

ElGamal Signature system ensures the validity, integrity, and non-repudiation of messages. In this system, a signer has a private key for message signing and a corresponding public key for signature verification.The following describes how the ElGamal Signature Scheme operates.

1. Key generation
2. Signature generation
3. Signature verification
4. Key Generation: The signer chooses a significant prime number p and a simple root modulo p. The signer then chooses a secret number at random, X, and calculates = x mod p wherepublic key is (p, α, β) and the private key is x.
5. Signature Generation: If'm' is the message, the signer chooses a random number k and calculates r = k mod p to sign it. The signer then calculates s = (hash(m) - xr) \* k-1 mod (p-1) using hash functions. The signature is (r, s) and the hash function is hash(m).
6. Signature verification: The verifier must determine the following values: w = s-1 mod (p-1), u1 = hash(m) \* w mod (p-1), and u2 = r \* w mod p in order to validate the created signature (r, s) with the message "m."

The verifier then calculates v = (u1 \* u2 mod p). We can argue that the signature is valid if v and r have the same value, or if v = r.

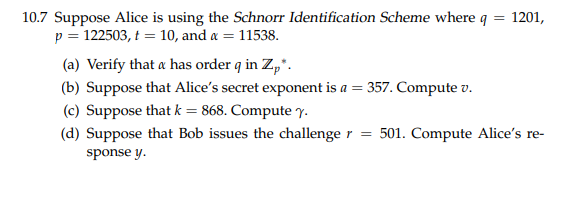
[2]

0s

# given values ( Exponents)  
p, alpha, beta, x, r, s = 31847, 5, 26379, 20543, 20679, 11082  
  
g = pow(alpha, x, p) # Calculate g = alpha^x mod p  
y1 = pow(beta, r, p)# Calculate y1 = beta^r mod p  
y2 = pow(r, s, p) # Calculate y2 = r^s mod p  
v = (y1 \* y2) % p   
  
print("valid signature" if v == g else "invalid signature")

valid signature

We can say that the signature is valid



Here,

p=122503 t= 10 α = 11538 q=1201

now,

a. Verify that α has order q in Zp∗.

Now, to verify that α has order q in Zp\*, we need to check the following conditions.

α^q mod p = 1 α^i mod p ≠ 1 for all i<q

Now finding α^q mod p = 11538^1201 mod 122503 = 1 Now, compute α ^ i mod p for i < q = 11538 ^ 1 mod 122503 ----------------i

= 11538

= 11538 ^ 2 mod 122503 -------------- ii

= 87186

= 11538 ^ 3 mod 122503 ------------------ iii

= 79935 In the same way,

= 11538 ^ 1200 mod 122503

= 85544

b suppose that Alice's secret component is 357 then ,

= α ^ a mod p

= 11538 ^ 357 mod 122503 = 114915

[3]

0s

# python code for b .  
p = 122503  
q = 1201  
t = 10  
α  = 11538  
  
  
a=357  
  
# Compute α ^ a mod p  
v = pow(α, a, p)  
  
print("Alice's public key V is ", v)

Alice's public key V is 114915

c suppose that k=868

=α ^ k mod p =11538 ^ 868 mod 122503 = 89937

[5]

0s

k = 868  
value= pow(α, k, p)  
  
print("Bob's public key is:", value)

Bob's public key is: 89937

d suppose that bob issues the challenge r=501, then Alice response would be ,

=(a +r*v) mod q = (357 + 501* 114915) mod 1201 =435

[6]

0s

r = 501  
# Compute Alice's response to the challenge  
y = (a + r \* v) % q  
  
print("Alice's response to the challenge:", y)

Alice's response to the challenge: 435

e Bob verifies y by ,

γ^y \* v^r mod p =α^(y+rv) mod p

for γ^y \* v^r mod p =89937 ^ 435 \* 114915 ^ 501 mod 122503 =96302

α^(y+rv) mod p = 11538 ^ (435 + 501\* 114915) mod 122503 = 35669

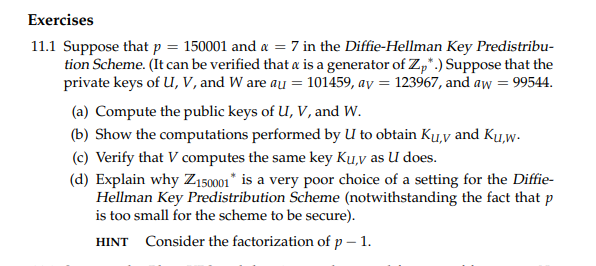
Since γ^y \* v^r mod p and α^(y+rv) mod p is not equal we cannot say Alice response is verified

[11]

0s

val1 = (pow(value, y, p) \* pow(v, r, p)) % p  
val2 = pow(α, y + r\*v, p)  
if val1 == val2:  
    print(" verified")  
else:  
    print(" not verified")

not verified



Here,

p=150001 α =7

private Keys: U =αu = 101459 V =αv = 123967 W =αw = 99544

also, The generator is α of Zp now,

a public keys are:

U = α'u = α^aU mod p = 7 ^ 101459 mod 150001 = 36138

V = α'v =α^aV mod p = 7 ^ 123967 mod 150001 = 10635

W =α'w = α^aW mod p

    = 7 ^ 99544 mod 150001  
    = 93325

b, now for the computations performed by U to get Ku,v and Ku,w

for K u,v

= (public V) ^ αu mod p = 106355 ^ 101459 mod 150001 = 75452

for K u,w

= (public W) ^ αu mod p = 93325 ^ 101459 mod 150001 = 119360

c, to verify if they share the same secret key,

here k u,v = 75452

now to check lets check the value of k v,u

= (public U) ^ αv mod p = 120714 ^ 123967 mod 150001 = 75452

Since, the secret key obtained is same, we can verify that V computes the same Key as U does

d,

Z1500018 is a poor choice of setting for the Diffie-Hallman Key Predistribution scheme because the factorization of p-1 is easy to figure out. Here ,

p-1 = 1500000 = 2 ,435 ,5

Also, this reuslts that there is not a large prime number but it has many small factors which makes it more vulnerable to ohlig-Hellman algorithm. Also, the value of p is too small to give security against attacks like brute force attacks.

Something which is a safe prime structure like p=2q+1 where the q is also prime should be used to ensure security

[ ]

x= 7

y= 101459

z= 150001

result = pow(x, y, z)

print (result)