Timing Cryptanalysis of Public-Key Cryptosystems

Chris Davis
ECE646 – Fall 2001

The Gist

• Timing characteristics of a cipher implementation can reveal aspects of the private (or secret) key.
• Applicable to asymmetric and symmetric ciphers.
Background

- Timing cryptanalysis came from new direction.
- Most types of cryptanalysis were based upon weaknesses of the mathematical model describing the cryptosystem.
- Efficiency is high-priority for vendors and implementers.
Prerequisites

- 4 requirements
  - Ability to measure total transformation times with desired key
  - Ability to capture plaintext or ciphertext messages used as input into the cipher
  - General design knowledge of the implementation
  - Ability to measure timing characteristics of a single iteration.

Example Scenario
Attack Description

- Kocher’s Method
  - Capture messages and measure transformation times
  - Reconstruct key through simulated trial and error
  - Detect errors through statistical analysis

- Kocher’s Method (cont.)
  - Key Reconstruction
    - Make guess for first bit of key. Assume the guess is correct.
    - Make guess for second bit of key. Measure time required for each transformation and compute variance.
    - Compare computed variance with expected variance
      - Correct values should cause variance to decrease
      - Incorrect values should cause the variance to increase.
Kocher’s Method

\[ T(M_1) - T_{b0=0}(M_1) \quad T(M_2) - T_{b0=0}(M_2) \quad T(M_3) - T_{b0=0}(M_3) \]

\[ T_{b1=1}(M_1) \quad T_{b1=1}(M_2) \quad T_{b1=1}(M_3) \]

Kocher’s Method

\[ T(M_1) - T_{b1=0}(M_1) \quad T(M_2) - T_{b1=0}(M_2) \quad T(M_3) - T_{b1=0}(M_3) \]

\[ T_{b1=1}(M_1) \quad T_{b1=1}(M_2) \quad T_{b1=1}(M_3) \]
UCL Crypto Group Method

• Chosen-plaintext (or ciphertext) attack
• Depends on the ability to construct two sets of messages that evince different timing characteristics in the implementation
• Has the ability of attacking multiple points within an implementation

Implementation

• Attacked Diffie-Hellman key exchange as implemented in RSAREF
• DH key exchange relies on left-to-right modular exponentiation
• RSAREF precomputes $x \mod N$, $x^2 \mod N$, and $x^3 \mod N$
• RSAREF also skips leading zero-bits => the first guess is assumed to be a one-bit.
Implementation

- Generated 65,000 “messages” for the DH key exchange based upon the same set of parameters.
- Since RSAREF’s modular exponentiation algorithm processes two bits per loop iteration, recovering the full key through timing cryptanalysis was not possible.
  - Average time of computation for 01, 10, and 11 bit-pairs was approximately equal

Implementation

- Any 00 bit-pair correct
- Any bit-pair containing a 1 is ambiguous and can be either 01, 10, or 11
- Was able to correctly identify 12-bits of the private value as 00
- Successfully reduced the keyspace for a 48-bit private value to $3^{18}$
Real-world Vulnerabilities

• Oakley key-exchange algorithm
  – Used in IKE (part of IPSec)
  – First and second Oakley groups use the Diffie-Hellman key exchange algorithm to determine session keys
  – A large collection of roaming users connection via IPSec could provide enough information for timing cryptanalysis

Countermeasures

• Masking (Kocher)
  – Select random X

\[
((M \times X)^d \mod N) \times ((X^{-1})^d \mod N) \mod N = (M^d \mod N) \times ((X \times X^{-1})^d \mod N) \mod N = M^d \mod N
\]
Countermeasures

• Blinding (Rivest)
  – Makes use of RSA characteristics

\[
(M \cdot (X^{-1})^e)^d \mod N \times [X \mod N] =
\]
\[
[M^d \mod N \times (X^{-1})^{ed} \mod N \times [X \mod N] =
\]
\[
M^d \mod N
\]