Project 3: Crypto – Have fun with RSA

Introduction to Information Security

Overview

- Five RSA tasks ordered by easy to difficult
- Python3
- Different student has different key, message, etc.
- Deadline: March 30 11:59 pm
- Further reading: Twenty Years of Attacks on the RSA Cryptosystem

Task1 – Get Familiar with RSA

Given the public key (e, N) and the private key (d) of RSA, decrypt the encrypted message c.

```
"1bf74f7f82241a64d6b44c76a367c663a2dc8bebf43b25f0c77a7ea7": {
  "c": "0x944...49",
  "e": "0x10001",
  "d": "0x567...81",
  "N": "0x9d7...af"
},
```

Task1 – Get Familiar with RSA (5 points)

- Encrypt m with (N , e) : $c = m^e \bmod N$
- Decrypt c with private key d: $m = c^d \bmod N$

```python
def decrypt_message(self, N, e, d, c):
    m = 0

    return m
```

Task 2 – Can I Have Some Salt With My Password? (10 points)

- Using salts before hashing to prevent a rainbow table attack
- Offers more protection for a weak password
- Given SHA256 hash of a common password, that includes a salt value
- Decode original password and salt value from hash
  - Salt value is from same list of common password (provided)

```
def crack_password_hash(self, password_hash, weak_password_list):
    password = 'abc'
    salt = '123'
    hashed_password = hashlib.sha256(password.encode() +
        salt.encode()).hexdigest()

    return password, salt
```

Task 3 – Attack Small Key Space (20 points)

- Given an N, you can factorize it into (q, p) and compute the private key d
- $d \equiv e^{-1} \bmod \phi(N)$, where $\phi(N)=(p-1)*(q-1)$
- Extended Euclidean Algorithm

```
def get_factors(self, n):
    p = 0
    q = 0

    return p, q
```

```
def get_private_key_from_p_q_e(self, p, q, e):
    d = 0

    return d
```

Task4 – Where Is Waldo? (30 points)

- Reference paper: Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices
  - Most relevant section: 2.1 RSA review
- Public key of yours and one of your classmate (Waldo) share a common factor
- Once Waldo is found, get your unique private key

```
def is_waldo(self, n1, n2):
    result = False

    return result
```

```
def get_private_key_from_n1_n2_e(self, n1, n2, e):
    d = 0

    return d
```

Task 5 – Broadcasting RSA Attack (35 points)

- The same message was encrypted by three different public keys and with the same small public exponent e = 3.
- Results in 3 different encrypted messages
- You can figure out the original message without knowing any of the private keys
- RSA broadcast attack
- Idea:

  c1 = m^3 mod n1

  c2 = m^3 mod n2

  c3 = m^3 mod n3

  You can get:

  m^3 mod n1*n2*n3

  by chinese remainder theorem

Task 5 – Broadcasting RSA Attack

- m = 42 is just a placeholder, replace it with the m you find

```
def recover_msg(self, N1, N2, N3, C1, C2, C3):
    m = 42
    # Note that 'm' should be an integer

    return m
```

Extra Details

- Do not alter import list
- Change student ID in crypto_proj .py
- Test with test_crypto_proj.py for *bdornier3*
- Runtime< 10 min
- Submit:
    - crypto_project.py
    - project3_report.pdf (reflection)