CS489/698 Privacy, Cryptography, Network and Data Security

Discrete Logarithm, Diffie-Hellman, ElGamal

Spring 2024, Monday/Wednesday 11:30am-12:50pm

Groups?

Groups - Sets with specific properties

A **group** is a set of elements (usually numbers) that are related to each other according to well-defined operations.

- Consider a group of prime order **q**, or Z_q^*
  - This boils down to the set of non-zero integers between 1 and q-1 modulo **q** → A finite group
  - For q = 5, we have group Z_5^* = {1,2,3,4}
  - In this group, operations are carried out mod 5:
    - 3 * 4 = 12 mod 5 = 2
    - 2^3 = 2 * 2 * 2 = 8 mod 5 = 3

Group axioms

To be a group, these sets should respect some axioms

- Closure
- Identity existence
- Associativity
- Inverse existence

- Groups can also be commutative and cyclic (up next)

Let's take a look at some of these axioms (using multiplication as the operation)

Closure

- For every x,y in the group, x * y is in the group
  - i.e., the multiplication of two group elements falls within the group too
- Example:
  - in Z_5^*, 2* 3 = 6 mod 5 = 1

Identity Existence

- There is an element **e** such that e * x = x * e = x
  - i.e., has an element **e** such that any element times **e** outputs the element itself

- Example:
  - In any Z_q^*, the identity element is 1
  - For Z_5^* : 1 * 3 = 3 mod 5 = 3

Associativity

- For any x, y, z in the group, (x * y) * z = x * (y * z)

- Example:
  - For Z_5^* : (2 * 3) * 4 = 1 * 4 = 2 * (3 * 4) = 2 * 2 = 4

Inverse Existence

- For any **x** in the group, there is a **y** such that x * y = y * x = 1

- Example:
  - For Z_5^* : 2 * 3 = 1 , 3 * 2 = 1 (2 and 3 are inverses)
  - 4 * 4 = 16 mod 5 = 1 (4 is its own inverse)

Abelian Groups

- Abelian groups are groups which are **commutative**
- This means that x * y = y * x for any group elements x and y

- Example:
  - For Z_5^* : 3 * 4 = 2 , 4 * 3 = 2

Cyclic groups

- A group is called **cyclic** if there is at least one element **g** such that its powers (g^1, g^2, g^3, …) mod p span all distinct group elements.
  - **g** is called the "generator" of the group

- Example:
  - For Z_5^*, there are two generators (2 and 3):
    - $2^1 = 2$, $2^2 = 4$, $2^3 = 3$, $2^4 = 1$
    - $3^1 = 3$, $3^2 = 4$, $3^3 = 2$, $3^4 = 1$

Cyclic subgroups

- We can have cyclic **subgroups** within larger finite groups

- Example:
    - Given field F_{607}, we can consider a cyclic subgroup of order p=5 as Z_5^*:

Discrete Logarithm Problem

The Discrete Logarithm Problem

$$h = g^x \text{ , find } x$$

The Discrete Logarithm Problem

$$h = g^x \text{ , find } x$$

Discrete: we are dealing with integers instead of real numbers

Logarithm: we are looking for the logarithm of **x** base **g**

- e.g., $\log_2 256 = 8$, since $2^8 = 256$

The Discrete Logarithm Problem

Given $(g,h) \in \mathbf{G} \times \mathbf{G}$, find $x \in \mathbf{Z}_q^*$ such that:

$$h = g^x$$

Here, **G** is a multiplicative group of prime order **q**, just like we saw during the examples. (But **q** is thousands of bits long)

Solutions to the Discrete Logarithm Problem?

If there's one solution, there are infinitely many

(thank you Fermat's little theorem and modular arithmetic "wrap-around")

How to solve DLP in cyclic groups of prime order?

- Is the group cyclic, finite, and abelian?

Has a generator that spans all elements

Has a limited number of elements

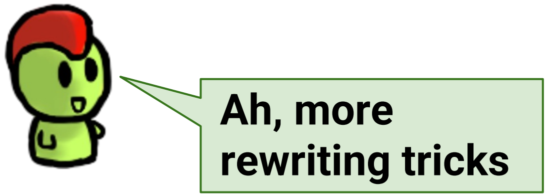
Multiplication is commutative

Baby-step/Giant-step algorithms!!!

Baby-Step/Giant-Step Algorithm?

- A cyclic group **G** = <g> which has prime order **p**
- $h \in G$, goal: find x (mod p) such that $h = g^x$

- Every element **x** ∈ G can be written as: **x** = $i + j*\lceil sqrt(p) \rceil$
  - For integers m, i, j satisfying $0 \le i, j \le m$.



Then:

$$h = g^{j + j*\lceil sqrt(p) \rceil}$$

$$g^i = h \,.\, (g^{-\lceil sqrt(p) \rceil})^j$$

Baby-Step/Giant-Step Algorithm? Notation.

- **$\log_g x \bmod p$** is obtained by comparing two lists:

$$g^i = h \,.\, (g^{-\lceil sqrt(p) \rceil})^j$$

When we find a coincidence, the equality holds and then $x = i + j * \lceil sqrt(p) \rceil$

Can we divide and conquer?

Baby-step/Giant-Step Algorithm

$g^i = h \,.\, (g^{-\lceil sqrt(p) \rceil})^j$

1. $x = i + j * \lceil sqrt(p) \rceil$

$g^i = h . (g^{-\lceil sqrt(p) \rceil})^j$

Baby-step/Giant-Step Algorithm

1. $x = i + j * \lceil sqrt(p) \rceil$

2. $0 \leq i, j < \lceil sqrt(p) \rceil$

Since 0≤x≤p, …

$g^i = h \,.\, (g^{-\lceil sqrt(p) \rceil})^j$

Baby-step/Giant-Step Algorithm

1. $x = i + j * \lceil sqrt(p) \rceil$
2. $0 \leq i, j < \lceil sqrt(p) \rceil$
3. Baby-step: $g_i \longleftarrow g^i$ for $0 \leq i < \lceil sqrt(p) \rceil$

Let's build some tables!

$g^i = h \,.\, (g^{-\lceil sqrt(p) \rceil})^j$

Baby-step/Giant-Step Algorithm

1. $x = i + j * \lceil sqrt(p) \rceil$
2. $0 \leq i, j < \lceil sqrt(p) \rceil$
3. Baby-step: $g_i \longleftarrow g^i$ for $0 \leq i < \lceil sqrt(p) \rceil$

Produces pairs: (g_i, i)

$g^i = h \cdot (g^{-\lceil sqrt(p) \rceil})^j$

Baby-step/Giant-Step Algorithm

1. $x = i + j * \lceil sqrt(p) \rceil$
2. $0 \leq i, j < \lceil sqrt(p) \rceil$
3. Baby-step: $g_i \longleftarrow g^i$ for $0 \leq i < \lceil sqrt(p) \rceil$
4. Giant-step: $h_j \longleftarrow h * g^{-j \lceil sqrt(p) \rceil}$, for $0 \leq j < \lceil sqrt(p) \rceil$

Produces pairs: (h_j, j)

Baby-step/Giant-Step Algorithm

$g^i = h \,.\, (g^{-\lceil \mathrm{sqrt}(p) \rceil})^j$

1. $x = i + j * \lceil \mathrm{sqrt}(p) \rceil$
2. $0 \leq i, j < \lceil \mathrm{sqrt}(p) \rceil$
3. Baby-step: $g_i \longleftarrow g^i$ for $0 \leq i < \lceil \mathrm{sqrt}(p) \rceil$
4. Giant-step: $h_j \longleftarrow h * g^{-j \lceil \mathrm{sqrt}(p) \rceil}$, for $0 \leq j < \lceil \mathrm{sqrt}(p) \rceil$

Produces pairs: (h_j, j)

Overall time and space $O(\mathrm{sqrt}(p))$

Baby-step/Giant-Step Alg m

1. x = i + j*[sqrt(p)]
2. 0≤ i, j < [sqrt(p)]
4. Gia sqrt(p)]

(h_j, j)

Note: For DLP in group G to be "difficult enough" (e.g., 2^{128} operations), needs prime order subgroup of size greater than 2^{256}

verall time a space O(sqrt(p))

DLP Example, $182 = 64^x \pmod{607}$

- Consider the subgroup of order 101 (Z_{101}^*) in $\boldsymbol{F}_{607}$, generated by g=64

i	64^i (mod 607)	*i*	“ ”
0		6	
1		7	
2		8	
3		9	
4		10	
5		-	

Baby-step: $g_i \longleftarrow g^i$ for $0 \le i < \lceil \mathrm{sqrt}(p) \rceil$

g = 64
$\lceil \mathrm{sqrt}(p) \rceil = 11$

DLP Example, $182 = 64^x \pmod{607}$

i	$64^i \pmod{607}$	i	" "
0	1	6	330
1	64	7	482
2	454	8	498
3	527	9	308
4	343	10	288
5	100	-	

Baby-step: $g_i \longleftarrow g^i$ for $0 \leq i < \lceil sqrt(p) \rceil$

$g = 64$

$\lceil sqrt(p) \rceil = 11$

DLP Example, $182 = 64^x (\text{mod } 607)$

Giant-step: $h_j \longleftarrow h * g^{-j \lceil sqrt(p) \rceil}$

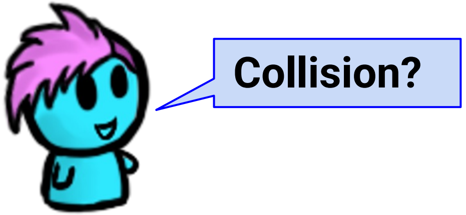
$g = 64$

$\lceil sqrt(p) \rceil = 11$

i	$182 * 64^{-11*j}$ (mod 607)	*i*	
0		6	
1		7	
2		8	
3		9	
4		10	
5		-	

DLP Example, $182 = 64^x \pmod{607}$

i		i	$64^i \pmod{607}$
0	1	6	330
1	64	7	482
2	454	8	498
3	527	9	308
4	343	10	288
5	100	-	



j		j	$182 \cdot 64^{-11 \cdot j} \pmod{607}$
0	182	6	60
1	143	7	394
2	69	8	483
3	271	9	76
4	343	10	580
5	573	-	

DLP Example, $182 = 64^x \pmod{607}$

i		i	$64^i \pmod{607}$
0	1	6	330
1	64	7	482
2	454	8	498
3	527	9	308
4	343	10	288
5	100	-	

Collision?

j		j	$182 \cdot 64^{-11 \cdot j} \pmod{607}$
0	182	6	60
1	143	7	394
2	69	8	483
3	271	9	76
4	343	10	580
5	573	-	

DLP Example, $182 = 64^x (\text{mod } 607)$

i		i	64^i (mod 607)
0	1	6	330
1	64	7	482
2	454	8	498
3	527	9	308
4	**343**	10	288
5	1[illegible]		

j		j	$182 * 64^{-11*j}$ (mod 607)
0	182	6	60
1	143	7	394
2	69	8	483
3	271	9	76
4	**343**	10	580

Match when **i=4** and **j=4**.

(i is not necessarily equal to j, but it happened on this run ¯_(ツ)_/¯

DLP Example, $182 = 64^x \pmod{607}$

i		i	$64^i \pmod{607}$
0	1	6	330
1	64	7	482
2	454	8	498
3	527	9	308
4	343	10	288
5	10		

Collision?

j		j	$182 * 64^{-11*j} \pmod{607}$
0	182	6	60
1	143	7	394
2	69	8	483
3	271	9	76
4	343	10	580

So: $x = 4 + 4*11 = 48$.

DLP Example, $182 = 64^x \pmod{607}$

i		*i*	64^i (mod 607)
0	1	6	330
1	64	7	482
2	454	8	498
3	527	9	308
4	343	10	288
5	100		

j		*j*	$182 \cdot 64^{-11 \cdot j}$ (mod 607)
0	182	6	60
1	143	7	394
2	69	8	483
3	271	9	76
4	343	10	580

So: $x = 4 + 4 \cdot 11 = 48$.

Diffie-Hellman

Diffie-Hellman Key Exchange

A public-key protocol published in 1976 by Whitfield Diffie and Martin Hellman

Allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure channel

Key used to encrypt subsequent communications using a symmetric key cipher

Diffie-Hellman Key Exchange

- Used for establishing a shared secret (lacks authentication; we'll see why this is bad)
- Assume as public parameters generator **g** and prime **p**
- Alice (resp. Bob) generates private value **a** (resp. **b**)

Diffie-Hellman Key Exchange

- Used for establishing a shared secret (lacks authentication; we'll see why this is bad)
- Assume as public parameters generator **g** and prime **p**
- Alice (resp. Bob) generates private value **a** (resp. **b**)

$(A = g^a \bmod p, g, p)$ →

← $(B = g^b \bmod p)$

$B^a = (g^b)^a = g^{ba}$

$A^b = (g^a)^b = g^{ab}$

Alice and Bob can derive the same value by exchanging public values and combining them with their private ones!

Diffie-Hellman Key Exchange

- Used for establishing a <u>shared secret</u> (lacks authentication; we'll see why this is bad)
- Assume as public parameters generator **g** and prime **p**
- Alice (resp. Bob) generates private value **a** (resp. **b**)

$A = g^a \bmod p$ →

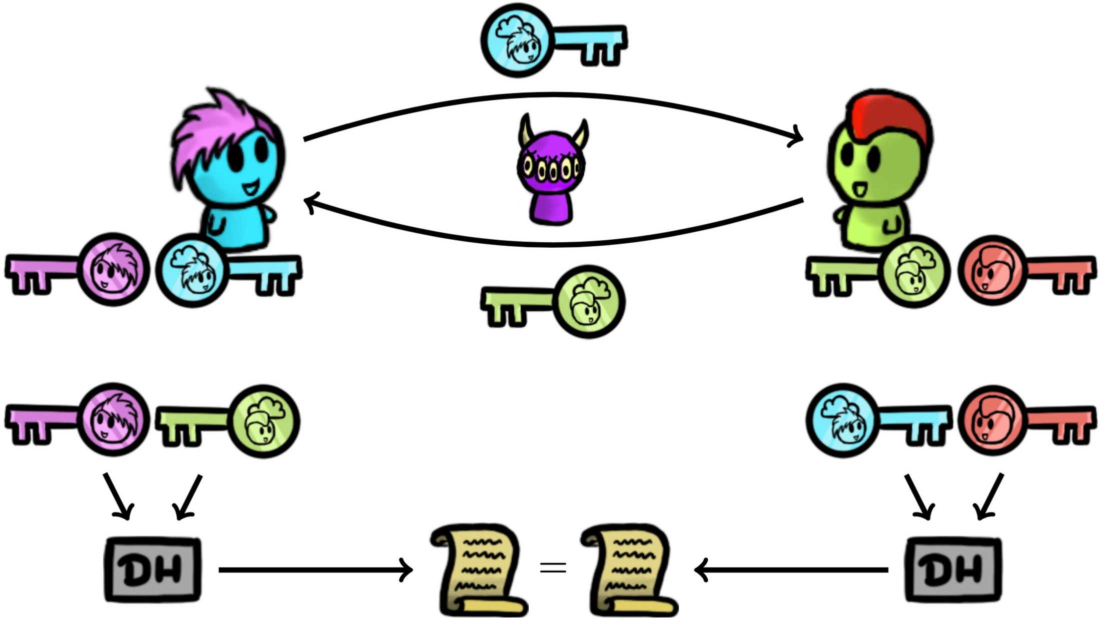
← $B = g^b \bmod p$

$B^a = (g^b)^a = g^{ba}$

$A^b = (g^a)^b = g^{ab}$

Resist keying temptation: the shared value should not <u>immediately</u> be used as a key. G^{ab} is a random element inside a group, but not necessarily a random bit string

Diffie-Hellman Key Exchange – Visualization



Diffie-Hellman relies on the DLP

DH can be broken by recovering the private value **a** from the public value $\mathbf{g^a}$

The adversary <u>must not</u> be able to solve the DLP

The Decisional Diffie-Hellman Problem

Given $\mathbf{g}$, $\mathbf{g}^a$, $\mathbf{g}^b$ distinguish $\mathbf{g}^{ab}$ from random $\mathbf{g}^c$

- An adversary should be unable to learn nothing about the secret $\mathbf{g}^{ab}$ after observing public values $\mathbf{g}^a$ and $\mathbf{g}^b$
    - Assume $\mathbf{g}^{ab}$ and $\mathbf{g}^c$ occur with the same probability

The Decisional Diffie-Hellman Problem

Given $\mathbf{g}$, $\mathbf{g^a}$, $\mathbf{g^b}$ distinguish $\mathbf{g^{ab}}$ from random $\mathbf{g^c}$

- An adversary should be unable to learn nothing about the secret $\mathbf{g^{ab}}$ after observing public values $\mathbf{g^a}$ and $\mathbf{g^b}$
    - Assume $\mathbf{g^{ab}}$ and $\mathbf{g^c}$ occur with the same probability

Useful assumption **beyond** DH key exchange! → **ElGamal** relies on the DDH assumption

ElGamal

- **1985 by Taher ElGamal**

ElGamal Public Key Cryptosystem

- Let ***p*** be a prime such that the DLP in $(\mathbf{Z}_p^*, .)$ is infeasible
- Let α be a generator in $\mathbf{Z}_p^*$
- ***PubK*** $=\{(p, \alpha, a, \beta): \beta \equiv \alpha^a \pmod{p}\}$

- For message **m** and secret random **k** in $\mathbf{Z}_{p-1}$:
  - $e_K(m,k) = (y_1, y_2)$, where $\mathbf{y_1} = \alpha^k \bmod p$ and $\mathbf{y_2} = m\beta^k \bmod p$

- For y_1, y_2 in $\mathbf{Z}_p^*$:
  - $d_K(y_1, y_2) = y_2(y_1^a)^{-1} \bmod p$

ElGamal: The Keys

1. Bob picks a “large” prime **p** and a generator **α**.
   a. Assume message m is an integer $0 < m < p$
2. Bob picks secret integer **a**
3. Bob computes $\boldsymbol{\beta} \equiv \alpha^a \pmod{p}$

ElGamal: The Keys

1. Bob picks a “large” prime **p** and a generator **α**.
    a. Assume message m is an integer $0 < m < p$
2. Bob picks secret integer **a**
3. Bob computes $\boldsymbol{\beta} \equiv \alpha^a \pmod{p}$
4. Bob’s public key is **(p, α, β)**

ElGamal: The Keys

1. Bob picks a “large” prime **p** and a generator **α**.
    a. Assume message m is an integer $0 < m < p$

2. Bob picks secret integer **a**
3. Bob computes $\boldsymbol{\beta} \equiv \alpha^a \pmod{p}$
4. Bob’s public key is **(p, α, β)** 
5. Bob’s private key is **a** 

ElGamal: Encryption

Bob's $Pub_K \rightarrow$ **(p, α, β)**

Bob's $Priv_K \rightarrow$ **a**

$\boldsymbol{\beta} \equiv \alpha^a \pmod{p}$

I choose secret integer **k**

ElGamal: Encryption

Bob's $Pub_K \rightarrow$ **(p, α, β)**

Bob's $Priv_K \rightarrow$ **a**

$\beta \equiv \alpha^a \pmod{p}$

I choose secret integer **k**

Compute $\mathbf{y_1} \equiv \alpha^k \pmod{p}$

Bob's Pub_K → **(p, α, β)**

Bob's $Priv_K$ → **a**

$\beta \equiv \alpha^a \pmod p$

ElGamal: Encryption

I choose secret integer **k**

Compute $\mathbf{y_1} \equiv \alpha^k \pmod p$

Compute $\mathbf{y_2} \equiv \beta^k\, m \pmod p$

Bob's Pub_K → $(\mathbf{p}, \boldsymbol{\alpha}, \boldsymbol{\beta})$

Bob's $Priv_K$ → $\mathbf{a}$

$\boldsymbol{\beta} \equiv \alpha^a \pmod{p}$

ElGamal: Encryption

I choose secret integer **k**

Compute $\mathbf{y_1} \equiv \alpha^{\mathbf{k}} \pmod{p}$

Compute $\mathbf{y_2} \equiv \beta^k m \pmod{p}$

Send $\mathbf{y_1}$ and $\mathbf{y_2}$ to Bob

Bob's $Pub_K \rightarrow$ **(p, α, β)**

Bob's $Priv_K \rightarrow$ **a**

$\boldsymbol{\beta} \equiv \alpha^a \pmod{p}$

ElGamal: Decryption

I choose secret integer **k**

Compute $\mathbf{y_1} \equiv \alpha^{\mathbf{k}} \pmod{p}$

Compute $\mathbf{y_2} \equiv \beta^k m \pmod{p}$

Compute $\mathbf{y_1 y_2^{-a}} \equiv m \pmod{p}$

Send $\mathbf{y_1}$ and $\mathbf{y_2}$ to Bob

Bob's $Pub_K \rightarrow$ **(p, α, β)**
Bob's $Priv_K \rightarrow$ **a**
$\boldsymbol{\beta} \equiv \alpha^a \pmod p$

ElGamal: Decryption

I choose secret integer **k**

Compute $\mathbf{y_1} \equiv \alpha^{\mathbf{k}} \pmod p$

Compute $\mathbf{y_2} \equiv \beta^k m \pmod p$

Compute $\mathbf{y_1 y_2^{-a}} \equiv m \pmod p$

Send $\mathbf{y_1}$ and $\mathbf{y_2}$ to Bob

Bob can <u>decrypt</u> since:
$\mathbf{y_2 y_1^{-a}} \equiv \beta^k m (\alpha^k)^{-a} \equiv m \pmod p$

ElGamal Informal Summary

- The plaintext m is “hidden” by multiplying it by β^k to get y_2

I receive ct = (y_1,y_2)

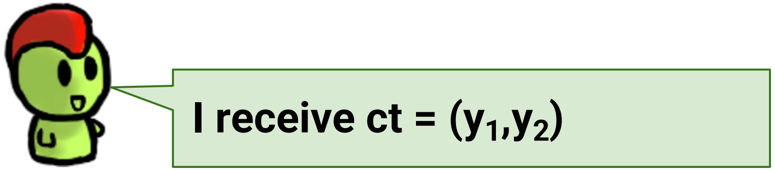
ElGamal Informal Summary

- The plaintext m is “hidden” by multiplying it by β^k to get y_2
- The ciphertext includes α^k so that Bob can compute β^k from α^k (because Bob knows a)

I receive ct = (y_1,y_2)

ElGamal Informal Summary

- The plaintext m is “hidden” by multiplying it by β^k to get y_2
- The ciphertext includes α^k so that Bob can compute β^k from α^k (because Bob knows a)
- Thus, Bob can “reveal” m by dividing y_2 by β^k



ElGamal Informal Summary

- The plaintext m is “hidden” by multiplying it by β^k to get y_2
- The ciphertext includes α^k so that Bob can compute β^k from α^k (because Bob knows a)
- Thus, Bob can “reveal” m by dividing y_2 by β^k

I receive ct = (y_1,y_2)

Let’s see an example!

Bob's Pub_K → **(p, α, β)**

Bob's $Priv_K$ → **a** = 765

β≡α^a (mod p)

Example

- Let **p**=2579 , **α** = 2 , **β** = 2^{765} mod 2579 = 949

Bob's Pub_K → **(p, α, β)**
Bob's $Priv_K$ → **a** = 765
$\beta \equiv \alpha^a \pmod{p}$

Example

- Let **p**=2579 , **α** = 2 , **β** = 2^{765} mod 2579 = 949

I want to send **m**=1299 to Bob. I choose **k** = 853 for my random integer

Bob's $Pub_K \rightarrow$ **(p, α, β)**

Bob's $Priv_K \rightarrow$ **a** $= 765$

$\beta \equiv \alpha^a \pmod p$

Example

- Let $\mathbf{p}=2579$, $\boldsymbol{\alpha} = 2$, $\boldsymbol{\beta} = 2^{765} \bmod 2579 = 949$

I want to send **m**=1299 to Bob. I choose **k** = 853 for my random integer

$\mathbf{y_1} \equiv \alpha^k \pmod p$

$\mathbf{y_2} \equiv \beta^k m \pmod p$

Bob's Pub_K → **(p, α, β)**
Bob's $Priv_K$ → **a** = 765
$\boldsymbol{\beta} \equiv \alpha^a \pmod{p}$

Example

- Let **p**=2579 , **α** = 2 , **β** = 2^{765} mod 2579 = 949

I want to send **m**=1299 to Bob. I choose **k** = 853 for my random integer

$\mathbf{y_1} \equiv \alpha^k \pmod{p}$

$\mathbf{y_2} \equiv \beta^k\, m \pmod{p}$

- $\mathbf{y_1}$ = 2^{853} mod 2579 = 435
- $\mathbf{y_2}$=1299*949^{853} mod 2579 = 2396

Send y_1, y_2 to Bob

Bob's Pub_K → **(p, α, β)**

Bob's $Priv_K$ → **a** = 765

$\boldsymbol{\beta} \equiv \alpha^a \pmod{p}$

Example

- Bob now has $\mathbf{y_1}$ and $\mathbf{y_2}$
  - $y_1 = 2^{853} \bmod 2579 = 435$
  - $y_2 = 1299 * 949^{853} \bmod 2579 = 2396$

I received **y** = (435, 2396)

Bob's Pub_K → $\mathbf{(p, \alpha, \beta)}$

Bob's $Priv_K$ → $\mathbf{a} = 765$

$\boldsymbol{\beta} \equiv \alpha^a \pmod{p}$

Example

- Bob now has $\mathbf{y_1}$ and $\mathbf{y_2}$
  - $y_1 = 2^{853} \bmod 2579 = 435$
  - $y_2 = 1299 * 949^{853} \bmod 2579 = 2396$

I received $\mathbf{y}$ = (435, 2396)

$\mathbf{y_2 y_1^{-a}} \equiv \beta^k \, m \, (\alpha^k)^{-a} \equiv m \pmod{p}$

- $m = 2396 * 435^{-765} \bmod 2759 = 1299$

Bob's $Pub_K \rightarrow$ **(p, α, β)**

Bob's $Priv_K \rightarrow$ **a** = 765

$\boldsymbol{\beta} \equiv \alpha^a \pmod p$

Example

- Bob now has $\mathbf{y_1}$ and $\mathbf{y_2}$
  - $y_1 = 2^{853} \bmod 2579 = 435$
  - $y_2 = 1299 * 949^{853} \bmod 2579 = 2396$

I received **y** = (435, 2396)

$\mathbf{y_2 y_1^{-a}} \equiv \beta^k\, m\, (\alpha^k)^{-a} \equiv m \pmod p$

- $m = 2396 * 435^{-765} \bmod 2759 = 1299$

Nice! That's the plaintext I wanted to send.

Bob's Pub_K → $\mathbf{(p, \alpha, \beta)}$

Bob's $Priv_K$ → $\mathbf{a} = 765$

$\boldsymbol{\beta} \equiv \alpha^a \pmod{p}$

Example

- Bob now has $\mathbf{y_1}$ and $\mathbf{y_2}$
  - $y_1 = 2^{853} \bmod 2579 = 435$
  - $y_2 = 1299 * 949^{853} \bmod 2579 = 2396$

I received $\mathbf{y} = (435, 2396)$

$\mathbf{y_2 y_1^{-a}} \equiv \beta^k \, m \, (\alpha^k)^{-a} \equiv m \pmod{p}$

- $m = 2396 * 435^{-765} \bmod 2759 = 1299$

Insecure if the adversary can compute $\mathbf{a} = \log_\alpha \beta$

Nice! That's the plaintext I wanted to send.

Bob's Pub_K → **(p, α, β)**

Bob's $Priv_K$ → **a** = 765

$\beta \equiv \alpha^a \pmod p$

Example

- Bob now has $\mathbf{y_1}$ and $\mathbf{y_2}$
  - $y_1 = 2^{853} \bmod 2579 = 435$
  - $y_2 = 1299 * 949^{853} \bmod 2579 = 2396$

I received **y** = (435, 2396)

$\mathbf{y_2 y_1^{-a}} \equiv \beta^k\, m\, (\alpha^k)^{-a} \equiv m \pmod p$

- $m = 2396 * 435^{-765} \bmod 2759 = 1299$

Insecure if the adversary can compute $\mathbf{a} = \log_\alpha \beta$

To be secure, DLP must be infeasible in $\boldsymbol{Z}_p$*

Nice! That's the plaintext I wanted to send.

Network Security - Next class