The University of Sydney
Math3024 Elementary Cryptography and Protocols

Semester 1 **Exercises for Week 11** 2004

# RSA Cryptosystems

The RSA cryptosystem is based on the difficulty of factoring large integers into its composite primes.

Based on Fermat's little theorem, we know that $a^m \equiv 1 \bmod p$ exactly when $p-1$ divides $m$. Therefore we recover the identity $a^u \equiv a \bmod p$ where $u$ is of the form $1+(p-1)r$. Now given any $e$ such that $e$ and $p-1$ have no common divisors, there exists a $d$ such that $ed \equiv 1 \text{ mod } p-1$. In other words, $u = ed$ is of the form $1+(p-1)r$. This means that the map

$$a \mapsto a^e \text{ mod } p$$

followed by

$$a^e \text{ mod } p \mapsto (a^e \text{ mod } p)^d \text{ mod } p \equiv a^{ed} \text{ mod } p = a \text{ mod } p$$

are inverse maps. This only works for a prime $p$.

1. Use Magma to find a large prime $p$ and to compute inverse exponentiation pairs $e$ and $d$. The following functions are of use:

`RandomPrime`, `Random`, `GCD`, `XGCD`, and `InverseMod`.

The RSA cryptosystem is based on the fact that for primes $p$ and $q$ and any integer $e$ with no common factors with $p-1$ and $q-1$, it is possible to find an $d_1$ such that

$$\begin{aligned} ed_1 &\equiv 1 \text{ mod } (p-1), \\ ed_2 &\equiv 1 \text{ mod } (q-1). \end{aligned}$$

Using the Chinese remainder theorem, it is possible to then find the unique $d$ such that

$$d = d_1 \text{ mod } (p-1) \text{ and } d = d_2 \text{ mod } (q-1)$$

in the range $1 \leq d < (p-1)(q-1)$. This $d$ has the property that

$$a^{ed} \equiv a \text{ mod } n.$$

The send a message securely, the public key $(e, n)$ is used. First we encoding the message as an integer $a \text{ mod } n$, then form the ciphertext $a^e \text{ mod } n$. The recipient recovers the message using the secret exponent $d$.

2. Use your exponents $e$, $d$, verify the identities mod $p$:

$$(a^e)^d \equiv a \text{ mod } n,\ (a^d)^e \equiv a \text{ mod } n,\ \text{and } a^{ed} \equiv a \text{ mod } n,$$

for various random values of $a$.

Note that after construction of $d$, the primes $p$ and $q$ are not needed, but that without knowing the original factorization of $n$, Fermat's little theorem does not apply, and finding the inverse exponent for $e$ is considered a hard problem.

We use the RSA cryptosystem in Magma as follows. First begin with encoding ASCII text numerically:

```
> C := RSACryptosystem(128);
> Encoding(C,"The dog ate my lunch.");
0101010001101000011001010010000001100100011011110110011100100\
0000110000101110100011001010010000001101101011110010010000001\
1011000111010101101110011000110110100000101110
> Decoding($1);
The dog ate my lunch.
```

Note that the encoding / decoding operations here are true inverses.

**Caution:** Decoding ciphertext might render an xterm nonfunctional, since the resulting ASCII text might contain escape characters which reset the terminal display.

To encipher, first we must create a key pair:

```
> K, L := RandomKeys(C);
> K;
[ 49338921862830381807760291818994034053,
  86398677368792768067556452456311743331 ]
```

This returns a pair of inverse keys `K` and `L`. We will consider `K` to be the public `K` and `L` to be the private key.

**N.B.** The argument to `RSACryptosystem` specifies the number of bits in the RSA modulus. With a value of 128, the modulus is of size $2^{128}$, or about 39 decimal digits. Each of the primes is of size approximately 20 decimal digits. This particular example can be easily broken by the factorization:

```
> time Factorization(86398677368792768067556452456311743331);
[ <6046864213681032211, 1>, <14288178850339607921, 1> ]
Time: 3.310
```

3. Use the above factorization to reproduce the private key `L` (generated but not printed above) for this `K`.

4. Why is the choice for which key is the public key and which key is the private key arbitrary? Practice encoding, decoding, enciphering, and deciphering with the RSA cryptosystem. Why do the functions `Enciphering` and `Deciphering` return the same values?