* RSA Factorization Attack

Description:

**RSA (Rivest-Shamir-Adleman)** is a widely used public-key cryptosystem that allows secure data transmission. It works by generating two keys: a **public key** for encryption and a **private key** for decryption. RSA's security is based on the difficulty of factoring large prime numbers.

How RSA works:

**Step 1: Key Generation**

1. **Choose Two Large Prime Numbers**:  
   Select two large prime numbers, denoted as p and q. These primes should be chosen randomly and kept secret.
   * Example: Let p = 61 and q =53
2. **Compute n** (Modulus):  
   Multiply the two primes to get n, which will be part of both the public and private keys.

n=p × q

* + Example: n=61×53=3233

1. **Compute ϕ(n)** (Euler’s Totient Function):  
   Calculate ϕ(n), which represents the number of integers less than n that are relatively prime to n. For RSA, ϕ(n) is calculated as:

ϕ(n)=(p−1) × (q−1)

* + Example: ϕ(n)=(61−1) × (53−1) = 60×52=3120

1. **Choose Public Exponent e**:  
   Select an integer e such that 1 <e <ϕ(n) and gcd(e,ϕ(n)) =1 (i.e., e and ϕ(n) are coprime). The value of e is usually chosen as 65537, a common choice because it ensures efficient encryption.
   * Example: Let e=17
2. **Compute Private Key d**:  
   Calculate the private key d as the modular multiplicative inverse of e modulo ϕ(n) .This can be done using the Extended Euclidean Algorithm. d satisfies:

d×e≡1 (mod ϕ(n))

Example: d=2753 (since 17×2753≡1 (mod 3120)

**Keys:**

* **Public Key**: (n,e)
* **Private Key**: (n,d)

**Step 2: Encryption**

1. **Convert the Message to a Number**:  
   Represent the plaintext message M as an integer m, where 0≤m<n This is usually done by using some form of encoding (e.g., ASCII, UTF-8).
   * Example: Let m=65 (if the message is represented by ASCII value of 'A').
2. **Encrypt the Message**:  
   The ciphertext c is generated using the public key (n,e) by raising m to the power of e and then taking the modulus n:

c=m^e (mod n)

* + Example: c=65^17 (mod 3233) =2790

**Step 3: Decryption**

1. **Decrypt the Ciphertext**:  
   To decrypt, use the private key (n, d) to raise the ciphertext c to the power of d and take the modulus n:

m=c^d (mod n)

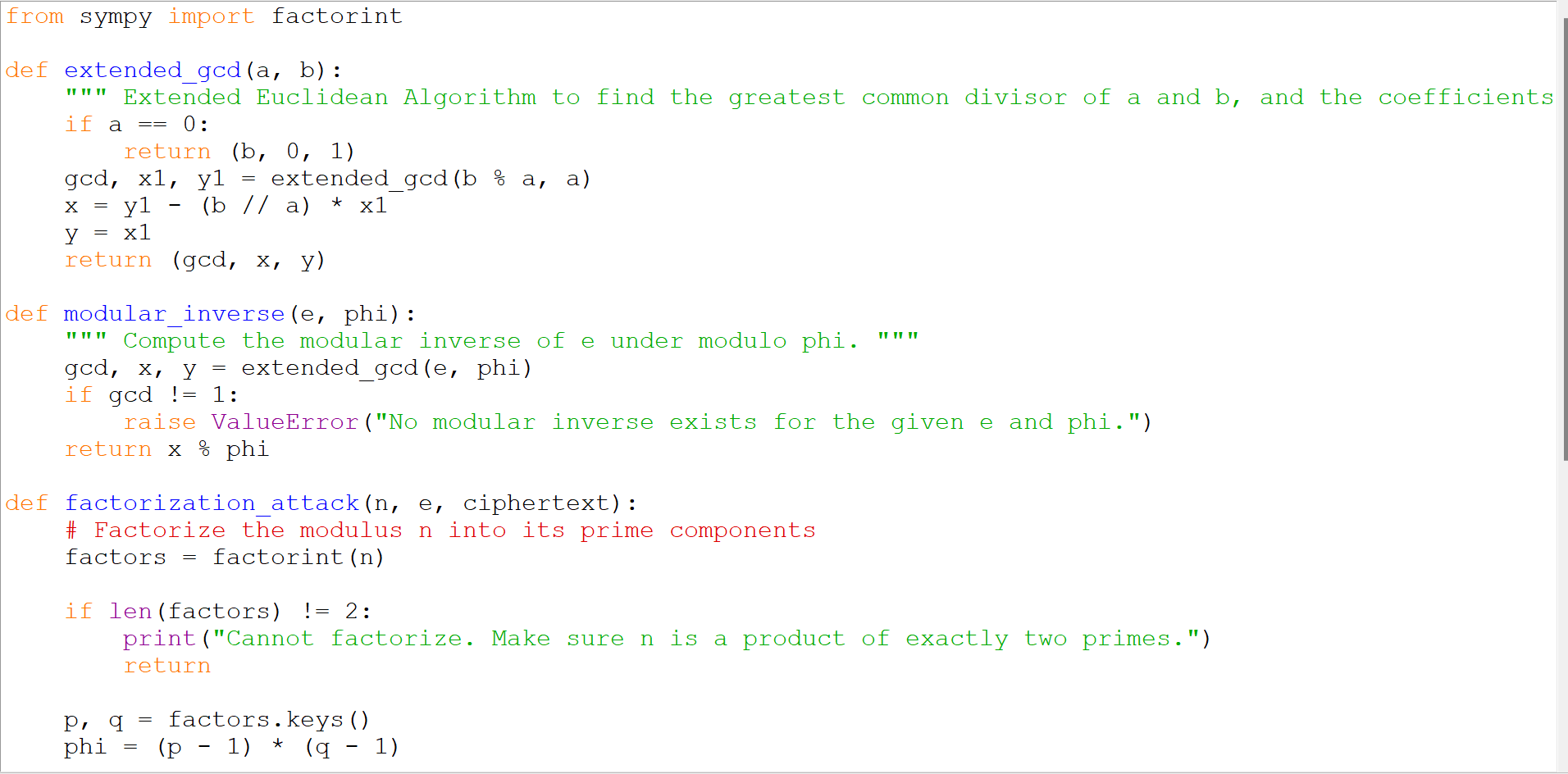
* + Example: m=2790^2753 (mod 3233)=65

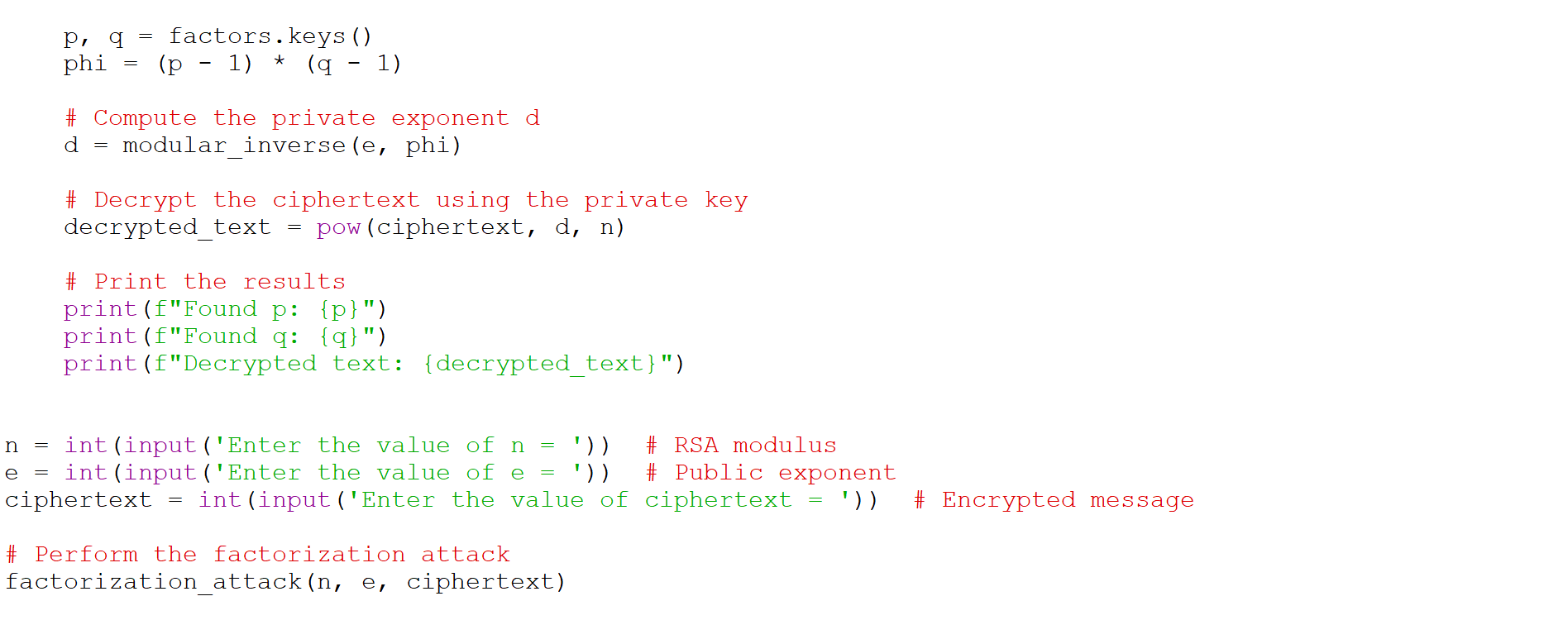
1. **Convert the Number Back to Text**:  
   The integer m is then converted back to the original plaintext message using the reverse of the encoding scheme.
   * Example: m=65 corresponds to the letter 'A' in ASCII.

Attacks on RSA and Countermeasures:

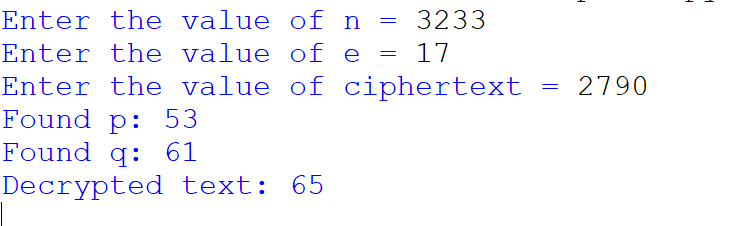
1. **Factorization Attack**:
   * **Attack**: If an attacker can factor n into p and q, they can compute ϕ(n) and retrieve the private key d.
   * **Solution**: Use larger key sizes (2048-bit or more), making factorization computationally infeasible.
2. **Timing Attack**:
   * **Attack**: Based on measuring the time taken for decryption, attackers can infer the private key.
   * **Solution**: Implement constant-time operations for decryption.
3. **Chosen Ciphertext Attack (CCA)**:
   * **Attack**: Involves sending manipulated ciphertexts to the decryption oracle and analyzing the responses to deduce the private key.
   * **Solution**: Use padding schemes like **Optimal Asymmetric Encryption Padding (OAEP)**, which introduces randomness to the encryption process.

Source code:





Output:



**Factorization Attack:**

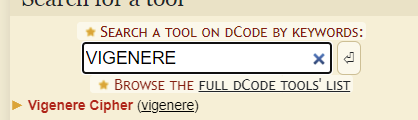
In a factorization attack, the security of RSA can be broken if an attacker successfully factors the modulus n into its two prime factors, p and q. Once these primes are found, the attacker can easily calculate ϕ(n) and retrieve the private key using the Extended Euclidean Algorithm.

* **How to Prevent It**:
  1. **Large Key Sizes**: Use key sizes of at least 2048 bits or 4096 bits, which makes factorization extremely time-consuming with current computational power.
  2. **Random Prime Generation**: Ensure that the primes p and q are chosen randomly and are sufficiently large.
  3. **Use Strong Primes**: Use primes that are specially chosen to be resistant to certain factorization techniques.
* Cracking the Ciphers without Knowing the Key value:

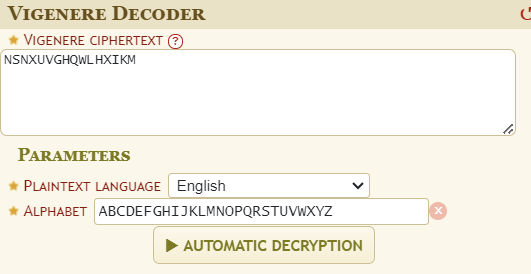
In the below website we can crack any of the simple ciphers like Vigenere ,Caesar, playfair ciphers and etc without knowing the key/ just by guessing the key size.

<https://www.dcode.fr/en>

Select The cipher for which we need to crack the encrypted text.

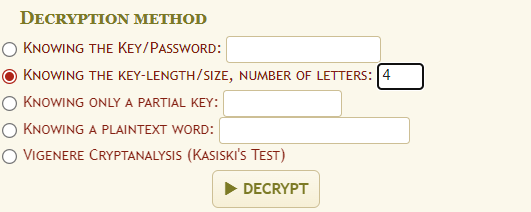


Write the Encrypted text in the box so that we can crack the Encrypted text.

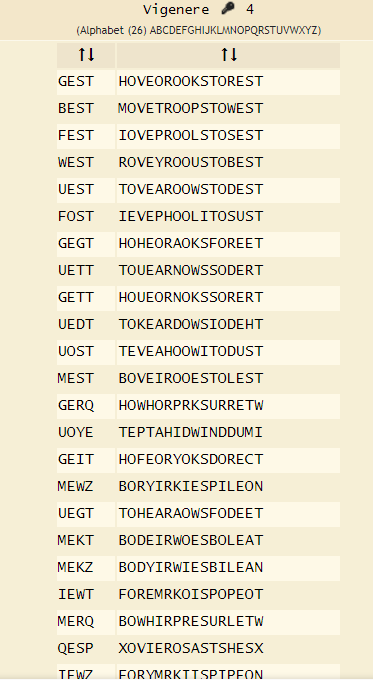


Select The Decryption Method from the following

Over here the Knowing key length size is choosen (the tester can make a guess of how many letters the key would be through this without knowing the key we can get to know the number of possible outcomes)



Over here the possible outcomes for the key size 4 is given below.



In the above website basically Brute force attack is performed.

**Brute Force Attacks on Ciphers**

Brute force attacks are a type of cryptanalytic attack where the attacker tries all possible keys or combinations until the correct one is found. It is considered the simplest form of attack, but its feasibility depends on the size of the keyspace.

The **Vigenère cipher** uses a repeating key to shift letters based on the positions defined by the key. It is more complex than a Caesar cipher because it uses multiple shifts based on the keyword.

**Working of Brute Force Attack:**

* **Keyspace**: The main challenge in brute-forcing the Vigenère cipher is the size of the keyspace. If the key length is known, the attack involves trying every possible combination of shifts corresponding to each letter of the key.
  + If the key length l is small, brute-forcing becomes feasible, as there are 26^l possible keys (where 26 represents the number of letters in the alphabet).
* **When Key Length is Unknown**: If the key length is not known, the attacker would first try to deduce the key length, possibly using methods like the **Kasiski examination** and then attempt brute-forcing each letter of the key.