[thm]Lemma [thm]Fact
[thm]Example

Week 5: Discrete Logarithms

Jay Daigle

Occidental College

September 26, 2019

Merkle's Puzzles

Merkle's Puzzles

1. Bob generates N different symmetric keys and attaches an identification code i to each of them.

Merkle's Puzzles

1. Bob generates N different symmetric keys and attaches an identification code i to each of them.
2. For each key, Bob encrypts a message of the form "This is the ith key on my list. The key is K_i." He uses an encryption algorithm that is possible but computationally expensive to brute force.

Merkle's Puzzles

1. Bob generates N different symmetric keys and attaches an identification code i to each of them.
2. For each key, Bob encrypts a message of the form "This is the ith key on my list. The key is K_i." He uses an encryption algorithm that is possible but computationally expensive to brute force.
3. Bob sends Alice all N of the messages generated this way. Alice chooses one at random and brute-force decrypts it, and sends the identifier to Bob.

Merkle's Puzzles

1. Bob generates N different symmetric keys and attaches an identification code i to each of them.
2. For each key, Bob encrypts a message of the form "This is the ith key on my list. The key is K_i." He uses an encryption algorithm that is possible but computationally expensive to brute force.
3. Bob sends Alice all N of the messages generated this way. Alice chooses one at random and brute-force decrypts it, and sends the identifier to Bob.
4. Bob and Alice can now communicate using the symmetric key they have agreed on.

Definition

A number $a \in \mathbb{Z}/m\mathbb{Z}$ is a *unit* modulo m if a has an inverse modulo m.
The set of units is $\mathbb{Z}/m\mathbb{Z}^\times$.
If p prime: $\mathbb{Z}/p\mathbb{Z}^\times = \{1, 2, \ldots, p-1\}$.

Definition

A number $a \in \mathbb{Z}/m\mathbb{Z}$ is a *unit* modulo m if a has an inverse modulo m.
The set of units is $\mathbb{Z}/m\mathbb{Z}^\times$.
If p prime: $\mathbb{Z}/p\mathbb{Z}^\times = \{1, 2, \ldots, p-1\}$.

Definition

A number $g \in \mathbb{Z}/p\mathbb{Z}$ is a *primitive root* modulo p if
$\{g, g^2, g^3, \ldots, g^{p-1}\} = \mathbb{Z}/p\mathbb{Z}^\times$.

Definition

A number $a \in \mathbb{Z}/m\mathbb{Z}$ is a *unit* modulo m if a has an inverse modulo m.
The set of units is $\mathbb{Z}/m\mathbb{Z}^\times$.
If p prime: $\mathbb{Z}/p\mathbb{Z}^\times = \{1, 2, \ldots, p-1\}$.

Definition

A number $g \in \mathbb{Z}/p\mathbb{Z}$ is a *primitive root* modulo p if $\{g, g^2, g^3, \ldots, g^{p-1}\} = \mathbb{Z}/p\mathbb{Z}^\times$.

Fact

If $g \in \mathbb{Z}/p\mathbb{Z}^\times$ *then* $\#\{g^i \mod p : 1 \leq i \leq p-1\} | p-1$.

Definition

A number $a \in \mathbb{Z}/m\mathbb{Z}$ is a *unit* modulo m if a has an inverse modulo m.
The set of units is $\mathbb{Z}/m\mathbb{Z}^\times$.
If p prime: $\mathbb{Z}/p\mathbb{Z}^\times = \{1, 2, \ldots, p-1\}$.

Definition

A number $g \in \mathbb{Z}/p\mathbb{Z}$ is a *primitive root* modulo p if $\{g, g^2, g^3, \ldots, g^{p-1}\} = \mathbb{Z}/p\mathbb{Z}^\times$.

Fact

If $g \in \mathbb{Z}/p\mathbb{Z}^\times$ then $\#\{g^i \mod p : 1 \leq i \leq p-1\} | p-1$.
Thus in particular, if we compute $g, g^2, \ldots, g^{(p-1)/2}$ and we haven't found a number equivalent to 1, then we know g is a primitive root.

Definition

Let p be prime, g a primitive root mod p, and $h \in \mathbb{Z}/p\mathbb{Z}^{\times}$.
If $g^x \equiv h \mod m$, then x is a **discrete logarithm of** h **to the base** g **modulo** m.

Definition

Let p be prime, g a primitive root mod p, and $h \in \mathbb{Z}/p\mathbb{Z}^{\times}$.
If $g^x \equiv h \mod m$, then x is a **discrete logarithm of** h **to the base** g **modulo** m.

Some authors will call this the *index* of h with respect to g, denoted $\mathrm{ind}_g(h)$.

Definition

Let p be prime, g a primitive root mod p, and $h \in \mathbb{Z}/p\mathbb{Z}^{\times}$.
If $g^x \equiv h \mod m$, then x is a **discrete logarithm of** h **to the base** g **modulo** m.

Some authors will call this the *index* of h with respect to g, denoted $\operatorname{ind}_g(h)$.

Fact

1. $\log_g(1) = 0$

Definition

Let p be prime, g a primitive root mod p, and $h \in \mathbb{Z}/p\mathbb{Z}^{\times}$.
If $g^x \equiv h \mod m$, then x is a **discrete logarithm of h to the base g modulo m**.

Some authors will call this the *index* of h with respect to g, denoted $\mathrm{ind}_g(h)$.

Fact

1. $\log_g(1) = 0$
2. $\log_g(ab) \equiv \log_g(a) + \log_g(b) \mod p - 1$

Definition

Let p be prime, g a primitive root mod p, and $h \in \mathbb{Z}/p\mathbb{Z}^\times$.
If $g^x \equiv h \mod m$, then x is a **discrete logarithm of** h **to the base** g **modulo** m.

Some authors will call this the *index* of h with respect to g, denoted $\mathrm{ind}_g(h)$.

Fact

1. $\log_g(1) = 0$
2. $\log_g(ab) \equiv \log_g(a) + \log_g(b) \mod p-1$
3. $\log_g(a^r) \equiv r\log_g(a) \mod p-1.$

Diffie-Hellman Algorithm

Diffie-Hellman Algorithm

Alice and Bob wish to exchange a key. They follow the following steps:

1. Choose a large prime p, and a non-zero integer $g \in \mathbb{Z}/p\mathbb{Z}^{\times}$.

Diffie-Hellman Algorithm

Alice and Bob wish to exchange a key. They follow the following steps:

1. Choose a large prime p, and a non-zero integer $g \in \mathbb{Z}/p\mathbb{Z}^{\times}$.
2. Alice chooses a secret integer a, and Bob chooses a secret integer b. Neither party reveals this integer to anyone.

Diffie-Hellman Algorithm

Alice and Bob wish to exchange a key. They follow the following steps:

1. Choose a large prime p, and a non-zero integer $g \in \mathbb{Z}/p\mathbb{Z}^{\times}$.
2. Alice chooses a secret integer a, and Bob chooses a secret integer b. Neither party reveals this integer to anyone.
3. Alice computes $A \equiv g^a \mod p$ and Bob computes $B \equiv g^b \mod p$, and they (publicly) exchange these values with each other.

Diffie-Hellman Algorithm

Alice and Bob wish to exchange a key. They follow the following steps:

1. Choose a large prime p, and a non-zero integer $g \in \mathbb{Z}/p\mathbb{Z}^{\times}$.
2. Alice chooses a secret integer a, and Bob chooses a secret integer b. Neither party reveals this integer to anyone.
3. Alice computes $A \equiv g^a \mod p$ and Bob computes $B \equiv g^b \mod p$, and they (publicly) exchange these values with each other.
4. Now Alice computes $A' \equiv B^a \mod p$ and Bob computes $B' \equiv A^b \mod p$.

Diffie-Hellman Algorithm

Alice and Bob wish to exchange a key. They follow the following steps:

1. Choose a large prime p, and a non-zero integer $g \in \mathbb{Z}/p\mathbb{Z}^\times$.
2. Alice chooses a secret integer a, and Bob chooses a secret integer b. Neither party reveals this integer to anyone.
3. Alice computes $A \equiv g^a \mod p$ and Bob computes $B \equiv g^b \mod p$, and they (publicly) exchange these values with each other.
4. Now Alice computes $A' \equiv B^a \mod p$ and Bob computes $B' \equiv A^b \mod p$.
5. $A' \equiv B' \mod p$, so Alice and Bob use this shared information as their key.

Fast Exponentiation

Fast Exponentiation

1. Compute g^{2^k} for $2^k \leq a$. That is, compute $g, g^2, g^4, g^8, \ldots, g^{2^k}$. We can do this by repeated squaring, without computing intermediate powers.

Fast Exponentiation

1. Compute g^{2^k} for $2^k \leq a$. That is, compute $g, g^2, g^4, g^8, \ldots, g^{2^k}$. We can do this by repeated squaring, without computing intermediate powers.
2. Now express the exponent a in binary. That is, write $a = c_0 + c_1 \cdot 2 + c_2 \cdot 2^2 + \cdots + c_k 2^k$, where $c_i \in \{0, 1\}$.

Fast Exponentiation

1. Compute g^{2^k} for $2^k \leq a$. That is, compute $g, g^2, g^4, g^8, \ldots, g^{2^k}$. We can do this by repeated squaring, without computing intermediate powers.
2. Now express the exponent a in binary. That is, write $a = c_0 + c_1 \cdot 2 + c_2 \cdot 2^2 + \cdots + c_k 2^k$, where $c_i \in \{0, 1\}$.
3. Now we can compute

$$
\begin{aligned}
g^a = g^{c_0 + c_1 \cdot 2 + c_2 \cdot 2^2 + \cdots + c_k 2^k} &= g^{c_0} g^{c_1 \cdot 2} g^{c_2 \cdot 2^2} \cdots g^{c_k 2^k} \\
&= g^{c_0} (g^2)^{c_1} (g^{2^2})^{c_2} \cdots (g^{2^k})^{c_k}.
\end{aligned}
$$

Fast Exponentiation

1. Compute g^{2^k} for $2^k \leq a$. That is, compute $g, g^2, g^4, g^8, \dots, g^{2^k}$. We can do this by repeated squaring, without computing intermediate powers.
2. Now express the exponent a in binary. That is, write $a = c_0 + c_1 \cdot 2 + c_2 \cdot 2^2 + \cdots + c_k 2^k$, where $c_i \in \{0, 1\}$.
3. Now we can compute

$$
\begin{aligned}
g^a = g^{c_0 + c_1 \cdot 2 + c_2 \cdot 2^2 + \cdots + c_k 2^k} &= g^{c_0} g^{c_1 \cdot 2} g^{c_2 \cdot 2^2} \cdots g^{c_k 2^k} \\
&= g^{c_0} (g^2)^{c_1} (g^{2^2})^{c_2} \cdots (g^{2^k})^{c_k}.
\end{aligned}
$$

But we already know g^{2^i} for each i, and the c_i are all either 0 or 1 so don't involve any computation. So we only have to multiply up to k things together here.

Shanks's Babystep-Giantstep Algorithm

Shanks's Babystep-Giantstep Algorithm

Suppose we have a prime number p and a primitive root g, and an integer A, and we want to find an integer x such that $g^x \equiv A \mod p$.

Shanks's Babystep-Giantstep Algorithm

Suppose we have a prime number p and a primitive root g, and an integer A, and we want to find an integer x such that $g^x \equiv A \mod p$. Then

1. let $n = 1 + \lfloor \sqrt{p} \rfloor$. Thus $n > \sqrt{p}$.

Shanks's Babystep-Giantstep Algorithm

Suppose we have a prime number p and a primitive root g, and an integer A, and we want to find an integer x such that $g^x \equiv A \mod p$. Then

1. let $n = 1 + \lfloor \sqrt{p} \rfloor$. Thus $n > \sqrt{p}$.
2. (Baby steps) Calculate $g^0, g^1, g^2, \ldots, g^n \mod p$. Find an inverse for $g^n \mod p$.

Shanks's Babystep-Giantstep Algorithm

Suppose we have a prime number p and a primitive root g, and an integer A, and we want to find an integer x such that $g^x \equiv A \mod p$. Then

1. let $n = 1 + \lfloor \sqrt{p} \rfloor$. Thus $n > \sqrt{p}$.
2. (Baby steps) Calculate $g^0, g^1, g^2, \ldots, g^n \mod p$. Find an inverse for $g^n \mod p$.
3. (Giant steps) Calculate $A, A \cdot g^{-n}, A \cdot g^{-2n}, \ldots, A \cdot g^{-n^2} \mod p$.

Shanks's Babystep-Giantstep Algorithm

Suppose we have a prime number p and a primitive root g, and an integer A, and we want to find an integer x such that $g^x \equiv A \mod p$. Then

1. let $n = 1 + \lfloor \sqrt{p} \rfloor$. Thus $n > \sqrt{p}$.
2. (Baby steps) Calculate $g^0, g^1, g^2, \dots, g^n \mod p$. Find an inverse for $g^n \mod p$.
3. (Giant steps) Calculate $A, A \cdot g^{-n}, A \cdot g^{-2n}, \dots, A \cdot g^{-n^2} \mod p$.
4. Find a match between these two lists, so that we have $g^i \equiv hg^{-jn}$.

Shanks's Babystep-Giantstep Algorithm

Suppose we have a prime number p and a primitive root g, and an integer A, and we want to find an integer x such that $g^x \equiv A \mod p$. Then

1. let $n = 1 + \lfloor \sqrt{p} \rfloor$. Thus $n > \sqrt{p}$.
2. (Baby steps) Calculate $g^0, g^1, g^2, \ldots, g^n \mod p$. Find an inverse for $g^n \mod p$.
3. (Giant steps) Calculate $A, A \cdot g^{-n}, A \cdot g^{-2n}, \ldots, A \cdot g^{-n^2} \mod p$.
4. Find a match between these two lists, so that we have $g^i \equiv hg^{-jn}$.
5. Then $x = i + jn$ is a solution to $g^x \equiv h \mod p$.