor how the choice of parameters affects the amount of time spent running the NFS algorithm.

Ultimately, users will be allowed to specify the algorithms used for each step of the NFS algorithm in addition to specifying NFS parameters. Potentially, users will be able to store the results of each intermediate step and perform analyses of the performance of each option. Some level of this functionality will be provided at the end of this phase of the project—allowing users to select between msieve's bucket sort line sieve, GGNFS' classical line sieve, and GGNFS' lattice sieve. As mentioned in Section I, there are many more algorithms that can be added to this CrypTool extension.

VI. TESTING

The CrypTool update will be tested by generating a series of test numbers with known factors. One of the simplest mechanisms for generating these test numbers will be to use the MAGMA Computational Algebra System. MAGMA provides the *RandomPrime()* function, which accepts the size of the desired prime as a parameter. By creating test cases composed of primes of a chosen size, large (50+ digit) tests can be performed with known factors. Also, because the extension will be based on existing NFS implementations, the sieving step—when performed with the same polynomials—should produce identical relations to the original NFS implementations.

In addition to testing the functional correctness of the CrypTool extension, experiments will be performed on its performance. In particular, tests comparing the effects of different parameters and different sieves against the performance of the algorithm. By using the same value of n , $f(x)$, and m for these tests, the results will rule out any inconsistencies that may be introduced based on these probabilistic parameters.

VII. EXPERIMENTS

Tests will be performed to verify the correctness of the CrypTool update:

- Factor a 50-digit number
- Factor an 80-digit number

Tests will also be performed to analyze the performance effects of the NFS parameters:

- How does the choice of α 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?

VIII. TENTATIVE SCHEDULE

Table 2 shows the tentative schedule for this project.

TABLE II
PROJECT MILESTONES

2/23	Draft project specification
2/25	Discussion with instructor
3/1	Final project specification CrypTool version chosen CrypTool version compiled
3/8	Review of msieve code complete
3/15	Begin work on integrating msieve code with CrypTool
3/24	First progress report
3/30	Complete integrating msieve
4/6	Begin integration of GGNFS code
4/14	Second progress report
4/20	Complete integration of GGNFS code Complete experiments
4/28	Final progress report Draft presentation
5/5	Complete loose ends
5/8	Draft project report
5/12	Final presentation Final project report

IX. POSSIBLE CHANGES

The scope of the project will change depending on its progress. To ensure a deliverable by 5/12/2008, the project will be divided into two phases: msieve integration and GGNFS integration. This will ensure that at least one NFS implementation will be integrated with CrypTool, providing a testing ground for the experiments discussed in Section 7. This also provides some buffer time should integrating msieve with CrypTool prove difficult. Should this project run ahead of schedule, more experiments will be conducted or an alternate polynomial selection algorithm will be added to the extension.

X. TABLE OF CONTENTS

- 1) Introduction
- 2) Overview of NFS
- 3) Objectives
- 4) Overview of CrypTool
- 5) GGNFS and Msieve
- 6) NFS Wrapper
- 7) Experiment Results
- 8) Future
- 9) Conclusion

REFERENCES

- [1] K. Aoki, J. Franke, T. Kleinjung, A. K. Lenstra, and D. A. Osvik, "A kilobit special number field sieve factorization," in *Advances in Cryptology ASIACRYPT 2007*, ser. Lecture Notes in Computer Science, vol. 4833. Springer Berlin, Nov 2007, pp. 1–12.
- [2] K. Aoki and H. Ueda, "Sieving using bucket sort," in *Advances in Cryptology - ASIACRYPT 2004*, ser. Lecture Notes in Computer Science, vol. 3329. Springer Berlin, Nov 2004, pp. 92–102.
- [3] M. E. Briggs, "An introduction to the general number field sieve," Masters Thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, April 1998.
- [4] J. Buhler, H. Lenstra Jr., and C. Pomerance, "Factoring integers with the number field sieve," in *The development of the number field sieve*, ser. Lecture Notes in Mathematics, A. Lenstra and H. Lenstra Jr., Eds. Berlin: Springer-Verlag, 1993, vol. 1554, pp. 50–94.
- [5] S. Byrnes. (2005, May) The number field sieve. [Online]. Available: <http://www.geocities.com/sbyrnes321/math129-finalpaper.pdf>
- [6] C. Card. factor-by-gnfs. [Online]. Available: <https://sourceforge.net/projects/factor-by-gnfs/>
- [7] M. Case. (2003) A beginners guide to the general number field sieve. [Online]. Available: <http://islab.oregonstate.edu/koc/ece575/03Project/Case/paper.pdf>
- [8] S. Cavallar, B. Dodson, A. Lenstra, P. Leyland, W. Lioen, P. L. Montgomery, and B. Murphy, "Factorization of RSA-140 using the number field sieve," in *Advances in Cryptology - ASIACRYPT99*, ser. Lecture Notes in Computer Science, vol. 1716. Springer Berlin, 1999, pp. 195–207.
- [9] CrypTool. [Online]. Available: <http://www.cryptool.com/>
- [10] CrypToolLinux. [Online]. Available: <http://www.cryptoolinux.net>
- [11] N. Guo, L. T. Yang, M. Lin, and J. P. Quinn, "A parallel GNFS integrated with the block wiedemanns algorithm for integer factorization," in *2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing*. IEEE Computer Society, Sep 2006, pp. 45–50.
- [12] T. Izu, J. Kogure, and T. Shimoyama, "An FPGA implementation of the sieving step in the number field sieve method," in *Cryptographic Hardware and Embedded Systems - CHES 2007*, ser. Lecture Notes in Computer Science, vol. 4727. Springer Berlin, Aug 2007, pp. 364–377.
- [13] T. Izu, J. Kogure, and T. Shimoyama. (2007, Sep) CAIRN 3: An FPGA implementation of the sieving step with the lattice sieving. [Online]. Available: http://www.hyperelliptic.org/tanja/SHARCS/talks07/sharcs2007_cairn3_3.pdf
- [14] P. L. Jensen, "Integer factorization," Masters Thesis, University of Copenhagen, Copenhagen, Denmark, Dec 2005.
- [15] T. Kleinjung, "On polynomial selection for the general number field sieve," *Mathematics of Computation*, vol. 75, no. 256, pp. 2037–2047, Jun 2006.
- [16] C. Monico. GNU general number field sieve. [Online]. Available: <http://www.math.ttu.edu/~cmonico/software/ggnfs>
- [17] B. Murphy, "Polynomial selection for the number field sieve integer factorisation algorithm," Ph.D. dissertation, Australian National University, Jul 1999.
- [18] J. Papadopoulos. Integer factorization source code. [Online]. Available: <http://www.boo.net/~jasonp/qs.html>
- [19] P. Stevenhagen. (2004, Oct) The number field sieve. [Online]. Available: <http://websites.math.leidenuniv.nl/algebra/nfs.pdf>
- [20] L. T. Yang, L. Xu, and M. Lin, "Integer factorization by a parallel GNFS algorithm for public key cryptosystems," in *Embedded Software and Systems*, ser. Lecture Notes in Computer Science, vol. 3820. Springer Berlin, Nov 2005, pp. 683–695.