Chapter 7 Discrete Logarithms

What Are Discrete Logarithms?

Let p be a prime number. An integer α is a primitive root modulo p if for every β relatively prime to p there is an integer x such that $\beta \equiv \alpha^x \pmod{p}$. For a fixed α and a given β, an integer x with this property is a discrete logarithm of β base α modulo p.

What Are Discrete Logarithms?

Let p be a prime number. An integer α is a primitive root modulo p if for every β relatively prime to p there is an integer x such that $\beta \equiv \alpha^x \pmod{p}$. For a fixed α and a given β, an integer x with this property is a discrete logarithm of β base α modulo p.

To avoid confusion with ordinary logs, we sometimes call this the index of β base α modulo p.

What Are Discrete Logarithms?

Let p be a prime number. An integer α is a primitive root modulo p if for every β relatively prime to p there is an integer x such that $\beta \equiv \alpha^x \pmod{p}$. For a fixed α and a given β, an integer x with this property is a discrete logarithm of β base α modulo p.

To avoid confusion with ordinary logs, we sometimes call this the index of β base α modulo p.

In general, for many integers n, there is no primitive root modulo n.

Example of Primitive Root

Example

$p = 7, \alpha = 3$

Example of Primitive Root

Example

$p = 7, \alpha = 3$

$$
\begin{aligned}
3^1 &= 3 \\
3^2 &= 9 \equiv 2 \\
3^3 &= 27 \equiv 6 \\
3^4 &= 81 \equiv 4 \\
3^5 &= 243 \equiv 5 \\
3^6 &= 729 \equiv 1
\end{aligned}
$$

Example of Primitive Root

Example

$p = 7, \alpha = 3$

$$\begin{aligned} 3^1 &= 3 \\ 3^2 &= 9 \equiv 2 \\ 3^3 &= 27 \equiv 6 \\ 3^4 &= 81 \equiv 4 \\ 3^5 &= 243 \equiv 5 \\ 3^6 &= 729 \equiv 1 \end{aligned}$$

By taking successive powers, we see that everything non-zero modulo 7 is a power of 3 modulo 7. This would be called brute force verification.

Example of No Primitive Root

In contrast, consider $n = 8$. Candidates would be $1, 3, 5, 7$.

$\alpha = 1$: all powers give 1

Example of No Primitive Root

In contrast, consider $n = 8$. Candidates would be $1, 3, 5, 7$.

$\alpha = 1$: all powers give 1

$\alpha = 3$: all powers produce 1 or 3

Example of No Primitive Root

In contrast, consider $n = 8$. Candidates would be $1, 3, 5, 7$.

$\alpha = 1$: all powers give 1

$\alpha = 3$: all powers produce 1 or 3

$\alpha = 5$: all powers produce 1 or 5

Example of No Primitive Root

In contrast, consider $n = 8$. Candidates would be $1, 3, 5, 7$.

$\alpha = 1$: all powers give 1

$\alpha = 3$: all powers produce 1 or 3

$\alpha = 5$: all powers produce 1 or 5

$\alpha = 7$: all powers produce 7 or 1

Example of No Primitive Root

In contrast, consider $n = 8$. Candidates would be $1, 3, 5, 7$.

$\alpha = 1$: all powers give 1

$\alpha = 3$: all powers produce 1 or 3

$\alpha = 5$: all powers produce 1 or 5

$\alpha = 7$: all powers produce 7 or 1

So, there is no primitive root modulo 8.

Example of No Primitive Root

In contrast, consider $n = 8$. Candidates would be $1, 3, 5, 7$.

$\alpha = 1$: all powers give 1

$\alpha = 3$: all powers produce 1 or 3

$\alpha = 5$: all powers produce 1 or 5

$\alpha = 7$: all powers produce 7 or 1

So, there is no primitive root modulo 8.

Notation: $x = L_\alpha(\beta)$.

A Primitive Root Theorem

Theorem

The only integers n for which there is a primitive root modulo n are those of the form:

1. $n = p^e$ *with an odd prime,* $e \geq 1$
2. $n = 2p^e$ *with an odd prime* p, $e \geq 1$
3. $n = 2, 4$

A Primitive Root Theorem

Theorem

The only integers n for which there is a primitive root modulo n are those of the form:

1. $n = p^e$ *with an odd prime,* $e \geq 1$
2. $n = 2p^e$ *with an odd prime* p, $e \geq 1$
3. $n = 2, 4$

We will not prove this since we are focusing on primes and not general n's.

Using Fermat's Little Theorem

Remember ... $\alpha^{p-1} \equiv 1 \pmod{p}$. This can be a useful tool. In fact, $p-1$ is the smallest integer n such that $\alpha^n \equiv 1 \pmod{p}$. We can use this when trying to solve for x.

Using Fermat's Little Theorem

Remember ... $\alpha^{p-1} \equiv 1 \pmod{p}$. This can be a useful tool. In fact, $p-1$ is the smallest integer n such that $\alpha^n \equiv 1 \pmod{p}$. We can use this when trying to solve for x.

Here is the technique:
We know $\alpha^{p-1} \equiv 1 \pmod{p}$. So, we know that $\alpha^{\frac{p-1}{2}} \equiv -1 \pmod{p}$ which lets us write $\left(\alpha^{\frac{p-1}{2}}\right)^2 \equiv 1 \pmod{p}$.

Using Fermat's Little Theorem

Remember ... $\alpha^{p-1} \equiv 1 \pmod{p}$. This can be a useful tool. In fact, $p-1$ is the smallest integer n such that $\alpha^n \equiv 1 \pmod{p}$. We can use this when trying to solve for x.

Here is the technique:
We know $\alpha^{p-1} \equiv 1 \pmod{p}$. So, we know that $\alpha^{\frac{p-1}{2}} \equiv -1 \pmod{p}$ which lets us write $\left(\alpha^{\frac{p-1}{2}}\right)^2 \equiv 1 \pmod{p}$.

Consider our congruence $\beta \equiv \alpha^x \pmod{p}$. Then we have

$$\beta^{\frac{p-1}{2}} \equiv \alpha^{x\frac{p-1}{2}} \pmod{p} \equiv (-1)^x \pmod{p}$$

If $\beta^{\frac{p-1}{2}} \equiv 1 \pmod{p}$, then x is even.

Example of Solving Congruences

Example

Solve $3^x \equiv 7 \pmod{19}$.

Example of Solving Congruences

Example

Solve $3^x \equiv 7 \pmod{19}$.

$$\beta^{\frac{p-1}{2}} \equiv 7^9 \equiv 1 \pmod{19}$$

Example of Solving Congruences

Example

Solve $3^x \equiv 7 \pmod{19}$.

$$\beta^{\frac{p-1}{2}} \equiv 7^9 \equiv 1 \pmod{19}$$

So we know x must be even. By brute force, we find that $x = 6$.

Example of Solving Congruences

Example

Solve $3^x \equiv 7 \pmod{19}$.

$$\beta^{\frac{p-1}{2}} \equiv 7^9 \equiv 1 \pmod{19}$$

So we know x must be even. By brute force, we find that $x = 6$.

Given α, x, p, these are easy to calculate, But, the problem arises when we want x and are given α, β, p. In fact, for large p, the difficulty is on the same order as the factoring of primes for the RSA algorithm. But, as already seen, for small p, brute force can do the trick.

We Need A Better Way ...

Computing discrete logarithms this way, however, is naive.

We Need A Better Way ...

Computing discrete logarithms this way, however, is naive.

To compute $L_\alpha(\beta)$ in $\mathbb{Z}/p$ for a primitive root $\alpha \pmod p$, compute (all $\pmod p$) $\alpha^1, \alpha^2, \ldots$ until one of the powers of $\alpha \pmod p$ is the target β.

We Need A Better Way ...

Computing discrete logarithms this way, however, is naive.

To compute $L_\alpha(\beta)$ in $\mathbb{Z}/p$ for a primitive root $\alpha \pmod p$, compute (all $\pmod p$) $\alpha^1, \alpha^2, \ldots$ until one of the powers of $\alpha \pmod p$ is the target β.

For p large enough for cryptographic purposes, we'd need p to be 100 digits or more, making brute force infeasible making it necessary to use other methods as this method takes $O(p)$ trials. So, we want to reduce the number of trials.

Baby-Step Giant-Step

This algorithm works in any acyclic group G, assuming the computation of the group operations are not too expensive. This runtime is on the order of the square root of $|G|$, that is $O(\sqrt{|G|})$.

Baby-Step Giant-Step

This algorithm works in any acyclic group G, assuming the computation of the group operations are not too expensive. This runtime is on the order of the square root of $|G|$, that is $O(\sqrt{|G|})$.

Reminder: a cyclic group $G =< a >$ can be generated by a single element.

Baby-Step Giant-Step

This algorithm works in any acyclic group G, assuming the computation of the group operations are not too expensive. This runtime is on the order of the square root of $|G|$, that is $O(\sqrt{|G|})$.

Reminder: a cyclic group $G =< a >$ can be generated by a single element.

Let G be a cyclic group with generator α. We want to compute the discrete logarithm of some other element $\beta \in G$ base α. Let n be the order $|G|$ of G and let $m = \lfloor \sqrt{n} \rfloor$. So, m is an integer, just less than $\sqrt{n}$. When we take $x = L_\alpha(\beta)$, we can write $x = m \cdot i + j$ where $0 \leq i, j < m$.

Baby-Step Giant-Step

The equation $\beta = \alpha^{mi+j}$ can be rearranged to $\beta(\alpha^{-m})^i = \alpha^j$.

Baby-Step Giant-Step

The equation $\beta = \alpha^{mi+j}$ can be rearranged to $\beta(\alpha^{-m})^i = \alpha^j$.

Thus, we first compute the m quantities α^j, $0 \leq j < m$ and put them in some sorted look-up table. Then begin to compute successively the m quantities $\beta, \beta \cdot \alpha^{-m}, \beta \cdot \alpha^{-2m}, \ldots, \beta \cdot \alpha^{-mi}, \ldots$

Baby-Step Giant-Step

The equation $\beta = \alpha^{mi+j}$ can be rearranged to $\beta(\alpha^{-m})^i = \alpha^j$.

Thus, we first compute the m quantities α^j, $0 \leq j < m$ and put them in some sorted look-up table. Then begin to compute successively the m quantities $\beta, \beta \cdot \alpha^{-m}, \beta \cdot \alpha^{-2m}, \ldots, \beta \cdot \alpha^{-mi}, \ldots$

At the i^{th} step compare α^j and $\beta \cdot \alpha^{-mi}$. If they are equal, then $L_\alpha(\beta) = mi + j$.

Baby-Step Giant-Step

The equation $\beta = \alpha^{mi+j}$ can be rearranged to $\beta(\alpha^{-m})^i = \alpha^j$.

Thus, we first compute the m quantities α^j, $0 \leq j < m$ and put them in some sorted look-up table. Then begin to compute successively the m quantities $\beta, \beta \cdot \alpha^{-m}, \beta \cdot \alpha^{-2m}, \ldots, \beta \cdot \alpha^{-mi}, \ldots$

At the i^{th} step compare α^j and $\beta \cdot \alpha^{-mi}$. If they are equal, then $L_\alpha(\beta) = mi + j$.

So this construction of the list of α^j's takes $\sqrt{n}$ steps initially. Each of the $O(\sqrt{n})$ comparisons of $\beta \cdot \alpha^{-mi}$ to the α^j's must be done in much less than $O(\sqrt{n})$ time, or the total number of steps would be $O(n)$, which is no better than brute force.

Baby-Step Giant-Step

The equation $\beta = \alpha^{mi+j}$ can be rearranged to $\beta(\alpha^{-m})^i = \alpha^j$.

Thus, we first compute the m quantities α^j, $0 \leq j < m$ and put them in some sorted look-up table. Then begin to compute successively the m quantities $\beta, \beta \cdot \alpha^{-m}, \beta \cdot \alpha^{-2m}, \ldots, \beta \cdot \alpha^{-mi}, \ldots$

At the i^{th} step compare α^j and $\beta \cdot \alpha^{-mi}$. If they are equal, then $L_\alpha(\beta) = mi + j$.

So this construction of the list of α^j's takes $\sqrt{n}$ steps initially. Each of the $O(\sqrt{n})$ comparisons of $\beta \cdot \alpha^{-mi}$ to the α^j's must be done in much less than $O(\sqrt{n})$ time, or the total number of steps would be $O(n)$, which is no better than brute force.

Therefore, the list of α^j's must be sorted and ordered in a fashion so that testing whether or not an element $\beta \cdot \alpha^{-mi}$ is on the list very fast. For example, in simple cases, this testing can be done in only $L_2(m)$ comparisons.

Baby-Step Giant-Step

Although this algorithm makes sense in any group, the necessary sorting of the α^j's is easiest to describe in a tangible class of examples, such as $\mathbb{Z}/p^*$ for prime p.

Baby-Step Giant-Step

Although this algorithm makes sense in any group, the necessary sorting of the α^j's is easiest to describe in a tangible class of examples, such as $\mathbb{Z}/p^*$ for prime p.

Let α be the primitive root modulo p. The order n of the group $G = \mathbb{Z}/p^*$ is $n = p - 1$. Let m be the greatest integer not greater than $\sqrt{p-1}$. To compute $L_\alpha \beta$ in G, first compute

$$(0, \alpha^0 \pmod p), (1, \alpha^1 \pmod p), (2, \alpha^2 \pmod p), \ldots, (m-1, \alpha^{m-1} \pmod p)$$

Baby-Step Giant-Step

We sort these ordered pairs by size of their second component, so that testing whether some $g \in G$ is in the list will only take $L_2 m$ comparisons. Then, to be a little smart about comparisons,

1. Compute $c = \alpha^{-m} \pmod{p}$

Baby-Step Giant-Step

We sort these ordered pairs by size of their second component, so that testing whether some $g \in G$ is in the list will only take $L_2 m$ comparisons. Then, to be a little smart about comparisons,

1. Compute $c = \alpha^{-m} \pmod{p}$
2. Initialize $x = \beta$

Baby-Step Giant-Step

We sort these ordered pairs by size of their second component, so that testing whether some $g \in G$ is in the list will only take $L_2 m$ comparisons. Then, to be a little smart about comparisons,

1. Compute $c = \alpha^{-m} \pmod{p}$
2. Initialize $x = \beta$
3. For $i = 0$ to $i = m - 1$, if x is the element α^j in the list $\alpha^0, \alpha^1, \ldots, \alpha^{m-1}$ then $L_\alpha \beta = mi + j$

Baby-Step Giant-Step

We sort these ordered pairs by size of their second component, so that testing whether some $g \in G$ is in the list will only take $L_2 m$ comparisons. Then, to be a little smart about comparisons,

1. Compute $c = \alpha^{-m} \pmod{p}$
2. Initialize $x = \beta$
3. For $i = 0$ to $i = m - 1$, if x is the element α^j in the list $\alpha^0, \alpha^1, \ldots, \alpha^{m-1}$ then $L_\alpha \beta = mi + j$

Otherwise replace x by $x \cdot c \pmod{p}$ and $i = i + 1$.

BSGS Example

Example

Compute $L_2 3 \pmod{29}$.

BSGS Example

Example

Compute $L_2 3 \pmod{29}$.

Here, $p = 29, \beta = 3, \alpha = 2$. Then, $m = \lfloor \sqrt{29 - 1} \rfloor = 5$. Thus, we compute $(0, \alpha^0), \ldots, (4, \alpha^4 \pmod{29})$.

$$(0, 1), (1, 2), (2, 4), (3, 8)(4, 16)$$

BSGS Example

Example

Compute $L_2 3 \pmod{29}$.

Here, $p = 29, \beta = 3, \alpha = 2$. Then, $m = \lfloor \sqrt{29 - 1} \rfloor = 5$. Thus, we compute $(0, \alpha^0), \ldots, (4, \alpha^4 \pmod{29})$.

$$(0, 1), (1, 2), (2, 4), (3, 8)(4, 16)$$

$c \equiv 2^{-5} \pmod{29} \equiv 10$.

BSGS Example

Example

Compute $L_2 3 \pmod{29}$.

Here, $p = 29, \beta = 3, \alpha = 2$. Then, $m = \lfloor \sqrt{29 - 1} \rfloor = 5$. Thus, we compute $(0, \alpha^0), \ldots, (4, \alpha^4 \pmod{29})$.

$$(0, 1), (1, 2), (2, 4), (3, 8)(4, 16)$$

$c \equiv 2^{-5} \pmod{29} \equiv 10$.
Initialize $x = \beta = 3$. This is not on the list.

BSGS Example

Example

Compute $L_2 3 \pmod{29}$.

Here, $p = 29, \beta = 3, \alpha = 2$. Then, $m = \lfloor \sqrt{29 - 1} \rfloor = 5$. Thus, we compute $(0, \alpha^0), \ldots, (4, \alpha^4 \pmod{29})$.

$$(0, 1), (1, 2), (2, 4), (3, 8)(4, 16)$$

$c \equiv 2^{-5} \pmod{29} \equiv 10$.
Initialize $x = \beta = 3$. This is not on the list.
Replace x by $x \cdot c$, which is $3 \cdot 10 \pmod{29} = 1$, which is on the list.
That is,

$$3 \cdot 2^{-5} = 2^0$$

BSGS Example

Example

Compute $L_2 3 \pmod{29}$.

Here, $p = 29, \beta = 3, \alpha = 2$. Then, $m = \lfloor \sqrt{29 - 1} \rfloor = 5$. Thus, we compute $(0, \alpha^0), \ldots, (4, \alpha^4 \pmod{29})$.

$$(0, 1), (1, 2), (2, 4), (3, 8)(4, 16)$$

$c \equiv 2^{-5} \pmod{29} \equiv 10$.
Initialize $x = \beta = 3$. This is not on the list.
Replace x by $x \cdot c$, which is $3 \cdot 10 \pmod{29} = 1$, which is on the list.
That is,

$$3 \cdot 2^{-5} = 2^0$$

By rearranging as planned, this gives

$$L_2 3 = 5 \cdot 1 + 0 = 5$$

in $\mathbb{Z}/\mathbb{Z}_{29}^*$.

Another BSGS Example

Example

Compute $L_2 3 \ (mod\ 101)$.

Another BSGS Example

Example

Compute $L_2 3 \ (mod\ 101)$.

Here, we have that $p = 101$, $\alpha = 2$ and $\beta = 3$. Then, $m = \sqrt{101 - 1} = 10$. So, we compute the pairs $(j, 2^j (mod\ 101))$, for $j = 0, \ldots, m - 1 = 9$; this list is

$$(0,1), (1,2), (2,4), (3,8), (4,16), (5,32), (6,64), (7,27), (8,54), (9,7)$$

Sorting by size of the second element, the list is

$$(0,1), (1,2), (2,4), (9,7), (3,8), (4,16), (7,27), (5,32), (8,54), (6,64)$$

Another BSGS Example

Example

Compute $L_2 3 \ (mod\ 101)$.

Here, we have that $p = 101$, $\alpha = 2$ and $\beta = 3$. Then, $m = \sqrt{101 - 1} = 10$. So, we compute the pairs $(j, 2^j (mod\ 101))$, for $j = 0, \ldots, m - 1 = 9$; this list is

$$(0,1), (1,2), (2,4), (3,8), (4,16), (5,32), (6,64), (7,27), (8,54), (9,7)$$

Sorting by size of the second element, the list is

$$(0,1), (1,2), (2,4), (9,7), (3,8), (4,16), (7,27), (5,32), (8,54), (6,64)$$

Also, $c \equiv 2^{-10} (mod\ 101) \equiv 65$. We initialize $x = \beta = 3$. This is not on the list, so we replace x by $x \cdot c$ as long as x is not on the list :

Another BSGS Example

$$
\begin{aligned}
3 \cdot 65((\text{mod } 101) &\equiv 94 \\
94 \cdot 65(\text{mod } 101) &\equiv 50 \\
50 \cdot 65(\text{mod } 101) &\equiv 18 \\
18 \cdot 65(\text{mod } 101) &\equiv 59 \\
59 \cdot 65(\text{mod } 101) &\equiv 98 \\
98 \cdot 65(\text{mod } 101) &\equiv 7
\end{aligned}
$$

Another BSGS Example

$$
\begin{aligned}
3 \cdot 65((\text{mod } 101) &\equiv 94 \\
94 \cdot 65(\text{mod } 101) &\equiv 50 \\
50 \cdot 65(\text{mod } 101) &\equiv 18 \\
18 \cdot 65(\text{mod } 101) &\equiv 59 \\
59 \cdot 65(\text{mod } 101) &\equiv 98 \\
98 \cdot 65(\text{mod } 101) &\equiv 7
\end{aligned}
$$

That is,

$$3 \cdot (2^{-10})^6 \equiv 7 \equiv 2^9 (mod\ 101)$$

Rearranging as planned, this gives

$$L_2(3) = 10 \cdot 6 + 9 = 69 \text{ in } \mathbb{Z}/\mathbb{Z}^*_{101}$$

The Index Calculus

This is the best known method for computing discrete logarithms in cyclic groups G. This algorithm cannot be applied effectively to all cyclic groups, but when it can be applied it is a sub-exponential time algorithm. It is probabilistic.

The Index Calculus

This is the best known method for computing discrete logarithms in cyclic groups G. This algorithm cannot be applied effectively to all cyclic groups, but when it can be applied it is a sub-exponential time algorithm. It is probabilistic.

Note: At the outset, the subset S of G used in the algorithm, the so-called factor group, must be chosen well. Unfortunately, it is not generally clear how to make a good choice, or even that there is a good choice of factor basis. Timing of the algorithm amounts to adjustment of the choice of factor basis, but in general there may be no good choice to achieve sub-exponential runtime.

Factor Groups

Definition

A subgroup H of a group G is normal if its left and right cosets coincide; that is, if H is a normal subgroup of G, then

$$gH = Hg \; \forall \; g \in G$$

In other words, the left and right cosets are the same. This can also be defined in terms of conjugates, which is to say H is a normal subgroup of G if for all $g \in G$,

$$gHg^{-1} = H$$

The notation we use for normal subgroups is $H \triangleleft G$.

Factor Groups

Definition

Let H be a normal subgroup of a group G. Then G/H, read "G mod H" is the system whose set of elements is

$$G/H = \{aH \mid a \in G\}$$

and whose operation is defined by the rule $aHbH = abH$.
In other words, the factor group is the set of all left cosets of H in G.
We often use N for the normal subgroup.

The Index Calculus

Let α be a generator of a cyclic group G of order $n = |G|$. Let β be another element of G, whose logarithm base α we want to compute.

The Index Calculus

Let α be a generator of a cyclic group G of order $n = |G|$. Let β be another element of G, whose logarithm base α we want to compute.

1. Choose a factor base: Choose a 'small' subset s of G, called the factor basis

$$S = \{p_1, \ldots, p_k\}$$

The Index Calculus

Let α be a generator of a cyclic group G of order $n = |G|$. Let β be another element of G, whose logarithm base α we want to compute.

1. Choose a factor base: Choose a 'small' subset s of G, called the factor basis

$$S = \{p_1, \ldots, p_k\}$$

2. Collect relations: Choose random integers in the range $0 \leq k \leq n-1$ and try to express α^k as a product of elements of S. If such a relation exists

$$\alpha^k = p_1^{k_1} \ldots p_t^{k_t}$$

take logarithms of both sides to obtain

$$k = k_1 \cdot L_\alpha p_1 + \ldots + k_t \cdot L_\alpha p_t (\text{mod } n)$$

Repeat this until there is somewhat more than t relations.

The Index Calculus

3. Find logarithms $L_\alpha p_i$: Solve the system of relations to find the logarithms base α of the elements of the factor basis. If the system is indeterminate, go back and generate more relations.

The Index Calculus

3. Find logarithms $L_\alpha p_i$: Solve the system of relations to find the logarithms base α of the elements of the factor basis. If the system is indeterminate, go back and generate more relations.
4. Finding other logarithms: Given $\beta \in G$, pick a random integer k and try to express $\beta \cdot \alpha^k$ in the form

$$\beta \cdot \alpha^k = p_1^{k_1} \cdots p_t^{kt}$$

If successful. solve for

$$L_\alpha \beta = ((k_1 \cdot L_\alpha p_1 + \ldots + k_t \cdot L_\alpha p_k) - k) (\text{mod } n)$$

If unsuccessful, choose a different k and repeat.

Notes

1. Certainly the precomputation of logarithms of elements of the factor basis need not be repeated for computations of varying elements of the group.

Notes

1. Certainly the precomputation of logarithms of elements of the factor basis need not be repeated for computations of varying elements of the group.
2. An examination of the algorithm makes it clear the factor basis S must be chosen so that 'many' of the elements of G are expressed as products of powers of elements of S with 'small' exponents. Of course, this idea and its effects must be quantified to optimize the algorithm in any particular case.

Notes

1. Certainly the precomputation of logarithms of elements of the factor basis need not be repeated for computations of varying elements of the group.
2. An examination of the algorithm makes it clear the factor basis S must be chosen so that 'many' of the elements of G are expressed as products of powers of elements of S with 'small' exponents. Of course, this idea and its effects must be quantified to optimize the algorithm in any particular case.
3. When the cyclic group in question is $G = \mathbb{Z}/p^*$, or prime p, the factor basis should be the first t primes, Of course, choosing a wise t is the issue. Testing whether a randomly produced group element is a product of elements from the factor basis (with small positive integer exponents) can be done very quickly by trial division by elements of the factor basis.

Using the Index Calculus

Example

Let $G = \mathbb{Z}/\mathbb{Z}_{29}^*$. The order of the group is 28. Use the primitive root $\alpha = 11$ Try a factor basis $S = \{2, 3, 5\}$. Start generating relations, by considering 'random' powers of $\alpha (\text{mod } 29)$. Bear in mind that we want to solve the system of relations modulo 28.

$$\alpha^2 (\text{mod } 29) = 11^2 (\text{mod } 29) \equiv 5$$

Using the Index Calculus

Example

Let $G = \mathbb{Z}/\mathbb{Z}_{29}^*$. The order of the group is 28. Use the primitive root $\alpha = 11$ Try a factor basis $S = \{2, 3, 5\}$. Start generating relations, by considering 'random' powers of $\alpha (\text{mod } 29)$. Bear in mind that we want to solve the system of relations modulo 28.

$$\alpha^2 (\text{mod } 29) = 11^2 (\text{mod } 29) \equiv 5$$
$$\alpha^3 (\text{mod } 29) = 11^3 (\text{mod } 29) \equiv 26 \ (\textit{fail})$$

Using the Index Calculus

Example

Let $G = \mathbb{Z}/\mathbb{Z}_{29}^*$. The order of the group is 28. Use the primitive root $\alpha = 11$ Try a factor basis $S = \{2, 3, 5\}$. Start generating relations, by considering 'random' powers of $\alpha (\text{mod } 29)$. Bear in mind that we want to solve the system of relations modulo 28.

$$\alpha^2 (\text{mod } 29) = 11^2 (\text{mod } 29) \equiv 5$$
$$\alpha^3 (\text{mod } 29) = 11^3 (\text{mod } 29) \equiv 26 \ (\textit{fail})$$
$$\alpha^4 (\text{mod } 29) = 11^4 (\text{mod } 29) \equiv 14 \ (\textit{fail})$$

Using the Index Calculus

Example

Let $G = \mathbb{Z}/\mathbb{Z}_{29}^*$. The order of the group is 28. Use the primitive root $\alpha = 11$ Try a factor basis $S = \{2, 3, 5\}$. Start generating relations, by considering 'random' powers of $\alpha (\text{mod } 29)$. Bear in mind that we want to solve the system of relations modulo 28.

$$\alpha^2 (\text{mod } 29) = 11^2 (\text{mod } 29) \equiv 5$$
$$\alpha^3 (\text{mod } 29) = 11^3 (\text{mod } 29) \equiv 26 \ (\textit{fail})$$
$$\alpha^4 (\text{mod } 29) = 11^4 (\text{mod } 29) \equiv 14 \ (\textit{fail})$$
$$\alpha^6 (\text{mod } 29) = 11^6 (\text{mod } 29) \equiv 9 = 3^2$$

Using the Index Calculus

Example

Let $G = \mathbb{Z}/\mathbb{Z}_{29}^*$. The order of the group is 28. Use the primitive root $\alpha = 11$ Try a factor basis $S = \{2, 3, 5\}$. Start generating relations, by considering 'random' powers of $\alpha \pmod{29}$. Bear in mind that we want to solve the system of relations modulo 28.

$$\alpha^2 \pmod{29} = 11^2 \pmod{29} \equiv 5$$
$$\alpha^3 \pmod{29} = 11^3 \pmod{29} \equiv 26 \ (\textit{fail})$$
$$\alpha^4 \pmod{29} = 11^4 \pmod{29} \equiv 14 \ (\textit{fail})$$
$$\alpha^6 \pmod{29} = 11^6 \pmod{29} \equiv 9 = 3^2$$
$$\alpha^7 \pmod{29} = 11^7 \pmod{29} \equiv 12 = 2^2 \cdot 3$$

Using the Index Calculus

Example

Let $G = \mathbb{Z}/\mathbb{Z}_{29}^*$. The order of the group is 28. Use the primitive root $\alpha = 11$ Try a factor basis $S = \{2, 3, 5\}$. Start generating relations, by considering 'random' powers of $\alpha (\text{mod } 29)$. Bear in mind that we want to solve the system of relations modulo 28.

$$\alpha^2 (\text{mod } 29) = 11^2 (\text{mod } 29) \equiv 5$$
$$\alpha^3 (\text{mod } 29) = 11^3 (\text{mod } 29) \equiv 26 \ (\textit{fail})$$
$$\alpha^4 (\text{mod } 29) = 11^4 (\text{mod } 29) \equiv 14 \ (\textit{fail})$$
$$\alpha^6 (\text{mod } 29) = 11^6 (\text{mod } 29) \equiv 9 = 3^2$$
$$\alpha^7 (\text{mod } 29) = 11^7 (\text{mod } 29) \equiv 12 = 2^2 \cdot 3$$
$$\alpha^9 (\text{mod } 29) = 11^9 (\text{mod } 29) \equiv 2$$

Using the Index Calculus

Example

Let $G = \mathbb{Z}/\mathbb{Z}_{29}^*$. The order of the group is 28. Use the primitive root $\alpha = 11$ Try a factor basis $S = \{2, 3, 5\}$. Start generating relations, by considering 'random' powers of $\alpha (\text{mod } 29)$. Bear in mind that we want to solve the system of relations modulo 28.

$$\begin{aligned}
\alpha^2 (\text{mod } 29) = 11^2 (\text{mod } 29) &\equiv 5 \\
\alpha^3 (\text{mod } 29) = 11^3 (\text{mod } 29) &\equiv 26 \ (\textit{fail}) \\
\alpha^4 (\text{mod } 29) = 11^4 (\text{mod } 29) &\equiv 14 \ (\textit{fail}) \\
\alpha^6 (\text{mod } 29) = 11^6 (\text{mod } 29) &\equiv 9 = 3^2 \\
\alpha^7 (\text{mod } 29) = 11^7 (\text{mod } 29) &\equiv 12 = 2^2 \cdot 3 \\
\alpha^9 (\text{mod } 29) = 11^9 (\text{mod } 29) &\equiv 2
\end{aligned}$$

Solving the system modulo 28 to obtain the logarithms of the factor base happens to be easy in this case.

Using the Index Calculus

- From the first relation, we directly obtain $L_{11}5 = 2$.

Using the Index Calculus

- From the first relation, we directly obtain $L_{11}5 = 2$.
- The fourth relation gives $6 = 2 \cdot L_{11}3 (\text{mod } 28)$, which does not uniquely determine the log of 3 since the coefficient in front has a common factor with the modulus 28.

Using the Index Calculus

- From the first relation, we directly obtain $L_{11}5 = 2$.
- The fourth relation gives $6 = 2 \cdot L_{11}3 (\text{mod } 28)$, which does not uniquely determine the log of 3 since the coefficient in front has a common factor with the modulus 28.
- The last relation gives $L_{11}2 = 9$

Using the Index Calculus

- From the first relation, we directly obtain $L_{11}5 = 2$.
- The fourth relation gives $6 = 2 \cdot L_{11}3 (\text{mod } 28)$, which does not uniquely determine the log of 3 since the coefficient in front has a common factor with the modulus 28.
- The last relation gives $L_{11}2 = 9$
- We can use the next to last relation

$$7 = 2 \cdot L_{11}2 + L_{11}3 (\text{mod } 28)$$

to obtain $L_{11}3 = 17$.

Using the Index Calculus

Where does this last one come from?

$$\begin{aligned} 11^7 \pmod{28} &\equiv 2^2 \cdot 3 \\ \Rightarrow 7 &= L_{11} 2^2 \cdot 3 \\ \Rightarrow 7 &= 2 \cdot L_{11} 2 + L_{11} 3 \end{aligned}$$

Using the Index Calculus

Where does this last one come from?

$$\begin{aligned} 11^7 (\text{mod } 28) &\equiv 2^2 \cdot 3 \\ \Rightarrow 7 &= L_{11}2^2 \cdot 3 \\ \Rightarrow 7 &= 2 \cdot L_{11}2 + L_{11}3 \end{aligned}$$

And now, since $L_{11}2 = 9$, we can solve for $L_{11}3$.

$$\begin{aligned} 7 &= 2 \cdot L_{11}2 + L_{11}3 \\ 7 &= 2 \cdot 9 + L_{11}3 \\ -11 &= L_{11}3 \\ L_{11}3 &= 17 \end{aligned}$$

Using the Index Calculus

This finishes the precomputation for the factor basis $\{2, 3, 5\}$.

Using the Index Calculus

This finishes the precomputation for the factor basis $\{2, 3, 5\}$.

To compute $L_{11}7$, for example, multiply the target $\beta = 7$ by 'random' powers of $\alpha \equiv 11 (\text{mod } 29)$ and look for results expressible in terms of the factor basis.

Using the Index Calculus

This finishes the precomputation for the factor basis $\{2, 3, 5\}$.

To compute $L_{11}7$, for example, multiply the target $\beta = 7$ by 'random' powers of $\alpha \equiv 11 (\text{mod } 29)$ and look for results expressible in terms of the factor basis.

$$\beta \cdot \alpha (\text{mod } 29) \equiv 7 \cdot 11 (\text{mod } 29) = 19$$

which is a fail.

Using the Index Calculus

This finishes the precomputation for the factor basis $\{2, 3, 5\}$.

To compute $L_{11}7$, for example, multiply the target $\beta = 7$ by 'random' powers of $\alpha \equiv 11 (\text{mod } 29)$ and look for results expressible in terms of the factor basis.

$$\beta \cdot \alpha (\text{mod } 29) \equiv 7 \cdot 11 (\text{mod } 29) = 19$$

which is a fail.

$$\beta \cdot \alpha^2 (\text{mod } 29) \equiv 7 \cdot 11^2 (\text{mod } 29) \equiv 6 = 2 \cdot 3$$

Using the Index Calculus

This finishes the precomputation for the factor basis $\{2, 3, 5\}$.

To compute $L_{11}7$, for example, multiply the target $\beta = 7$ by 'random' powers of $\alpha \equiv 11 (\text{mod } 29)$ and look for results expressible in terms of the factor basis.

$$\beta \cdot \alpha (\text{mod } 29) \equiv 7 \cdot 11 (\text{mod } 29) = 19$$

which is a fail.

$$\beta \cdot \alpha^2 (\text{mod } 29) \equiv 7 \cdot 11^2 (\text{mod } 29) \equiv 6 = 2 \cdot 3$$

Therefore,

$$L_{11}7 = L_{11}2 + L_{11}3 - 2 = 9 + 17 - 2 = 24$$

Using the Index Calculus

This finishes the precomputation for the factor basis $\{2, 3, 5\}$.

To compute $L_{11}7$, for example, multiply the target $\beta = 7$ by 'random' powers of $\alpha \equiv 11 (\text{mod } 29)$ and look for results expressible in terms of the factor basis.

$$\beta \cdot \alpha (\text{mod } 29) \equiv 7 \cdot 11 (\text{mod } 29) = 19$$

which is a fail.

$$\beta \cdot \alpha^2 (\text{mod } 29) \equiv 7 \cdot 11^2 (\text{mod } 29) \equiv 6 = 2 \cdot 3$$

Therefore,

$$L_{11}7 = L_{11}2 + L_{11}3 - 2 = 9 + 17 - 2 = 24$$

Note: This algorithm becomes noticeably more efficient than the others for considerably larger groups.

Diffie-Hellman Key Exchange

This is both the original public-key idea and an important mechanism in current use. The practical point is that symmetric ciphers are generally much faster than asymmetric ones (for both hardware and software reasons), so the public-key cipher is merely used to set up a private key, known as a session key, intended to be used only for a single conversation.

Diffie-Hellman Key Exchange

This is both the original public-key idea and an important mechanism in current use. The practical point is that symmetric ciphers are generally much faster than asymmetric ones (for both hardware and software reasons), so the public-key cipher is merely used to set up a private key, known as a session key, intended to be used only for a single conversation.

This protocol makes use of the particulars involving discrete logarithms, and of course, in relatively difficulty in computing discrete logarithms. Therefore, it not only works for the $\mathbb{Z}/\mathbb{Z}_p$ discrete logarithm situations, but also generalizes to arbitrary finite fields, elliptical curves, or any other group structure where logarithms make sense.

Diffie-Hellman Key Exchange

- First, the sender and receiver need to agree on a large prime and a primitive element (root) $\alpha (\text{mod } p)$. These need not be kept secret and can be shared by a group of users.

Diffie-Hellman Key Exchange

- First, the sender and receiver need to agree on a large prime and a primitive element (root) $\alpha (\text{mod } p)$. These need not be kept secret and can be shared by a group of users.
- The sender chooses a large random integer X, privately computes $X \equiv \alpha^x (\text{mod } p)$ and sends X to a receiver by a possibly insecure channel.

Diffie-Hellman Key Exchange

- First, the sender and receiver need to agree on a large prime and a primitive element (root) $\alpha \pmod{p}$. These need not be kept secret and can be shared by a group of users.
- The sender chooses a large random integer X, privately computes $X \equiv \alpha^x \pmod{p}$ and sends X to a receiver by a possibly insecure channel.
- Meanwhile, the receiver similarly chooses a large random integer Y, privately computes $Y \equiv \alpha^y \pmod{p}$ and sends Y to the sender across the possibly insecure channel.

Diffie-Hellman Key Exchange

- Then, the sender privately computes $k = Y^x \pmod{p}$ and the receiver symmetrically computes $k' \equiv X^y \pmod{p}$.

Diffie-Hellman Key Exchange

- Then, the sender privately computes $k = Y^x \pmod p$ and the receiver symmetrically computes $k' \equiv X^y \pmod p$.

In fact, $k \equiv k' \pmod p$ as follows:

Computing modulo p, we have

$$k' = X^y = (\alpha^x)^y = \alpha^{xy} = (\alpha^y)^x = k \pmod p$$

Diffie-Hellman Key Exchange

- Then, the sender privately computes $k = Y^x \pmod p$ and the receiver symmetrically computes $k' \equiv X^y \pmod p$.

In fact, $k \equiv k' \pmod p$ as follows:

Computing modulo p, we have

$$k' = X^y = (\alpha^x)^y = \alpha^{xy} = (\alpha^y)^x = k \pmod p$$

But no one else on the network obtains k unless they can compute discrete logarithms. Then, the key $k(= k')$ can be used directly or indirectly as a key for the asymmetric cipher.

Diffie-Hellman Key Exchange Security

For security, the prime modulus should be large, as the security of the Diffie-Hellman key exchange lies in the fact that, while it is relatively easy to calculate exponentials modulo p, it is very difficult to calculate discrete logarithms. For large primes, the latter task is considered infeasible.

Diffie-Hellman Example

Example

This key exchange is based on the use of the prime number $p = 353$ and a primitive root $\alpha = 3$. A and B select secret keys $x = 97$ and $y = 233$. Each computes its public key:

Diffie-Hellman Example

Example

This key exchange is based on the use of the prime number $p = 353$ and a primitive root $\alpha = 3$. A and B select secret keys $x = 97$ and $y = 233$. Each computes its public key:
A computes $X \equiv 3^{97} (\text{mod } 353) \equiv 40$

Diffie-Hellman Example

Example

This key exchange is based on the use of the prime number $p = 353$ and a primitive root $\alpha = 3$. A and B select secret keys $x = 97$ and $y = 233$. Each computes its public key:

A computes $X \equiv 3^{97} (\text{mod } 353) \equiv 40$

B computes $Y \equiv 3^{233} (\text{mod } 353) \equiv 248$

Diffie-Hellman Example

Example

This key exchange is based on the use of the prime number $p = 353$ and a primitive root $\alpha = 3$. A and B select secret keys $x = 97$ and $y = 233$. Each computes its public key:

A computes $X \equiv 3^{97} (\text{mod } 353) \equiv 40$

B computes $Y \equiv 3^{233} (\text{mod } 353) \equiv 248$

After they exchange public keys, they can each compute the common secret key:

Diffie-Hellman Example

Example

This key exchange is based on the use of the prime number $p = 353$ and a primitive root $\alpha = 3$. A and B select secret keys $x = 97$ and $y = 233$. Each computes its public key:

A computes $X \equiv 3^{97} (\text{mod } 353) \equiv 40$

B computes $Y \equiv 3^{233} (\text{mod } 353) \equiv 248$

After they exchange public keys, they can each compute the common secret key:

A computes $K = (Y)^x (\text{mod } 353) \equiv 248^{97} (\text{mod } 353) \equiv 160$

Diffie-Hellman Example

Example

This key exchange is based on the use of the prime number $p = 353$ and a primitive root $\alpha = 3$. A and B select secret keys $x = 97$ and $y = 233$. Each computes its public key:

A computes $X \equiv 3^{97} \pmod{353} \equiv 40$

B computes $Y \equiv 3^{233} \pmod{353} \equiv 248$

After they exchange public keys, they can each compute the common secret key:

A computes $K = (Y)^x \pmod{353} \equiv 248^{97} \pmod{353} \equiv 160$

B computes $K = (X)^y \pmod{353} \equiv 40^{233} \pmod{353} \equiv 160.$

Diffie-Hellman Example

Example

This key exchange is based on the use of the prime number $p = 353$ and a primitive root $\alpha = 3$. A and B select secret keys $x = 97$ and $y = 233$. Each computes its public key:

A computes $X \equiv 3^{97} (\text{mod } 353) \equiv 40$

B computes $Y \equiv 3^{233} (\text{mod } 353) \equiv 248$

After they exchange public keys, they can each compute the common secret key:

A computes $K = (Y)^x (\text{mod } 353) \equiv 248^{97} (\text{mod } 353) \equiv 160$

B computes $K = (X)^y (\text{mod } 353) \equiv 40^{233} (\text{mod } 353) \equiv 160$.

We assume that a hacker would have that $p = 353$, $X = 40$, $Y = 248$ and $\alpha = 3$.

Diffie-Hellman Example

Example

This key exchange is based on the use of the prime number $p = 353$ and a primitive root $\alpha = 3$. A and B select secret keys $x = 97$ and $y = 233$. Each computes its public key:

A computes $X \equiv 3^{97} (\text{mod } 353) \equiv 40$

B computes $Y \equiv 3^{233} (\text{mod } 353) \equiv 248$

After they exchange public keys, they can each compute the common secret key:

A computes $K = (Y)^x (\text{mod } 353) \equiv 248^{97} (\text{mod } 353) \equiv 160$

B computes $K = (X)^y (\text{mod } 353) \equiv 40^{233} (\text{mod } 353) \equiv 160$.

We assume that a hacker would have that $p = 353$, $X = 40$, $Y = 248$ and $\alpha = 3$.

A brute force attack could work here since we are dealing with a relatively small prime. We would want to find x such that $3^x (\text{mod } 353) \equiv 40$ or $3^x (\text{mod } 353) \equiv 248$. The desired result is reached when the exponent value of 97, as $3^{97} (\text{mod } 353) \equiv 40$ is obtained.

Man-in-the-Middle Attack

Suppose A and B want to share a message and D is their adversary.

1. D prepares for the attack by generating two random private keys X_{D_1} and X_{D_2} and computes the corresponding public keys Y_{D_1} and Y_{D_2}.

Man-in-the-Middle Attack

Suppose A and B want to share a message and D is their adversary.

1. D prepares for the attack by generating two random private keys X_{D_1} and X_{D_2} and computes the corresponding public keys Y_{D_1} and Y_{D_2}.
2. A transmits X to B.

Man-in-the-Middle Attack

Suppose A and B want to share a message and D is their adversary.

1. D prepares for the attack by generating two random private keys X_{D_1} and X_{D_2} and computes the corresponding public keys Y_{D_1} and Y_{D_2}.
2. A transmits X to B.
3. D intercepts X and transmits Y_{D_1} to B. D also calculates $K_2 = (X)^{X_{D_2}} \pmod{p}$.

Man-in-the-Middle Attack

Suppose A and B want to share a message and D is their adversary.

1. D prepares for the attack by generating two random private keys X_{D_1} and X_{D_2} and computes the corresponding public keys Y_{D_1} and Y_{D_2}.
2. A transmits X to B.
3. D intercepts X and transmits Y_{D_1} to B. D also calculates $K_2 = (X)^{X_{D_2}} (\text{mod } p)$.
4. B receives Y_{D_1} and calculates $K_1 = (Y_{D_1})^y (\text{mod } p)$.

Man-in-the-Middle Attack

Suppose A and B want to share a message and D is their adversary.

1. D prepares for the attack by generating two random private keys X_{D_1} and X_{D_2} and computes the corresponding public keys Y_{D_1} and Y_{D_2}.
2. A transmits X to B.
3. D intercepts X and transmits Y_{D_1} to B. D also calculates $K_2 = (X)^{X_{D_2}} (\mathrm{mod}\, p)$.
4. B receives Y_{D_1} and calculates $K_1 = (Y_{D_1})^y (\mathrm{mod}\, p)$.
5. B transmits Y to A.

Man-in-the-Middle Attack

Suppose A and B want to share a message and D is their adversary.

1. D prepares for the attack by generating two random private keys X_{D_1} and X_{D_2} and computes the corresponding public keys Y_{D_1} and Y_{D_2}.
2. A transmits X to B.
3. D intercepts X and transmits Y_{D_1} to B. D also calculates $K_2 = (X)^{X_{D_2}} \pmod{p}$.
4. B receives Y_{D_1} and calculates $K_1 = (Y_{D_1})^{y} \pmod{p}$.
5. B transmits Y to A.
6. D intercepts Y and transmits Y_{D_2} to A. D calculates $K_1 \equiv (Y)^{X_{D_1}} \pmod{p}$.

Man-in-the-Middle Attack

Suppose A and B want to share a message and D is their adversary.

1. D prepares for the attack by generating two random private keys X_{D_1} and X_{D_2} and computes the corresponding public keys Y_{D_1} and Y_{D_2}.
2. A transmits X to B.
3. D intercepts X and transmits Y_{D_1} to B. D also calculates $K_2 = (X)^{X_{D_2}} \pmod{p}$.
4. B receives Y_{D_1} and calculates $K_1 = (Y_{D_1})^{y} \pmod{p}$.
5. B transmits Y to A.
6. D intercepts Y and transmits Y_{D_2} to A. D calculates $K_1 \equiv (Y)^{X_{D_1}} \pmod{p}$.
7. A receives Y_{D_2} and calculates $K_2 \equiv (Y_{D_1})^{x} \pmod{p}$.

Man-in-the-Middle Attack

Suppose A and B want to share a message and D is their adversary.

1. D prepares for the attack by generating two random private keys X_{D_1} and X_{D_2} and computes the corresponding public keys Y_{D_1} and Y_{D_2}.
2. A transmits X to B.
3. D intercepts X and transmits Y_{D_1} to B. D also calculates $K_2 = (X)^{X_{D_2}} \pmod{p}$.
4. B receives Y_{D_1} and calculates $K_1 = (Y_{D_1})^{y} \pmod{p}$.
5. B transmits Y to A.
6. D intercepts Y and transmits Y_{D_2} to A. D calculates $K_1 \equiv (Y)^{X_{D_1}} \pmod{p}$.
7. A receives Y_{D_2} and calculates $K_2 \equiv (Y_{D_1})^{x} \pmod{p}$.

At this point, A and B think they share a secret key, but instead B and D share K_1 and a and D share K_2. All future communications between A and B are now compromised.

Man-in-the-Middle Attack

The Man-in-the-Middle attack is the most effective attack on the Diffie-Hellman key exchange.

Man-in-the-Middle Attack

The Man-in-the-Middle attack is the most effective attack on the Diffie-Hellman key exchange.

The key exchange is vulnerable to such an attack because it does not authenticate participants. If digital signatures and public-key certificates are included, then this attack is overcome.

ElGamal Cryptosystem

In 1984, Taher Elgamal announced a public-key scheme based on discrete logarithms closely related to the Diffie-Hellman technique. The ElGamal cryptosystem is used in some form in a number of standards.

ElGamal Cryptosystem

In 1984, Taher Elgamal announced a public-key scheme based on discrete logarithms closely related to the Diffie-Hellman technique. The ElGamal cryptosystem is used in some form in a number of standards.

As with Diffie-Hellman, the global elements of ElGamal are the prime numbers p and α, which is a primitive root of p.

How ElGamal Works

A generates a private/public key pair as follows:

1. Generate a random integer X_A such that $1 < X_A < p - 1$.

How ElGamal Works

A generates a private/public key pair as follows:

1. Generate a random integer X_A such that $1 < X_A < p - 1$.
2. Compute $Y_A \equiv \alpha^{X_A} \pmod{p}$.

How ElGamal Works

A generates a private/public key pair as follows:

1. Generate a random integer X_A such that $1 < X_A < p - 1$.
2. Compute $Y_A \equiv \alpha^{X_A} \pmod{p}$.
3. A's private key is X_A; A's public key is $\{p, \alpha, Y_A\}$

How ElGamal Works

Any user B that has access to A's public key can encrypt a message as follows:

1. Represent the message as an integer M in the range of $0 \leq M \leq p - 1$. Longer messages are represented as a sequence of blocks, with each block being an integer less than p.

How ElGamal Works

Any user B that has access to A's public key can encrypt a message as follows:

1. Represent the message as an integer M in the range of $0 \leq M \leq p-1$. Longer messages are represented as a sequence of blocks, with each block being an integer less than p.
2. Choose a random integer k such that $1 \leq k \leq p-1$.

How ElGamal Works

Any user B that has access to A's public key can encrypt a message as follows:

1. Represent the message as an integer M in the range of $0 \leq M \leq p-1$. Longer messages are represented as a sequence of blocks, with each block being an integer less than p.
2. Choose a random integer k such that $1 \leq k \leq p-1$.
3. Compute a one-time key $K \equiv (Y_A)^k \pmod{p}$.

How ElGamal Works

Any user B that has access to A's public key can encrypt a message as follows:

1. Represent the message as an integer M in the range of $0 \leq M \leq p-1$. Longer messages are represented as a sequence of blocks, with each block being an integer less than p.
2. Choose a random integer k such that $1 \leq k \leq p-1$.
3. Compute a one-time key $K \equiv (Y_A)^k \pmod{p}$.
4. Encrypt M as the pair of integers (C_1, C_2) where

$$C_1 \equiv \alpha^k \pmod{p}$$

How ElGamal Works

Any user B that has access to A's public key can encrypt a message as follows:

1. Represent the message as an integer M in the range of $0 \leq M \leq p-1$. Longer messages are represented as a sequence of blocks, with each block being an integer less than p.
2. Choose a random integer k such that $1 \leq k \leq p-1$.
3. Compute a one-time key $K \equiv (Y_A)^k (\mathrm{mod}\, p)$.
4. Encrypt M as the pair of integers (C_1, C_2) where

$$C_1 \equiv \alpha^k (\mathrm{mod}\, p)$$
$$C_2 \equiv KM (\mathrm{mod}\, p)$$

How ElGamal Works

User A recovers the plaintext as follows:

1. Recover the key by computing $K \equiv (C_1)^{X_A} \pmod{p}$.

How ElGamal Works

User A recovers the plaintext as follows:

1. Recover the key by computing $K \equiv (C_1)^{X_A} (\mathrm{mod}\, p)$.
2. Compute $M \equiv (C_2 K^{-1}) (\mathrm{mod}\, p)$.

How ElGamal Works

Here is why ElGamal works: First, let's show how K is recovered by the decryption process:

$$\begin{aligned} K &\equiv (Y_A)^k \pmod{p} \\ K &\equiv (\alpha^{X_A} \pmod{p})^k \pmod{p} \\ K &\equiv \alpha^{kX_A} \pmod{p} \\ K &\equiv (C_1)^{X_A} \pmod{p} \end{aligned}$$

How ElGamal Works

Here is why ElGamal works: First, let's show how K is recovered by the decryption process:

$$\begin{aligned} K &\equiv (Y_A)^k \pmod{p} \\ K &\equiv (\alpha^{X_A} \pmod{p})^k \pmod{p} \\ K &\equiv \alpha^{kX_A} \pmod{p} \\ K &\equiv (C_1)^{X_A} \pmod{p} \end{aligned}$$

Next, using K, we recover the plaintext as:

$$\begin{aligned} C_2 &\equiv KM \pmod{p} \\ (C_2 K^{-1}) \pmod{p} &\equiv KMK^{-1} \pmod{p} \\ &\equiv M \pmod{p} \\ &\equiv M \end{aligned}$$

Example Using ElGamal

Example

Let's start with the prime field $GF(19)$. That is, $p = 19$. It has primitive roots $\{2, 3, 10, 13, 14, 15\}$. Choose $\alpha = 10$.

Example Using ElGamal

Example

Let's start with the prime field $GF(19)$. That is, $p = 19$. It has primitive roots $\{2, 3, 10, 13, 14, 15\}$. Choose $\alpha = 10$.

A generates a key pair as follows:

1. A chooses $X_A = 5$

Example Using ElGamal

Example

Let's start with the prime field $GF(19)$. That is, $p = 19$. It has primitive roots $\{2, 3, 10, 13, 14, 15\}$. Choose $\alpha = 10$.

A generates a key pair as follows:

1. A chooses $X_A = 5$
2. Then $Y_A \equiv \alpha^{X_A} \pmod{p} \equiv 10^5 \pmod{19} \equiv 3$.

Example Using ElGamal

Example

Let's start with the prime field $GF(19)$. That is, $p = 19$. It has primitive roots $\{2, 3, 10, 13, 14, 15\}$. Choose $\alpha = 10$.

A generates a key pair as follows:

1. A chooses $X_A = 5$
2. Then $Y_A \equiv \alpha^{X_A} (\text{mod } p) \equiv 10^5 (\text{mod } 19) \equiv 3$.
3. A's private key is 5; A's public key is $\{p, \alpha, Y_A\} = \{19, 10, 3\}$.

Example Using ElGamal

Suppose B wants to send the message with the value $M = 17$. Then:

1. B chooses $k = 6$.

Example Using ElGamal

Suppose B wants to send the message with the value $M = 17$. Then:

1. B chooses $k = 6$.
2. Then, $K \equiv (Y_A)^k \pmod{p} = 3^6 \pmod{19} \equiv 729 \pmod{19} \equiv 7$

Example Using ElGamal

Suppose B wants to send the message with the value $M = 17$. Then:

1. B chooses $k = 6$.
2. Then, $K \equiv (Y_A)^k \pmod{p} = 3^6 \pmod{19} \equiv 729 \pmod{19} \equiv 7$
3. So, we have
   - $C_1 \equiv \alpha^k \pmod{p} \equiv 10^6 \pmod{19} \equiv 11$

Example Using ElGamal

Suppose B wants to send the message with the value $M = 17$. Then:

1. B chooses $k = 6$.
2. Then, $K \equiv (Y_A)^k \pmod{p} = 3^6 \pmod{19} \equiv 729 \pmod{19} \equiv 7$
3. So, we have
   - $C_1 \equiv \alpha^k \pmod{p} \equiv 10^6 \pmod{19} \equiv 11$
   - $C_2 \equiv KM \pmod{p} \equiv 7 \cdot 17 \pmod{19} \equiv 119 \pmod{19} \equiv 5$

Example Using ElGamal

Suppose B wants to send the message with the value $M = 17$. Then:

1. B chooses $k = 6$.
2. Then, $K \equiv (Y_A)^k \pmod{p} = 3^6 \pmod{19} \equiv 729 \pmod{19} \equiv 7$
3. So, we have
   - $C_1 \equiv \alpha^k \pmod{p} \equiv 10^6 \pmod{19} \equiv 11$
   - $C_2 \equiv KM \pmod{p} \equiv 7 \cdot 17 \pmod{19} \equiv 119 \pmod{19} \equiv 5$
4. B sends the ciphertext $11, 5$.

Example Using ElGamal

For decryption:

1. A calculates

$$\begin{aligned} K &\equiv (C_1)^{X_A} (\text{mod } p) \\ &\equiv 11^5 (\text{mod } 19) \\ &\equiv 161051 (\text{mod } 19) \\ &\equiv 7 \end{aligned}$$

Example Using ElGamal

For decryption:

1. A calculates

$$\begin{aligned} K &\equiv (C_1)^{X_A} (\text{mod } p) \\ &\equiv 11^5 (\text{mod } 19) \\ &\equiv 161051 (\text{mod } 19) \\ &\equiv 7 \end{aligned}$$

2. Then K^{-1} in $GF(19) \equiv 11$

Example Using ElGamal

For decryption:

1. A calculates

$$\begin{aligned} K &\equiv (C_1)^{X_A} (\text{mod } p) \\ &\equiv 11^5 (\text{mod } 19) \\ &\equiv 161051 (\text{mod } 19) \\ &\equiv 7 \end{aligned}$$

2. Then K^{-1} in $GF(19) \equiv 11$
3. Finally, we have

$$\begin{aligned} M &\equiv (C_2 K^{-1}) (\text{mod } p) \\ &\equiv 5 \cdot 11 (\text{mod } 19) \\ &\equiv 55 (\text{mod } 19) \\ &\equiv 17 \end{aligned}$$