program for Diffie-Hellman protocol, each participant selects a secret number x and sends the other participant ax mod q for some public number a. What would happen if the participants sent each other xa for some public number a instead? Give at least one method Alice and Bob could use to agree on a key. Can Eve break your system without finding the secret numbers? Can Eve find the secret numbers?

# Diffie-Hellman demo: wrong protocol vs correct protocol + Eve brute-force on small params

# Simple, ready-to-run Python 3 script

def wrong\_protocol(a, q, x, y):

"""Alice/Bob send x^a mod q and y^a mod q (WRONG).

Alice computes (y^a)^x, Bob computes (x^a)^y."""

sendA = pow(x, a, q) # Alice sends x^a mod q

sendB = pow(y, a, q) # Bob sends y^a mod q

alice\_key = pow(sendB, x, q) # (y^a)^x = y^(a x)

bob\_key = pow(sendA, y, q) # (x^a)^y = x^(a y)

return sendA, sendB, alice\_key, bob\_key

def correct\_dh(g, p, x, y):

"""Standard Diffie-Hellman: send g^x, g^y and compute shared key g^{xy}."""

A = pow(g, x, p)

B = pow(g, y, p)

alice\_key = pow(B, x, p)

bob\_key = pow(A, y, p)

return A, B, alice\_key, bob\_key

def brute\_force\_discrete\_log(g, p, value):

"""Brute force discrete log: find x such that g^x % p == value (small p only)."""

for x in range(0, p):

if pow(g, x, p) == value:

return x

return None

if \_\_name\_\_ == "\_\_main\_\_":

# Small example numbers so brute force is demonstrable

# (DO NOT use such small primes in real crypto)

p = 467 # prime modulus (small for demo)

g = 2 # generator (public)

a = 3 # a public value in the 'wrong' scheme (user-chosen)

# Alice and Bob secrets

x = 15

y = 22

print("=== WRONG SCHEME (send x^a mod q) ===")

sendA, sendB, alice\_key\_wrong, bob\_key\_wrong = wrong\_protocol(a, p, x, y)

print(f"Alice sends (x^a mod p) = {sendA}")

print(f"Bob sends (y^a mod p) = {sendB}")

print("Alice's derived key:", alice\_key\_wrong)

print("Bob's derived key:", bob\_key\_wrong)

if alice\_key\_wrong == bob\_key\_wrong:

print("Keys match (unexpected).")

else:

print("Keys DO NOT match. Protocol fails for general secrets.\n")

print("=== CORRECT DIFFIE-HELLMAN (send g^x) ===")

A, B, alice\_key, bob\_key = correct\_dh(g, p, x, y)

print(f"Alice sends A = g^x mod p = {A}")

print(f"Bob sends B = g^y mod p = {B}")

print("Alice's derived key:", alice\_key)

print("Bob's derived key:", bob\_key)

if alice\_key == bob\_key:

print("Keys match (shared key = g^(x\*y) mod p).")

else:

print("Unexpected: keys differ.\n")

print("\n=== EVE (brute-force) ON SMALL p ===")

# Eve intercepts A and B and tries to find x and y by brute force (feasible only for small p)

recovered\_x = brute\_force\_discrete\_log(g, p, A)

recovered\_y = brute\_force\_discrete\_log(g, p, B)

print("Eve recovered x =", recovered\_x, "y =", recovered\_y)

if recovered\_x is not None and recovered\_y is not None:

# Eve can compute shared key:

eve\_key = pow(B, recovered\_x, p)

print("Eve computed shared key:", eve\_key)

else:

print("Eve failed to recover secrets by brute force (p too large for demo).")

print("\n--- Summary ---")

print("1) Sending x^a instead of g^x generally does NOT produce a common key.")

print("2) Correct DH: send g^x and g^y; both can compute g^(x\*y).")

print("3) Eve can break small-parameter DH by brute force (discrete log).")

print(" With large safe primes and proper g, discrete log is infeasible, so Eve cannot compute the shared key without solving hard math.") 