Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 6 (1): 76-79, 2006
ISSN 1812-5654

Improvements to Shank's Baby-step Giant-step Algorithm

Kaqish Malek
Department of Computer Information Systems, Al-Ahliyya Amman University,
Al Salt Road, P.O.Box:183, Amman 19328, Jordan

Abstract: The security of many cryptographic cryptosystems (e.g., RSA, ElGamal, Diffie-Hellman, Elliptic Curves Cryptosystems, etc.) depends on the apparent intractability of solving some number of theoretic problems e.g., The Discrete Logarithm Problem, Factorization Problem, Diffie-Hellman Problem, Quadratic Residuosity Problem, etc.[1]. These problems are generally regarded as being difficult if the associated parameters are carefully chosen. The Discrete Logarithm Problem can be defined as followed: If we assume Z_p (denotes the set of integers {0, 1, 2, …., p - 1}, where addition and multiplication are performed modulo p) is a finite cyclic group of order p, where α a generator of Z_p and $y \epsilon Z_p$, Then the Discrete Logarithm of y to the base α, denoted $\log_\alpha y$, is the unique integer x, $0 \leq x \leq p-1$, such that $y = \alpha^x$. Many algorithms have been introduced for solving such problems, this study described an improvement to Shank's baby-step giant-step algorithm for computing the discrete logarithm x of an element y (where, $y=\alpha^x \bmod p$, $y \epsilon Z_p$, α generator of Z_p). My improvements enable the computation of the discrete logarithm x faster than original Shank's algorithm this is done by using practically suitable choice of parameters, it also allows up to 50% less memory usage, it can be applied for parallel calculation of the discrete logarithm, it can be implemented on finite fields $F_{2^f}^*$, $F_{p^f}^*$ and other abelian groups e.g. elliptic curves, this attack is much more significant on elliptic curves groups, where the group size is much more smaller compared to finite fields groups and there is no known sub-exponential algorithm for computing discrete logarithms on elliptic curves unlike discrete logarithms in a finite field.

Key words: Cryptanalysis, Shank's algorithm, cryptosystems, mersenne list, baby-step giant-step, discrete logarithm problem

INTRODUCTION

The dramatically growing number of the internet users, services and the demand for connectivity to the Internet resources has forced standards organizations (e.g., IAB, IETF, IETG, etc.)[1-4] to specify a huge set of security protocols, algorithms and applications that provides security services (confidentiality, authentication, integrity, etc.) which meets the need for secure communication.

There is no single mechanism that will provide all the security services just listed before or perform all the function, however, we can note at this point that here is one particular element that underlies most of the security mechanisms in use: cryptographic techniques or generally cryptography (science of data encryption and decryption).

Cryptography has a old and fascinating history. Kahn´s book The Codebreaker[5] describes a 4000 years historical overview from its limited use by the Egyptians until the twentieth century where it played a central role in war telecommunication. Today Cryptography enables you to securely store sensitive information or transmit it across insecure networks so that it cannot be read by anyone except the intended recipient. By using a powerful tool such as encryption we gain privacy, authenticity, integrity and limited access to data.

Cryptographic systems can be divided in conventional cryptography systems and public key cryptography systems. In conventional cryptography, also called secret-key or symmetric-key encryption, one key is used both for encryption and decryption. The development of computers and communications systems in the 1960s brought with many such cryptographic systems like Data Encryption Standard (DES), Triple Data Encryption Standard (3DES), Advanced Encryption Standard (AES), The International Data Encryption Algorithm (IDEA), Blowfish, RC5, CAST, etc.

Diffie and Hellman[6] published new directions in cryptography. This paper introduced the revolutionary concept of public-key cryptography and solved many weaknesses and problems in private key cryptography e.g., key distribution and key management problem, user authentication, etc. Upon this, many public key cryptographic systems (e.g., RSA, ElGamal[7], Diffie-Hellman, Elliptic Curves (EC), Digital Signature Algorithm (DSA), etc.) were developed and the security of such cryptosystems are based on apparently difficult

mathematical number theory problems like the discrete logarithm problem over finite fields, the discrete logarithm problem over elliptic curves, integer factorization problem, Diffie Hellman problem, Quadratic residuosity problem, etc.

PROBLEM DEFINITION

If p is a prime number, then Z_p denotes the set of integers {0, 1, 2, …., p-1}, where addition and multiplication are performed modulo p. It is well-known that there exists a non-zero element $\alpha \epsilon Z_p$ such that each non-zero element in Z_p can be written as a power of α; such an element α is called a generator of Z_p. A group is called cyclic if such an element α exists.

A Field is a nonempty set F of elements with two operations "+" (called addition) and "·" (called multiplication) satisfying the following axioms: for all a, b, $c \in F$,

- F is closed under + and i.e., a + b and a·b are in F;
- Commutative laws: a + b = b + a, a·b = b·a;
- Associative laws: (a + b) + c = a + (b + c), a· (b·c) = (a·b)·c;
- Distributive law: a·(b + c) = a·b + a·c.

Furthermore, two distinct identity elements 0 and 1 (called the additive and multiplicative identities, respectively) must exist in F satisfying

- a + 0 = a for all $a \epsilon F$;
- a·1 = a and a·0 = 0 for all $a \epsilon F$;
- For any a in F, there exists an additive inverse element (a) in F such that a + (a) = 0;
- For any $a \neq 0$ in F, there exists a multiplicative inverse element a^{-1} in F such that $a \cdot a^{-1} = 1$.

Finite field of prime order or prime power q is commonly denoted F_q or GF(q) (for Galois field) and because Z_m is a field if and only if m is a prime, we denote the field Z_m by F_m. This is called a prime field.

Now the Discrete Logarithm Problem can be defined as followed: If we assume Z_p is a finite cyclic group of order p, where, α a generator of Z_p and $y \epsilon Z_p$. Then the discrete logarithm of y to the base α, denoted $\log_\alpha y$, is the unique integer x, $0 \leq x \leq p-1$, such that $y = \alpha^x$[8].

The Discrete Logarithm Problem (DLP) in Z_p has been the object of much study. The problem is generally regarded as being difficult if field parameters are carefully chosen. In particular, there is no known polynomial-time algorithm for the Discrete Logarithm problem. There are two types of algorithms for solving the discrete logarithm problem. Special-purpose algorithms[9] attempt to exploit special features of the prime p. In contrast, the running times of general-purpose algorithms (e.g., exhaustive search, Shank's Baby step Giant step, Pohlig Hellman[10], Pollard's rho algorithm[11], Index Calculus method, Number field Sieve etc.) depend only on the size of p. The fastest general-purpose algorithms known for solving the discrete logarithm problem are based on a method called the index-calculus and the best current algorithm known for the DLP is the number field sieve[12]. To thwart these attacks, p should have at least 150 digits and p-1 should have at least one large prime factor. The utility of the discrete logarithm problem in a cryptographic setting is that finding discrete logs is (probably) difficult, but the inverse operation of exponentiation can be computed efficiently (e.g., the square-and-multiply method), this property is known in number theory as trapdoor or one-way function.

In this paper I describe improvements to Shank's baby-step giant-step algorithm[13] for computing the discrete logarithm x of an element y (where, $y = \alpha^x \bmod p$, $y \epsilon Z_p$, α generator of and p is prime).

SHANK'S BABY-STEP GIANT-STEP ALGORITHM

The Baby-Step Giant-Step Algorithm for finding the discrete logarithm *x* in finite fields Z_p (where, α generator of a cyclic group Z_p of order p, an element, $\alpha^x = \beta$, $\beta \epsilon Z_p$) can be described as followed:

- Set m _$\leftarrow |\sqrt{p}|$
- Construct a table with entries (j, α^j) for $0 \leq j < m$. Sort this table by second component.
- Compute α^{-m} and set $\gamma \leftarrow \beta$
- For i from 0 to m-1 do the following:
  - i Check if γ is the second component of some entry in the table
  - ii If $\gamma = \alpha^j$ then return (x = im + j).
  - ii Set $\gamma \leftarrow \gamma * \alpha^{-m}$

This algorithm require storage for $O(\sqrt{p})$ group elements, If we assume $p = 10^{300} \approx 2^{1000}$, follows that $m = 10^{150}$.This means with 10^{150} pair of elements. If we assume that each entry (j, α^j) cost 250 Bytes, we would need $2{,}5*10^{143}$ Gigabytes memory. A comparison: The universe contains 10^{77} Atoms.

The table takes $O(\sqrt{p})$ multiplications to construct and $O(\sqrt{p}\ 1\ g(p))$ comparison to sort. After having constructed the table, step 4 takes $O(\sqrt{p})$ multiplication and $O(\sqrt{p})$ table look-ups.

Thus the algorithm can be implemented to run in $O(\sqrt{p})$ time.

IMPROVEMENTS TO SHANK'S ALGORITHM

List length : m (list length) can be assigned other values than Shank's step 1 propose. m can be assigned the maximum number of elements that can fits in user memory.

We set:

$$a = \left\lfloor \frac{p}{m} \right\rfloor$$

Then

$$p = am + s$$

Where, $0 < s < m$
After (a + 1) iterations we get:

$$(a + 1)m = am + m > am + s = p$$

This means that more than p elements of F^*_p will be tested, but:

$$am < am + s = p$$

The maximum number of iteration is

$$a + 1 = \left\lfloor \frac{p}{m} \right\rfloor + 1.$$

Thus, the required number of iteration for algorithm determination is contrary proportional to the length of the table

THE SEARCHING ALGORITHM

The Shank's algorithm F^*_p makes at each for-iteration $\log_2 (m+1)$ checks with a given element γ thus it is important that the table length equals a Mersenne-number $M_n = 2^n - 1$

A Mersenne List L_n (n>1) of length 2^n-1 with sorted elements have the following properties:

- Each Mersenne list has a middle element. His index is at 2^{n-1}.
- All element right (left) from the middle element build also Mersenne list L_{n-1}
- All 2^{n-1} Elements right (left) from the middle element are bigger (smaller) than the middle element

The maximum number of searching step in a Mersenne List L_n requires maximum n steps.

TABLE STRUCTURE

Instead of storing two entries in proposed Shank's table (step 2), I suggest using one dimension array, this will allow saving 50% of memory usage and enable us working with longer table.

MODIFIED SHANK'S ALGORITHM: The Modified Shank's Algorithm for finding the discrete logarithm *x* in finite fields Z_p (where,α generator of a cyclic group Z_p of order p, an element β, $\alpha^x = \beta$, $\beta \epsilon Z_p$) can be described as followed:

Step 1: Choose m as described in 5.1

Step 2: Construct a one dimension array A1 with entry

$(\overline{p} * \alpha^j + j)$ for $0 \le j < m, \overline{p} < p, \overline{p}$ is prime.

A1:	$\overline{p}\alpha^{j_0} + j_0$	$\overline{p}\alpha^{j_1} + j_1$	...	$\overline{p}\alpha^{j_{m-2}} + j_{m-2}$	$\overline{p}\alpha^{j_{m-1}} + j_{m-1}$

Step 3: Sort this table by α^j.

Step 4: Compute α^{-m} and set $\gamma \leftarrow \beta$

Step 5: For i from 0 to m-1 do the following:

5.1 Calculate α^{j_x}, j_x out of A1
5.2 Check if γ is equal α^{j_x} in the array A1 (using binary search)
5.3 If $\gamma = \alpha^{j_x}$ then return $(x = im + j_x)$.
5.4 Set $\gamma \leftarrow \gamma * \alpha^{-m}$

THE FIELDS WITH THE CHARACTERISTIC 2

Fields of characteristic 2 are often used because of many suitable properties. First they enable easily implementation in hardware. AND, OR, XOR, operation on {1, 0} alphabet can be done fast, Addition and Subtraction are identical, Quadratic operation can be done very fast and they also enable efficient usage of memory bits.

Let p be a prime number and let $f \epsilon Z+$. The field (unique up to isomorphism) with p^f elements is called the Galois field of order $q = p^f$, denoted by GF (p^f).

If $q = 2^f$ is the order of a field F_q, then all fields elements are polynomials of the form:

$$a_0 + a_1\alpha^1 + ... + a_{f-1}\alpha^{f-1}$$

Where, each and $a_i \epsilon \{0,1\}$ a generator of F_q. These polynomials can then be stored on computer in the form:

$$a_0, a_1, ..., a_{f-1}$$

Each a_i require only one bit, thus for storing of one F_q field element, f bits will be required and for sorting these element we could consider their binary form.

THE FIELDS OF ORDER $q = p^f$ (where, p>2, f >1)

If $q = p^f$ is the order of a field F_q, then all elements are polynomials of the form:

$$a_0 + a_1\alpha^1 + ... + a_{f-1}\alpha^{f-1}$$

Where, each $a_i \in \{0,1,...,p-1\}$ and α a generator of F_q. These polynomials can then be stored on computer in the form:

$$a_0, a_1, ..., a_{f-1}$$

Where, each a_i would require $[\log_2 p]$ bits, thus for storing one F_q field element $f^*[\log_2 p]$ bits are required. And for sorting field's element we could consider their binary form.

CONCLUSIONS

The security of public-key cryptographic systems is often based upon a single computationally hard problem (e.g. the discrete logarithm problem) however; they will no longer be secure if the corresponding hard problem is solved in the future.

Recently Quantum computers are often discussed as one of the most long-term threads to discrete logarithm cryptographic systems. Such machines could compute the discrete logarithm problem in polynomial time[14].

This technology is still at the beginning and thus it is expected that many years are needed to build such machine.

Second thread comes from Special purpose hardware devices for solving the discrete logarithm problem. It is unlikely that this approach will have impact, due to Pomerance experiment[15] of using internet computing power of some idle systems to solve factorization problems.

Other threads concern general purpose algorithms. The best known algorithm for solving the discrete logarithm problem for any finite field groups is the index calculus method, unfortunately it can not be transformed for elliptic curves groups, the best known algorithm for any other finite abelian group (such as elliptic curves) one can use the Pollard techniques[16], Shanks Baby step Giant step algorithm[17].

This study discussed new improvements that makes Shank's algorithm practically (using appropriate table length) and faster (using Mersenne list) with up to 50% less memory usage, it can also be implemented on finite fields F_{2^f}, F_{p^f} and other abelian groups e.g. elliptic curves, this attack is much more significant on elliptic curves groups, where the group size is much more smaller compared to finite fields groups, but the calculation of the discrete logarithm x in F^*_p where, p have alt least 150 digits and p-1 have at least one large prime factor, will stay hard.

Many years of study did not show new algorithms for solving the discrete logarithm problem in general, but maybe in future new mathematic insights will allow the definition of new attacks that can generally solve the Discrete Logarithm Problem in reasonable time and for any given parameter.

REFERENCES

1. The Internet Architecture Board, http://www.iab.org/
2. IEEE Computer Society, http://www.computer.org/
3. The Internet Engineering Task Force, http://www.ietf.org/
4. The International Telecommunication Union, http://www.itu.int/home/index.html
5. RFC 2631: DIFFIE-HELLMAN Key Agreement Method
6. Diffie, W. and M. Hellman, 1976. New directions in cryptography. IEEE Transactions on Information Theory, 22: 644-654.
7. ElGamal, T., 1985. A public-key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inform. Theory, 31:469-472.
8. Menezes, A., P. van Oorscot and S. Vanstone, 1999. Handbook of Applied Cryptography. CRC Press, ISBN: 8493-8523-7.
9. Denny, T. and D. Weber, 1998. The Solution of McCurley's Discrete Log Challenge. Advances in Cryptology-CRYPTO 98, Lecture Notes in Computer Science, colume 1462, Springer-Verlag, pp: 458-471.
10. Pohlig, S. and M. Hellman, 1978. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. IEEE Trans. Inform. Theory, 24: 106-110.
11. Pollard, J., 1978. Monte Carlo methods for index computation mod p. Mathematics of Computation, 32: 918-924.
12. Gordon, D., 1993. Discrete logarithms in GF(p) using the number field sieve. SIAM Discrete Mathematics, 6: 124-138.
13. Knuth, D., 1973. Sorting and Searching: The Art of Computer Programming, vol. 3, Addison-Wesley.
14. Shor, P.W., 1997. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput., 26: 1484-1509. http://www.research.att.com/shor.
15. Pomerance, C., J.W. Smith and R. Tuler, 1988. A pipeline architecture for factoring large integers with the quadratic sieve algorithm, SIAM J. Comput., 17: 387-403.