Public-Key Cryptanalysis (II): Square-Root Attacks

Phong Nguyễn
http://www.di.ens.fr/~pnguyen

INRIA and École normale supérieure, Paris, France

MPRI, 2010

Lessons for Encryption

- Encryption must be made probabilistic.
- But it must be done carefully.
- Defining security for encryption is tricky: it took more or less twenty years to find the right notion! We'll come back to it.

Lessons for Signature

- Messages must be preprocessed before being signed, to avoid trivial existential forgeries.
- But even with preprocessing, forgeries may be easier than the general problem.
- This highlights the importance of "provable security".
- Defining security for signature is much easier than for encryption.
- "Provably secure" deterministic signatures are possible, while "provably secure" deterministic encryption was not! One could argue that deterministic signatures are even preferable to probabilistic signatures.

Philosophy of Square Root Attacks

- There are many examples of brute force search attacks.
- These have exponential complexity but require little memory.
- In practice it is often the case that problems can be split up in a manner which allows a time/memory tradeoff.
- Hence, one can reduce the running time by increasing the memory requirement.
- Algorithms of this type are often called time/memory tradeoff or birthday attacks.
  They are also often called square-root attacks. The key idea is to split the secret in two equal parts. Let's see a few examples.

A Square-Root Attack on the Discrete Logarithm

- Let (N, e) be an RSA key.
- Suppose the RSA private exponent d satisfies $1 < d < B$.
- Choose a random $1 < m < N$ and compute $c = m^e \pmod{N}$.
- Then $m \equiv c^d \pmod{N}$.
- In other words, finding d may be viewed as a discrete logarithm problem.
- We describe the baby-step-giant-step algorithm due to Dan Shanks.

Baby-step-giant-step algorithm

- Define $M = \lfloor \sqrt{B} \rfloor$.
- Then d can be written as $d = d_1 + M d_2$ where $0 \le d_1 < M$ and $0 \le d_2 \le M + 1$. Hence, $m \equiv c^d \pmod{N}$ is rewritten as $m / c^{M d_2} \equiv c^{d_1} \pmod{N}$. We are now looking for collisions!
- For $i = 0, 1, \ldots, M - 1$ compute the baby steps $c^i \pmod{N}$.
- These values (together with the corresponding values of i) must be stored in a structure such as a binary tree or hash table.

Baby-step-giant-step algorithm

- Compute $C = c^M \pmod{N}$.
- For $j = 0, 1, \ldots$ compute the giant steps $m/C^j \pmod{N}$.
- For each value, check to see if it appears in the tree/table of baby steps.
  This is easy to do when the baby steps are stored in a binary tree or hash table.
- Once a match is found we have $c^i \equiv m/c^{Mj} \pmod{N}$ and so $d = i + Mj$.

Baby-step-giant-step algorithm

- Clearly the baby-step-giant-step algorithm is guaranteed to terminate with the correct answer if $1 \leq d < B$.
- The time and space complexity are both $\tilde{O}(\sqrt{B})$.
- Exercise: Show that if the available memory is only enough to store $M < \sqrt{B}$ integers modulo N then one can obtain an algorithm with time complexity $\tilde{O}(B/M)$.
- There is a completely different (and much more efficient) way to find the RSA private exponent d if it is small. This is the Wiener attack and it will be presented as a lattice attack.

A Square-Root Attack on the Message

- We started with showing brute force search attacks on RSA over either the message m or the private key d.
- We have improved the latter attack using the time/memory tradeoff.
- The key idea was an additive splitting of d, essentially into low-order and high-order halves.
- A time/memory tradeoff for the former case was proposed by Boneh, Joux and Nguyen [AC 2000].
- The idea in this case is to use a multiplicative splitting of the problem via $m = m_1 m_2$.

Boneh-Joux-N attack

- Suppose we know that $1 \leq m < B$ and we have

$$c \equiv m^e \pmod{N}.$$

- It might happen that m can be split as $m = m_1 m_2$ where $m_1, m_2 \approx \sqrt{m} \approx \sqrt{B}$.
- Splitting probabilities are listed in the paper of Boneh, Joux and N.
- For example, if $1 \leq m < 2^{64}$ then m can be split as a product $m_1 m_2$ where $1 \leq m_i < 2^{32}$ with probability 0.18.
- Extending to $1 \leq m_i < 2^{33}$ gives probability 0.29, and to 2^{34} gives probability 0.35.

Boneh-Joux-Nguyen attack

- Compute all the values ${m_1}^e \pmod{N}$ where $1 \leq m_1 \leq A\sqrt{B}$ (for some constant A).
- These values (together with the corresponding m_1) should be stored in a structure which is easily searched.
- For $1 \leq m_2 \leq A'\sqrt{B}$ (for some constant A') compute $c/{m_2}^e \pmod{N}$ and, for each value, see if this number appears in the earlier structure.
- If a match is found then we have $c/{m_2}^e \equiv {m_1}^e \pmod{N}$ in which case $c \equiv (m_1 m_2)^e \pmod{N}$ and so $m = m_1 m_2$.

Boneh-Joux-Nguyen attack

- The time complexity of this attack is $\tilde{O}(\sqrt{B})$.
- The storage requirement is also $\tilde{O}(\sqrt{B})$.
- Unlike the baby-step-giant-step method, this approach does not succeed for all inputs.

Low memory versions

- It is a surprising fact that many problems which can be solved by square-root methods actually can be solved by randomised algorithms of similar time complexity but with constant space requirements.
- For example, the Pollard ρ and λ methods solve the discrete logarithm problem with time complexity close to that of the baby-step-giant-step method, but with very small space requirements.
- Open problem: Give a low memory version of the BJN attack.

Low-Hamming Weight

- To speed up RSA it is tempting to choose d of a specific form.
- One way is to choose the private exponent to have low Hamming weight. This is especially true for El Gamal.
- The key idea (due to Coppersmith) is that if d is a random n-bit integer with low Hamming weight w then d can usually be split as $d = d_1 + 2^{\lfloor n/2 \rfloor} d_2$ where the d_i are $n/2$-bit integers with Hamming weight roughly $w/2$.
- Hence we can expect time/memory tradeoff attacks with complexity roughly

$$\binom{n/2}{w/2} \approx \sqrt{\binom{n}{w}}.$$

- See Stinson's paper for details.

The small CRT private exponent

- To speed up RSA, one could alternatively select d small.
- Due to the Wiener attack and the extension due to Boneh and Durfee one must take $d > N^{0.292}$.
- A better way to speed up RSA is to choose $N = pq$ and e so that the integers d_p and d_q satisfying

$$ed_p \equiv 1 \pmod{p-1} \quad \text{and} \quad ed_q \equiv 1 \pmod{q-1}$$

  are small.
- Exercise: Design an RSA key generation algorithm which produces keys of this form.
- It seems that the Wiener attack and its generalisations using lattices can no longer be applied in this case.
- Exercice (hard): Find a square-root attack.

Conclusions

- Any restriction of parameters to small ranges or to having special properties is a potential vulnerability.
- When attacking a cryptosystem, first seek a brute-force attack.
- Then try to refine this using the time/memory tradeoff by splitting the problem into parts.
- If this is successful then try to find a low memory version.

Selected references

- D. Boneh, A. Joux and P. Q. Nguyen, ‘Why textbook El Gamal and RSA encryption are insecure’, ASIACRYPT 2000, 30–43.
- A. J. Menezes, P. van Oorschot and S. Vanstone, ‘The handbook of applied cryptography’, CRC press (1997).
- D. Shanks, ‘Class number, a theory of factorisation and genera’, Proc. 1969 Number Theory Institute, AMS (1971) 415–440.
- N. Smart, ‘Cryptography: An introduction’, McGraw-Hill (2003).
- D. Stinson, ‘Some baby-step-giant-step algorithms for the low Hamming weight discrete logarithm problem’, *Math. Comp.* **71**, No. 237 (2001) 379–391.