program for RSA public-key encryption scheme, each user has a public key, e, and a private key, d. Suppose Bob leaks his private key. Rather than generating a new modulus, he decides to generate a new public and a new private key. Is this safe?

# Simple RSA demo: showing why reusing n after private key leak is unsafe

import math

# Original RSA parameters

p = 61

q = 53

n = p \* q

phi = (p - 1) \* (q - 1)

# Original keys

e1 = 17

# Compute d1 (private key)

def mod\_inverse(a, m):

for x in range(1, m):

if (a \* x) % m == 1:

return x

return None

d1 = mod\_inverse(e1, phi)

print("Original keys:")

print(f"n = {n}, e1 = {e1}, d1 = {d1}")

# Bob leaks d1, but tries to generate a new key pair using same n

e2 = 7

d2 = mod\_inverse(e2, phi)

print("\nBob's 'new' keys (same n):")

print(f"e2 = {e2}, d2 = {d2}")

# Attacker already knows d1 and n, so can recompute phi and any future private key

# phi = (e1\*d1 - 1) / k for some integer k

k = 1

possible\_phi = (e1 \* d1 - 1) // k

# Attacker recomputes new d easily

attacker\_d2 = mod\_inverse(e2, possible\_phi)

print("\nAttacker recomputed new d2 easily:", attacker\_d2)

# Demonstrate encryption/decryption still works with compromised info

message = 42

cipher = pow(message, e2, n)

decrypted = pow(cipher, attacker\_d2, n)

print("\nMessage:", message)

print("Ciphertext:", cipher)

print("Decrypted (by attacker):", decrypted)

