Chapter 7

Lecture 7 - Encryption

This session

- Public key systems
  - Sharing keys - Diffie-Hellman
  - Asymmetric encryption
  - RSA
Public key systems

✓ In 1976 Diffie and Hellman published the paper “New Directions in Cryptography”, which first introduced the idea of public key cryptography.

✓ Public key cryptography relies on the use of enciphering functions which are not realistically invertible unless you have a deciphering key.

    Easy to do one way - hard to do the other way.

CS3235 - Hugh Anderson's notes. Page number: 354

Not realistically invertible...

✓ The discrete logarithm problem:

    ✓ easy to calculate $n = g^k \text{ mod } p$ given $g$, $k$ and $p$,
    ✓ hard to calculate $k$ in the same equation, given $g$, $n$ and $p$.

CS3235 - Hugh Anderson's notes. Page number: 355
Diffie-Hellman key agreement

Two separated users create and share a secret key. A third party is not realistically able to calculate the shared key.

\[ p, g, a \quad g^a \mod p \rightarrow g^b \mod p \]
\[ p, g, b \quad g^b \mod p \rightarrow g^a \mod p \]

Ted

Knowledge different

- All participants know two system parameters \( p \), and \( g \)
- Alice and Bob each have a secret value (Alice has \( a \) and Bob has \( b \))
- Alice and Bob each calculate and exchange a public key (\( g^a \mod p \) for Alice and \( g^b \mod p \) for Bob).
- Ted knows \( g, p, g^a \mod p \) and \( g^b \mod p \), but not \( a \) or \( b \).
Diffie-Hellman key agreement

Both Alice and Bob can now calculate the value $g^{ab} \mod p$.

1. Alice calculates $(g^b \mod p)^a \mod p = (g^b)^a \mod p$.

2. Bob calculates $(g^a \mod p)^b \mod p = (g^a)^b \mod p$.

And of course $(g^b)^a \mod p = (g^a)^b \mod p = g^{ab} \mod p$ which is the shared key.

Ted has a much more difficult problem. It is difficult to calculate $g^{ab} \mod p$ without knowing either $a$ or $b$. The algorithmic run-time of the (so-far best) algorithm for doing this is in

$$O(e^{c\sqrt{\log r}})$$

where $c$ is small, but $\geq 1$, and $r$ is the number of bits in the number.
Diffie-Hellman key agreement

By contrast, the enciphering and deciphering process may be done in $O(r)$:

<table>
<thead>
<tr>
<th>Bit size</th>
<th>Enciphering</th>
<th>Discrete logarithm solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>23</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>1,386,282</td>
</tr>
<tr>
<td>1,000</td>
<td>1,000</td>
<td>612,700,000,000,000,000,000</td>
</tr>
</tbody>
</table>

Use of Diffie-Hellman key agreement

✔ To share a DES key...
This session

- Public key systems
  - Sharing keys - Diffie-Hellman
    - Asymmetric encryption
    - RSA

Uses of asymmetric encryption

1. Generating encrypted passwords with 1-way functions
2. Checking integrity by appending digital signature
3. Checking the authenticity of a message.
4. Encrypting timestamps with messages to prevent replay attacks.
5. Exchanging a key.
Asymmetric encryption

✔ Participants each have private and public keys
✔ Keys cannot be derived from each other

(Plaintext) $P$

$K_1[P]$ (K1[K2[P]] = P)

and also

(K1[K2[P]] = P)

✔ $K_1$ is private key for left participant, $K_2$ is her public key.
Authentication

✔ K1 is private key for left participant, K2 is her public key.

✔ J1 is private key for right participant, J2 is his public key.

This session

- Public key systems
  - Sharing keys - Diffie-Hellman
    * Asymmetric encryption
    * RSA
RSA (Rivest, Shamir, Adelman)

This public key system relies on the difficult problem of trying to find the complete factorization of a large composite\textsuperscript{15} integer whose prime factors\textsuperscript{16} are not known.

\textsuperscript{15}An integer larger than 1 is called composite if it has at least one divisor larger than 1.

\textsuperscript{16}The Fundamental Theorem of Arithmetic states that any integer \(N\) (greater than 0) may be expressed uniquely as the product of prime numbers.

RSA hacks

Two RSA-encrypted messages have been cracked:

- The inventors of RSA published a 129-digits (430 bits) RSA public key. In 1994, it was factored with 5000 MIPS-years of computing time.

- A year later, a 384-bit PGP key was cracked. It needed 1300 MIPS-years to factor the key in three months.

Note that these efforts each only cracked a single RSA key.
RSA hacks

If you happen to be able to factor the following number, please tell Hugh - we can split US$200,000\textsuperscript{17}!

```
251959084756578934940271832400483985714292821262040032027777113783604366202
0707595556264018525880784406918290641249515082189298559149176184502808489
120072844992687392807287776735971418347270261896350149718246911650776133
7985909570009733045974880842840179742910064245869181719511874612151517265
4632282216869987549182422433637259085141865462043576798423387184774447920
7399342365848238242811981638150106748104516603773060562016196762561338441
43603833904414952634432190111465754445417842402092461651572335077870774981
7125772467962926386356373289912154831438167898985040445364023527381951378
636564391212010397122822120720357
```

\textsuperscript{17}US$150,000 for me, US$50,000 for you...

---

RSA coding algorithms

Below are outlined the four processes needed for RSA encryption:

1. Creating a **public** key
2. Creating a **secret** key
3. **Encrypting** messages
4. **Decoding** messages
To create public key $K_p$

1. Select two different large primes $P$ and $Q$.

2. Assign $x = (P - 1)(Q - 1)$. (Does this ring a bell?)

3. Choose $E$ relative prime to $x$. (This must satisfy condition for $K_s$ given later)

4. Assign $N = P * Q$.

5. $K_p$ is $N$ concatenated with $E$.

To create private (secret) key $K_s$

1. Choose $D$: $D * E \mod x = 1$.
   
   (a) (i.e. multiplicative inverses)
   (b) another way: $DE = k(P - 1)(Q - 1) + 1$

2. $K_s$ is $N$ concatenated with $D$. 
To encode plain text \( m \)

1. Pretend \( m \) is a number.

2. Calculate \( c = m^E \mod N \).

To decode \( c \) back to \( m \)

1. Calculate \( m = c^D \mod N \).

2. ....WHY?....
...Why?...

c^D \mod N = m^{ED} \mod N
= m^{k(P-1)(Q-1)+1} \mod PQ
= m * m^{k(P-1)(Q-1)} \mod PQ
= m

- \text{so} \quad m^{P-1} \mod P = 1, \text{ SO } (m^{(P-1)})^{k(Q-1)} \mod P = 1

- \text{and so (tutorial)} \quad (m^{(P-1)})^{k(Q-1)} \mod PQ = 1.

RSA code

```perl
#!/usr/bin/perl -sp
s/\W//g;

and then

- echo "squeamish ossifrage" | ./rsa.perl -k=10001 -n=1967cb529 > msg.rsa

- ./rsa.perl -d -k=ac363601 -n=1967cb529 < msg.rsa
```