CS548 Advanced Information Security

An Improved Algorithm for Computing Logarithms over GF(p) and Its Cryptographic Significance Function

Stephen C. Pohlig and Margin E. Hellman

IEEE Transactions on Information Theory, 1978

2010. 03. 09.

Kanghoon Lee, AIPR Lab., KAIST

KAIST

Contents

- Introduction
- Use in Cryptography
- Mathematical Backgrounds
- Algorithm for $p = 2^n+1$
- Algorithm for Arbitrary Primes
- Discussion

What's the problem ?

Pair of Inverse Functions

$$y \equiv \alpha^x \quad (\mathrm{mod}\ p)$$

$$x \equiv \log_\alpha y \quad \text{over} \quad GF(p)$$

easy

difficult

where p is prime, α is a fixed primitive element of $GF(p)$

① $y \equiv \alpha^x\ (\mathrm{mod}\ p)$: $O(\log_2 p)$ time complexity (ex. $\alpha^{18} = (((\alpha^2)^2)^2)^2\alpha^2$)

② $x \equiv \log_\alpha y$ over $GF(p)$: Previously , $O(\sqrt{p})$ time & space complexity

One-way Function : Original problem – easy
Inverse problem – difficult

Really ?

KAIST

p-1 Must Have Large Prime Factor !

$$x \equiv log_{\alpha} y \quad over \quad GF(p)$$

- ✓ Can we solve this problem faster than $O(\sqrt{p})$ time ??

- ✓ Over GF(p), when **<u>p-1 has only small prime factors</u>**,

  the logarithm problem can be solved $O(log^2 p)$

- ✓ To make one-way function,
  <u>p-1 must have at least one large prime factor</u>

KAIST

Use in Cryptography

- For plain-text M, key K, cipher-text C with the restrictions

  $1 \le M, C \le p-1, \quad 1 \le K \le p-2, \quad GCD(K, p-1) = 1,$

  $$\boldsymbol{C \equiv M^K \ (mod\ \ p)}$$

- For deciphering operation,

  $$\boldsymbol{M \equiv C^D \ (mod\ \ p)}, \quad \text{where} \quad D \equiv K^{-1} \ (\bmod p-1)$$

  (*D* is uniquely determined because *GCD(K, p-1) = 1*)

- Finding the key K is equivalent to computing

  $$\boldsymbol{K \equiv log_M C} \quad \text{over } \boldsymbol{GF(p)}$$

Background – Finite Field (Algebra)

✓ ***GF(p)*** : Galois Field (a.k.a. Finite Field)

➔ A field that contains only finitely many elements

✓ Computations over *GF(p)*

ex. When *p=5* (i.e. *GF(5)*),

$$3 + 4 \equiv 2 \pmod{5}, \quad 3 - 4 \equiv 4 \pmod{5}$$
$$3^2 = 9 \equiv 4 \pmod{5}, \quad \log_3 4 \equiv 2 \pmod{5}$$

✓ **Primitive Element** : A generator of the multiplicative group of the field

ex. $3^1 \equiv 3, \quad 3^2 \equiv 4, \quad 3^3 \equiv 2, \quad 3^4 \equiv 1 \pmod{5}$

So, 3 is a primitive element of *GF(5)*

Background - Number Theory (1)

- **Euler's φ - function** (a.k.a. Euler's totient function)

$$\varphi(p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}) = (p_1 - 1) p_1^{e_1} (p_2 - 1) p_2^{e_2} \cdots (p_k - 1) p_k^{e_k} \;, \quad \text{where } p_i\text{'s are prime}$$

$$= p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k} \prod_{p_i} \left(1 - \frac{1}{p_i}\right)$$

- **The fraction ρ**

$$\rho = \frac{\varphi(p-1)}{p-1} = \prod_{p_i | (p-1)} \left(1 - \frac{1}{p_i}\right) \qquad \left(\forall p < 1.6 \times 10^{103} \Rightarrow p > 0.1\right)$$

- For primes of the form $p = 2p' + 1$, with p' prime, $\rho = \frac{1}{2}\left(1 - \frac{1}{p'}\right) \approx \frac{1}{2}$

Background - Number Theory (2)

- **Fermat's Little Theorem**

  $$z^{p-1} \equiv 1 \pmod{p}\ ,\quad 1 \le z \le p-1$$

- From the theorem

  $$z^{x} \equiv z^{x \,(\mathrm{mod}\, p-1)} \pmod{p}$$

- **Chinese Remainder Theorem**

  Suppose that $n_1, n_2, \ldots, n_k$ are positive integers which are pairwise coprime.

  For any integers $a_1, a_2, \ldots, a_k$, there exist an integer x (mod $n_1 n_2 \ldots n_k$) satisfying

  $$x \equiv a_1 \pmod{n_1}\ ,\quad x \equiv a_2 \pmod{n_2}\ , \cdots,\ x \equiv a_k \pmod{n_k}$$

- c.f. **Euler's Theorem**

  For any positive integer n, z ($GCD(n,z)=1$)

  $$z^{\varphi(n)} \equiv 1 \pmod{n}$$

An Algorithm for p = 2^n+1 (1)

- Given $\alpha,\ p,\ y$, (α is a primitive element of *GF(p)*)

  Must find *x* such that $y \equiv \alpha^x \pmod{p}$

- Let $x = \sum_{i=0}^{n-1} b_i 2^i$,

- Then, b_0 is determined by

$$y^{(p-1)/2} \pmod{p} = \begin{cases} +1\,, & \text{if } b_0 = 0 \\ -1\,, & \text{if } b_0 = 1 \end{cases}$$

($\because$) Since α is primitive, $\alpha^{(p-1)/2} \equiv -1 \pmod{p}$

Therefore, $y^{(p-1)/2} = (\alpha^x)^{(p-1)/2} \equiv (\alpha^{(p-1)/2})^x \pmod{p}$

An Algorithm for p = 2^n+1 (2)

- Now, b_1 is determined by letting

  $$z \equiv y\alpha^{-b_0} \equiv \alpha^{x_1} \pmod{p}, \quad \text{where } x_1 = \sum_{i=1}^{n-1} b_i 2^i$$

- Then,

  $$z^{(p-1)/4} \pmod{p} \equiv (\alpha^{x_1})^{(p-1)/4} \equiv (\alpha^{(p-1)/2})^{x_1/2} \equiv (-1)^{x_1/2}$$

  $$\equiv \begin{cases} +1, & b_1 = 0 \\ -1, & b_1 = 1 \end{cases}$$

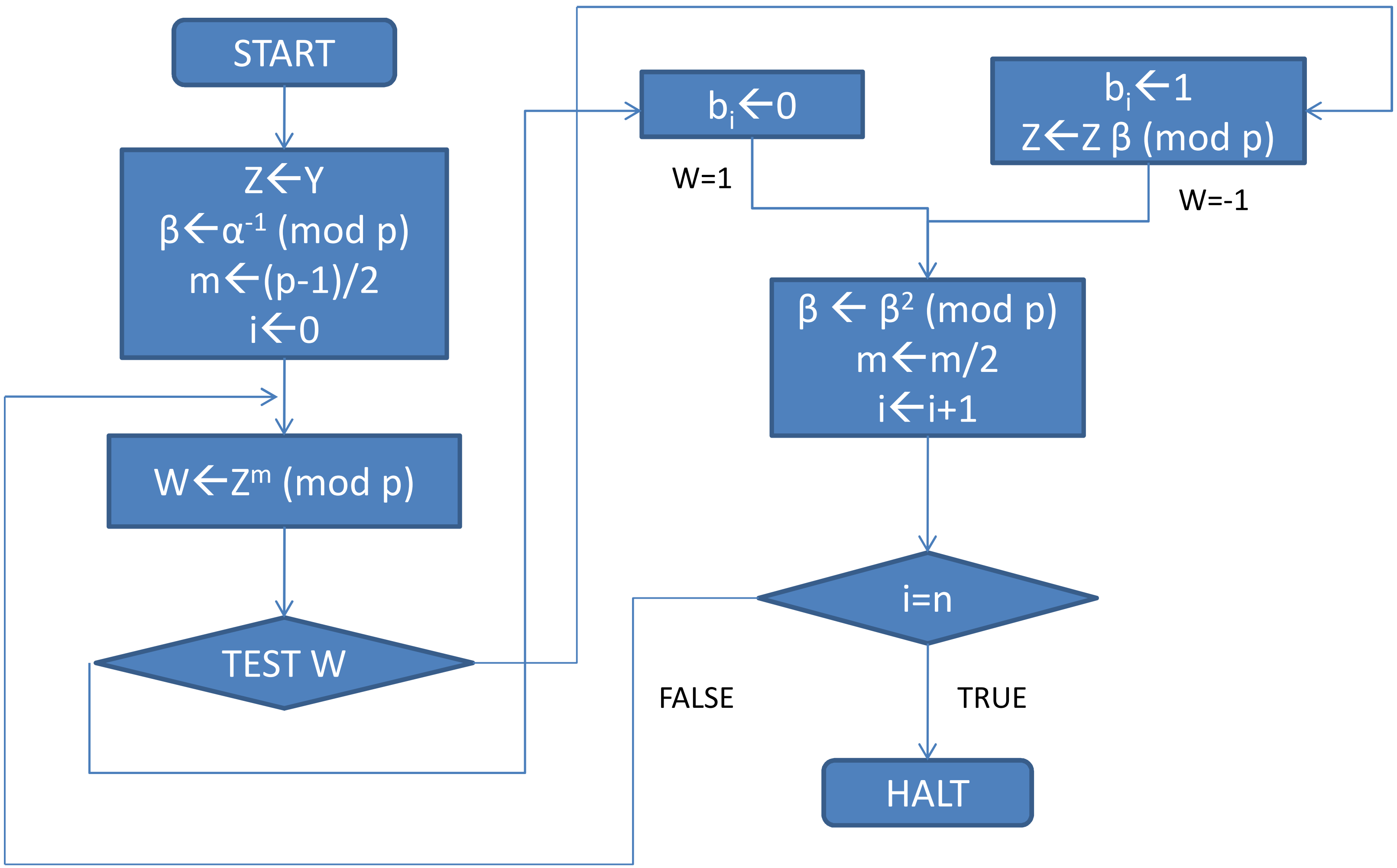
- Remaining bit of *x*

  $$m \equiv {p-1}\Big/{2^{i+1}}$$

  $$z \equiv \alpha^{x_i} \pmod{p}, \quad \text{where } x_i = \sum_{j=i}^{n-1} b_j 2^j$$

  $$z^m \pmod{p} \equiv \begin{cases} +1, & b_i = 0 \\ -1, & b_i = 1 \end{cases}$$

Flowchart for p = 2^n+1



KAIST

An Algorithm for Arbitrary Primes (1)

- Generalize the algorithm to arbitrary primes *p*
  - $2^{16}+1$ is the largest known prime of the form 2^n+1

- Let $p-1 = p_1^{n_1} p_2^{n_2} \cdots p_k^{n_k}$, $p_i < p_{i+1}$

  where the p_i are distinct primes and the n_i are positive integers

- By Chinese Remainder Theorem,

  if the value of $x(\bmod p_i^{n_i})$ is determined for all *i*, then

  $x \ (\bmod \prod_{i=1}^{k} p_i^{n_i}) = x(\bmod p-1) = x$

An Algorithm for Arbitrary Primes (2)

- Consider the following expansion of $x \pmod{p_i^{n_i}}$

$$x \pmod{p_i^{n_i}} = \sum_{j=0}^{n_i - 1} b_j p_i^{\,j} \;, \quad \text{where } 0 \le b_j \le p_i - 1$$

- Then,

$$y^{(p-1)/p_i} \equiv \alpha^{(p-1)x/p_i} \equiv \gamma_i^{\,x} \equiv \gamma_i^{\,b_0} \pmod{p}$$

$$\text{where } \gamma_i = \alpha^{(p-1)/p_i}$$

→ The resultant value uniquely determines b_0

An Algorithm for Arbitrary Primes (3)

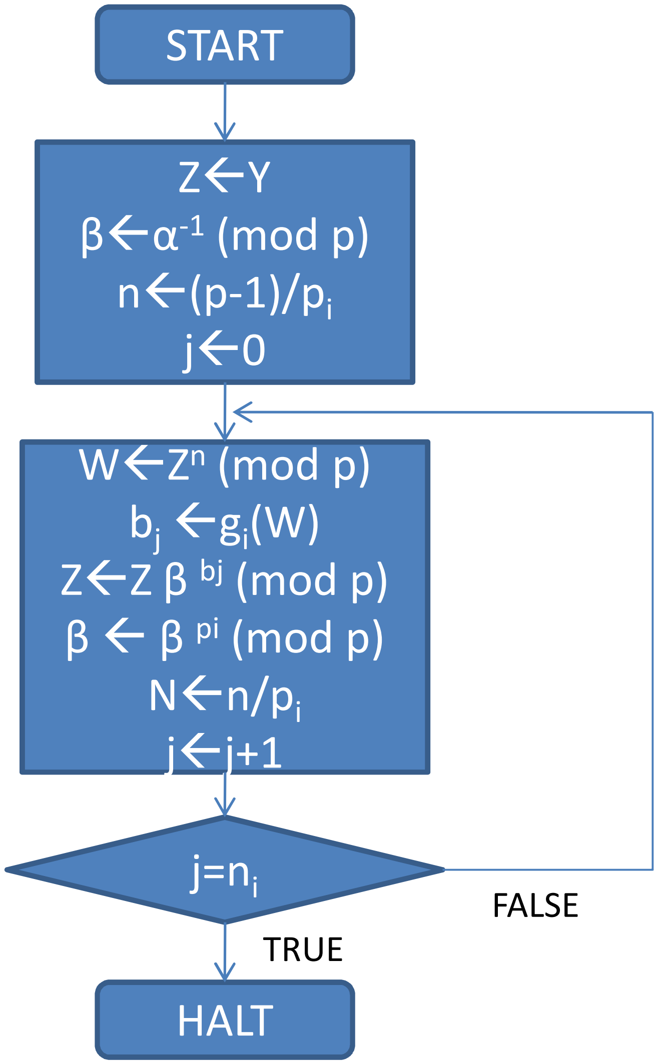
- ✓ The function $g_i(w)$ is defined by

$$\gamma_i^{\,g_i(w)} \equiv w \pmod{p}\,, \quad 0 \le g_i(w) \le p_i - 1$$

- ✓ The resultant value, $y^{(p-1)/p_i} \equiv \gamma_i^{\,b_0} \pmod{p}$ determines b_i by $g_i(w)$

- ✓ So, dominant computational requirement : **computing** $\boldsymbol{g_i(w)}$

Flowchart for arbitrary primes



Time & Space Complexity

Theorem

Let

$$p-1 = p_1^{n_1} p_2^{n_2} \cdots p_k^{n_k}, \qquad p_i < p_{i+1}$$

be the prime factorization of $p-1$, where p is prime, the p_i are distinct primes, and the n_i are positive integer.

Then, for any $\{r_i\}_{i=1}^{k}$ with all $0 \le r_i \le 1$, logarithms over *GF(p)* can be computed in $O\left(\sum_{i=1}^{k} n_i (\log_2 p + p_i^{1-r_i}(1+\log_2 p_i^{r_i}))\right)$ operations with $O\left(\log_2 p \cdot \sum_{i=1}^{k}(1+p_i^{r_i})\right)$ bits of memory.

Proof. [1]

KAIST

Discussion

- ✓ $p = 2^{448} \cdot 5^2 + 1$ is a prime that *p-1* has only small prime factors (i.e., 2, 5)
  - → 2+448 = 450 iterations of the loop in the flowchart
  - **Not one-way function**
  - → Dominant computational requirement: <u>450 exponentiations *mod p*</u>

- ✓ When $p = 2p' + 1$, (*p'* is also prime)

  (ex. $p' = 2^{121} \cdot 5^2 \cdot 7^2 \cdot 11^2 \cdot 13 \cdot 17 \cdot 19 \cdot 23 \cdot 29 \cdot 31 \cdot 37 \cdot 41 \cdot 43 \cdot 47 \cdot 53 \cdot 59 + 1$)
  - → Dominant computational requirement: $g(w)$
  - **One-way function**
  - → computing *g(w)*: <u>more than 10^{30} operations & 10^{30} bits of memory</u> (*r=1/2*)

References

[1] S. C. Pohlig and M. E. Hellman, An Improved Algorithm for Computing Logarithms over GF(p) and Its Cryptographic Significance, IEEE Transactions on Information Theory, 1978