Wiener's
Low Decryption Exponent Attack

- L Srinivas Naik
CS13B1021

Classical Cryptography problem:

- Fundamental objective is to enable two people to communicate over an insecure channel in such a way that some other person couldn’t understand it.

- General approaches:
  - Single/Private Key Cryptography
  - Public key Cryptography

Why Public Key Cryptography:

- Asymmetric unlike Private/Single key Cryptography
  - those who encrypt messages cannot decrypt messages

- No need of a secure channel to distribute keys.
  - Generates a public-key, which may be known by anybody, and can be used to encrypt messages.

- Possible to maintain Digital signatures
  - verify a message comes intact from the claimed sender

RSA Cryptosystem

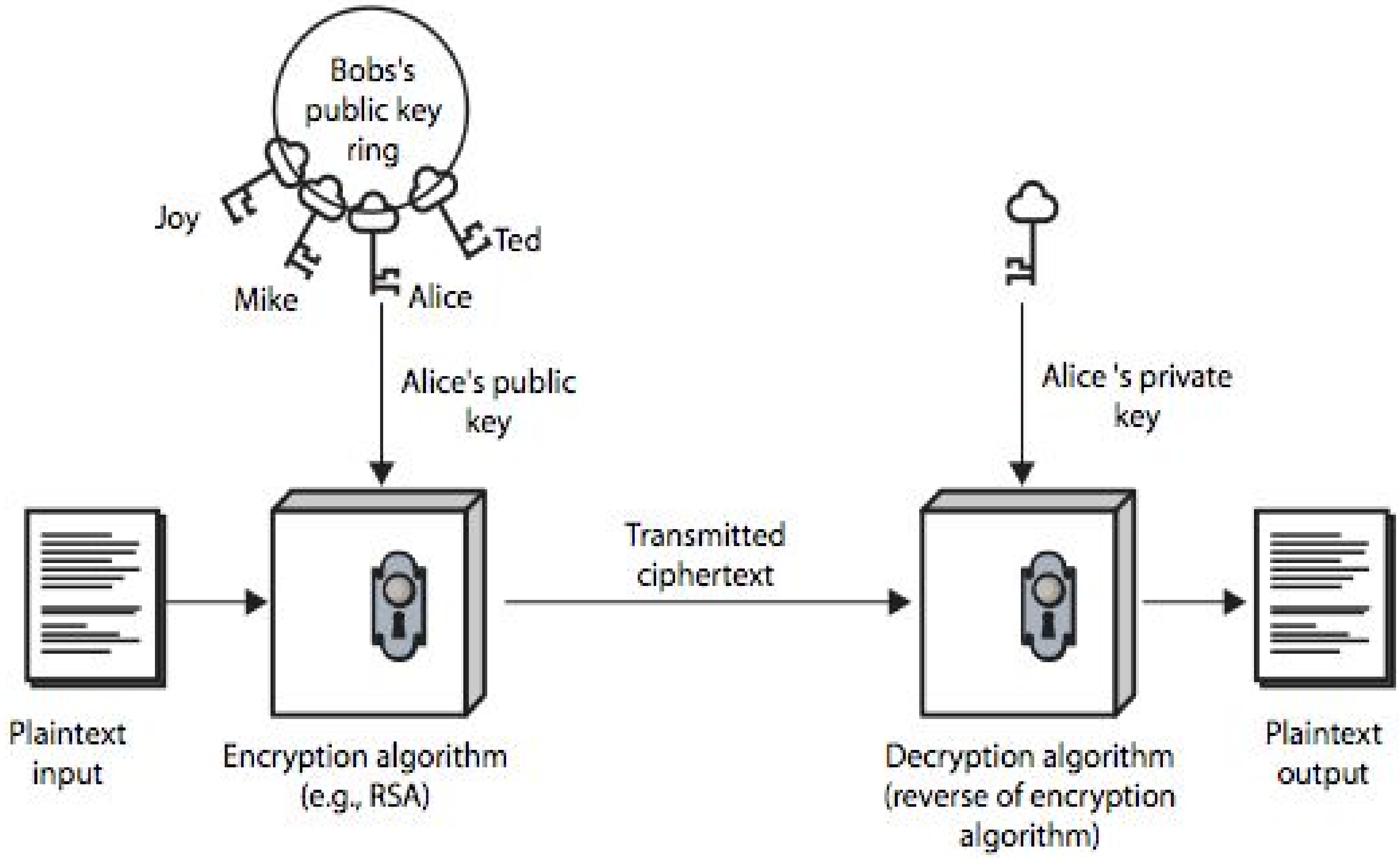
- RSA (Rivest, Shamir & Adleman) is best known and widely used public encryption scheme.

- It is currently “Work Horse” of internet security.

- It is highly secure due to usage of large numbers which have high cost of factorization.

- RSA is a trapdoor one to one function.

RSA Cryptosystem



(a) Encryption

RSA Parameter Generation:

- Select two random large prime numbers p, q. Compute **n = p*q**

- Compute Euler's Totient function for n, which comes out to be : $\boldsymbol{\varphi(n) = (p-1)*(q-1)}$

- Choose a random encryption key 'b' ($1 < b < \varphi(n)$) such that **gcd(b,φ(n))=1**

- Now compute decryption key 'a', such that **a*b = 1modn**

- Now the public key is (n,b) and the private key is (p,q,a).

RSA Algorithm

- To encrypt a message ‘x’ the sender:
  - obtains public key of recipient (n,b)
  - computes: $e(x) = x^b \bmod n$
  - e(x) is encrypted message.

- To decrypt the ciphertext ‘y’ the owner:
  - uses their private key (p,q,a)
  - computes: $d(y) = y^a \bmod n$
  - d(y) is decrypted message.

Wiener's attack

During the parameter generation, we need to randomly get an encryption exponent 'b', such that ($1 < b < \varphi(n)$) and $\gcd(b,\varphi(n)) = 1$. This takes up large time due to large range of 'b'.

So instead if we choose a small decryption exponent 'a' satisfying $3a<n^{1/4}$

- ➔ we can easily calculate 'b' using Extended euclidean algorithm.
- ➔ running time will be reduced to almost 75%.

Wiener’s attack

- We need to choose large prime numbers randomly.
- If we choose a random large prime number ‘q’ and look for another prime number ‘p’ such that $q<p<2q$ then we can get a good optimisation of running time.
- If $3a<n^{1/4}$ and $q<p<2q$ then the system is vulnerable to Wiener’s attack and we can easily factorize n into $p*q$.
- Wiener’s attack works based on the convergent theorem of Continued fractions.

Continued Fractions:

A continued fraction is an expression of the form

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots + \cfrac{1}{a_n}}}} \qquad = \qquad [a0,a1,a2,.....,an]$$

Theorem:
Suppose that gcd(a,b) - gcd(c,d) = 1 and $\left|\frac{a}{b} - \frac{c}{d}\right| < \frac{1}{2d^2}$.

then c/d is one of the convergents of the continued fractions expansion of a/b.

Since $ab \equiv 1 \pmod{\phi(n)}$, it follows that there is an integer t such that

$$ab - t\phi(n) = 1.$$

Since $n = pq > q^2$, we have that $q < \sqrt{n}$. Hence,

$$0 < n - \phi(n) = p + q - 1 < 2q + q - 1 < 3q < 3\sqrt{n}.$$

Now, we see that

$$\begin{aligned}
\left|\frac{b}{n} - \frac{t}{a}\right| &= \left|\frac{ba - tn}{an}\right| \\
&= \left|\frac{1 + t(\phi(n) - n)}{an}\right| \\
&< \frac{3t\sqrt{n}}{an} \\
&= \frac{3t}{a\sqrt{n}}.
\end{aligned}$$

Since $t < a$, we have that $3t < 3a < n^{1/4}$, and hence

$$\left|\frac{b}{n} - \frac{t}{a}\right| < \frac{1}{an^{1/4}}.$$

Finally, since $3a < n^{1/4}$, we have that

$$\left|\frac{b}{n} - \frac{t}{a}\right| < \frac{1}{3a^2}.$$

Therefore the fraction t/a is a very close approximation to the fraction b/n.

Wiener's attack

t/a is one of the continued fraction convergent of b/n.

If we know t/a we can calculate $\varphi(n) = (ab-1)/t$

Once n and $\varphi(n)$ are known we can easily calculate the prime numbers p and q solving the quadriatic equation:

$$x^2-(n-\varphi(n)+1)x+n = 0$$

We can use Euclidean algorithm to calculate convergents of continued fractions.

Wiener's Algorithm:

$(q_1, \ldots, q_m; r_m) \leftarrow \textsc{Euclidean algorithm}(b, n)$

$c_0 \leftarrow 1$

$c_1 \leftarrow q_1$

$d_0 \leftarrow 0$

$d_1 \leftarrow 1$

for $j \leftarrow 2$ **to** m

do
- $c_j \leftarrow q_j c_{j-1} + c_{j-2}$
- $d_j \leftarrow q_j d_{j-1} + d_{j-2}$
- $n' \leftarrow (d_j b - 1)/c_j$
- **comment:** $n' = \phi(n)$ if c_j/d_j is the correct convergent
- **if** n' is an integer
  - **then**
    - let p and q be the roots of the equation
      $x^2 - (n - n' + 1)x + n = 0$
    - **if** p and q are positive integers less than n
      then return (p, q)

return ("failure")

Euclid's Algorithm:

Algorithm 5.1: EUCLIDEAN ALGORITHM(a, b)

$r_0 \leftarrow a$
$r_1 \leftarrow b$
$m \leftarrow 1$
while $r_m \neq 0$

$$\textbf{do} \begin{cases} q_m \leftarrow \left\lfloor \frac{r_{m-1}}{r_m} \right\rfloor \\ r_{m+1} \leftarrow r_{m-1} - q_m r_m \\ m \leftarrow m + 1 \end{cases}$$

$m \leftarrow m - 1$
return $(q_1, \ldots, q_m; r_m)$
comment: $r_m = \gcd(a, b)$

Extended Euclid's algorithm

Algorithm 5.2: EXTENDED EUCLIDEAN ALGORITHM(a, b)

$$
\begin{array}{l}
a_0 \leftarrow a \\
b_0 \leftarrow b \\
t_0 \leftarrow 0 \\
t \leftarrow 1 \\
s_0 \leftarrow 1 \\
s \leftarrow 0 \\
q \leftarrow \left\lfloor \frac{a_0}{b_0} \right\rfloor \\
r \leftarrow a_0 - qb_0 \\
\textbf{while } r > 0 \\
\quad \textbf{do} \begin{cases}
temp \leftarrow t_0 - qt \\
t_0 \leftarrow t \\
t \leftarrow temp \\
temp \leftarrow s_0 - qs \\
s_0 \leftarrow s \\
s \leftarrow temp \\
a_0 \leftarrow b_0 \\
b_0 \leftarrow r \\
q \leftarrow \left\lfloor \frac{a_0}{b_0} \right\rfloor \\
r \leftarrow a_0 - qb_0
\end{cases} \\
r \leftarrow b_0 \\
\textbf{return } (r, s, t) \\
\textbf{comment: } r = \gcd(a, b) \text{ and } sa + tb = r
\end{array}
$$

Result

Two programs written

wiener_rsa.py - for generating parameters vulnerable to Wiener’s attack

wiener.py - for performing Wiener’s attack

```
sinu@sinu-SVF15212SNB: ~/Documents/sem7/mini-project
sinu@sinu-SVF15212SNB:~$ cd Documents/sem7/mini-project/
sinu@sinu-SVF15212SNB:~/Documents/sem7/mini-project$ python rsa_weiner.py
Enter prime number1:
37124508045065437
Enter prime number2:
25730318403586483
Public Key:
b:  65575331451138650209549747942969 7   n:  955225412576041659832131223688071
sinu@sinu-SVF15212SNB:~/Documents/sem7/mini-project$ python weiner.py
python: can't open file 'weiner.py': [Errno 2] No such file or directory
sinu@sinu-SVF15212SNB:~/Documents/sem7/mini-project$ python wiener.py
955225412576041659832131223688071
655753314511386502095497479429697
p:  37124508045065437  q:  25730318403586483
sinu@sinu-SVF15212SNB:~/Documents/sem7/mini-project$
```

References

- en.wikipedia.org/wiki/RSA_(cryptosystem)
- en.wikipedia.org/wiki/Wiener's_attack
- Cryptography - Theory and Practice by Douglas R. Stinson
- en.wikipedia.org/wiki/Continued_fraction
- www.math.jacobs-university.de/timorin/PM/continued_fractions.pdf
- Prime database -primes.utm.edu

THANK YOU