

Kevin Magee
ECE646
Project Specification
Fall 2003

Timing cryptanalysis of public key cryptosystems

1. Language, platform, and compiler used for a primary implementation. The program portability to other platforms.

Language: C++

Platform: Windows XP

Compiler: Visual C++ 6.0

Portability: Timing routines will be Windows proprietary but could be quickly rewritten for other operating systems.

2. Additional software required.

MIRACL

List of valid public/private key pairs from RSA

3. Detailed specification of the input and output of the program(s), including the exact format of input/output files.

Input:

Collection of plaintext messages and cipher text messages

Hexadecimal string representing the Private key

Output:

Hexadecimal string representing the actual private key

Timing measurements collected during samples measured in microseconds

Hexadecimal string representing the guess of the private key

Number of correct guesses

Number of incorrect guesses

Number of guesses made after an incorrect guess was made and before the error was detected and corrected.

Output can be redirected to files for storage.

4. Brief description of the function performed by the program(s), including any specific references to standards and detailed descriptions of the algorithms in the literature.

What is required for the attack is the capability to measure the transformation times from plain text to cipher text, or vice versa, from a series of known messages. The processing time for the transformations must be dependent on the private key. Additionally, the

algorithm and hardware used for the encrypting process must be well understood by the attacker.

The attack correlates the time taken for a specific iteration of the encrypting algorithm (the time dependent on one bit of the encrypting key) and the time taken to encrypt the rest of the encryption block. The attack proceeds by guessing the value of the n th bit (n starting at 0) of the encrypting key. For a correct guess the standard deviation of the encrypting times for the rest of the message block ($n > 0$) should be smaller than the standard deviation times found for an incorrect guess. Correct guesses should decrease standard deviations and incorrect guesses should increase standard deviations.

Modified encrypting program: I will begin by writing my own encrypting/decrypting public key routines. These routines will purposefully magnify the time difference required by partial operations during the encrypting/decrypting phase. Since this technique will artificially accentuate the varying processing times, it should make the timing attack easier. The length of the key will initially be small (32 bits) to further facilitate early successful attacks. I want to be able to successfully implement a timing attack in a simple environment before attacking more sophisticated routines.

“Honest” encrypting program - Next, I will reduce and eventually remove the artificial “delay()” function calls to develop an understanding of how the magnitude of the processing differences affect the timing attack. I will also extend the key length to determine how key length affects the timing attack.

Public domain encrypting program - Finally, I will use the public key encrypting key routines found in the MIRACL (since the source code is publicly available and can therefore be studied) to determine if I can successfully attack an optimized implementation of public-key encryption with an industrial strength key length. If time permits I will also attempt to attack the RSA signature generation process.

To perform the attack a series of plain text/public key and cipher text/public key pairs will be offered to the program. The program will record the timing measurements of the processing times needed for encrypting/decrypting the messages. The program will then perform a statistical analysis of the measurements in an attempt to discover the private key.

5. Procedures for testing the functionality and performance of the program(s). The source of test vectors.

Measure the correctness of the cryptanalysis

Measure the effectiveness based upon length of key (How accurate is the attack given longer keys?)

Measure the effectiveness based on difference in delta processing times (see section 4 above).

Source of test vectors will be self-generated for the early part of the project (see “Modified” and “Honest” encrypting programs above). For the final part of the project RSA developed key pairs will be used.

6. Plan of experiments to be performed using the program(s).

There will be three experimental phases:

- 1) experiments with the “Modified” encrypting program,
- 2) experiments with the “Honest” encrypting program,
- 3) experiments with the “Public Domain” encrypting program.

The outlines of these three programs are described in Section 4. The types of measurements to be collected are described in Section 5.

7. Time schedule, including intermediate goals to be achieved by the dates of progress reports.

Progress Report 1)

Survey available literature

Develop timing measurement routines

Develop code to generate cipher text/clear text and private key

Develop “modified encrypting program” described in section 4 above

Progress Report 2)

Perform attacks against the “modified encrypting program”

Develop the “honest encrypting program” (section 4 above) and collect data on how varying key lengths and smaller processing deltas affect the timing attack

Progress Report 3)

Attack the MIRACL public key encryption routine with full recommended key lengths for public key encryption.

8. A list of possible areas, where the specification can change depending on the progress of the project.

An earlier attempt at this project (Chris Davis, Fall 2001) developed interesting results but was never able to “crack” an encrypted public-key message. I intend to develop this project so that the early phases of the project will produce successful attacks. This is why I will write my own encryption routines that will allow me to adjust processing variances and key lengths. Given the difficulty of attacking commercial public-key implementations, I will learn more and will be able to offer a more interesting presentation to the class if I can produce some level of success for a timing attack before attacking commercial implementations.

9. List of literature.

- 1) Paul C. Kocher, "Cryptanalysis of Diffie-Hellman, RSA, DSS, and Other Systems Using Timing Attacks," extended abstract, December 1995.
- 2) J. Markoff, "Secure digital transactions just got a little less secure," New York Times, December 11, 1995.
- 3) B. Kaliski, "Timing Attacks on Cryptosystems," RSA Laboratories' Bulletin, no 2, January 23, 1996.
- 4) Paul C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," Proc. CRYPTO'96, pp. 104-113.
- 5) RSAREF library, available via ftp from ftp://ftp.rsa.com.
Handbook of Applied Cryptography , Menezes, Oorschot, Vanstone
- 6) Cryptography and Network Security, Stallings
- 7) "A Practical Implementation of the Timing Attack", Dehm, Koeune, Leroux, Mestre, Quisquater, Willems. UCL Crypto Group
- 8) "A Timing Attack Against Rijndael", Koeune, Quisquater. UCL Crypto Group
- 9) "Fast Modular Reduction with Precomputation", Chae Loon Him, Hyo Sun Hwang
- 10) GSL Reference Manual, version 1.0. 1 November 2001
"Careful Design and Integration of Cryptographic Primitives", Koeune, UCL Crypto Group
- 11) "Timing Attack: What Can be Achieved by a Powerful Adversary?", Hachez, Koeune