4 Public-Key Ciphers I

Reference Buchmann 8.1, 8.2, 8.3

A major problem for the UK military in the late 1960s was that cryptosystems were always symmetric–Alice and Bob *shared* a secret key. This changed in the 1970s with the discovery of what is now called public-key cryptosystems. They rely on the existence of special types of functions called *one-way functions* and the need for a shared secret key was removed.

4.1 One-Way Functions

Definition 4.1. A *one-way function* is a function that is easy to compute but hard to invert. In other words, if $f : X \to Y$ is a one-way function, then calculating $f(x)$ given x is easy, but calculating x given $f(x)$ is difficult; that is, given the output of the function, it is difficult to find any input which yields this output. □

Example 4.1.

The precise meanings of "easy" and "hard" can be described in terms of the resources (usually computing time) required to do the calculation.

Here, "easy" means that some algorithm can compute the function in a time proportional to some power of $\log n$, where n is the input size (i.e., bit length of the input). "Hard" means no such algorithm is known. Of course, this means that we do not know if a function really is a one-way function. All we can say is that with our current

knowledge, it is safe to assume that it is one-way. But someone coming up with a clever algorithm can always destroy this assumption.

With rare exceptions, almost the entire field of public-key cryptography rests on the existence of one-way functions.

The following are candidates for one-way functions. As we said above, it is not known whether these functions are indeed one-way. These are only conjectures supported by extensive research which has so far failed to produce any efficient inverting algorithms.

(i) *Factorising integers.* In spite of a great deal of research aimed at the construction of an efficient factoring algorithm, the best algorithms known for factoring an integer N run in time roughly

$$2^{\sqrt[3]{\log N}}.$$

If N has k bits, then this is $2^{k^{\frac{1}{3}}}$. Hence it is reasonable to believe that the function which multiplies a pair of prime numbers is one-way. The RSA cipher system is based on this function.

(ii) *Finding quadratic residues, that is, square roots in $\mathbb{Z}_m$.* The problem is as follows. Given $m > 1$ and $a \in \mathbb{Z}_m$, find $x \in \mathbb{Z}_m$ such that

$$x^2 \equiv a \bmod m.$$

(In fact, the equation doesn't always have a solution; for example,

$$x^2 \equiv 3 \bmod 5$$

has no solution.)

It can be shown that finding square roots modulo m (when they exist) is computationally equivalent to factoring m (that is, the two tasks are reducible to one another by fast algorithms). Hence, squaring modulo a composite m will be a one-way function as long as factoring is still "hard". The Rabin cryptosystem is based on this function. (Note that if m is prime, then finding square roots modulo m is easy.)

(iii) *Finding discrete logarithms.* Another computational number problem which is widely believed to be "hard" is that of finding discrete logarithms in $\mathbb{Z}_p^*$, where p is a (large!) prime. Thus exponentiation (taking powers of a generator) in $\mathbb{Z}_p^*$ is a reasonable candidate for a one-way function. The Elgamal encryption scheme is based on this function, as is the Diffie-Hellman key exchange.

Example 4.2.

4.2 The RSA Cipher

Named after Rivest, Shamir, and Adleman (1978), the RSA cipher is based on the fact that factorising numbers with large prime factors is in practice very difficult because no quick algorithm exists. As an example, take p and q to be large primes of the order of 500 digits and $n = pq$. Then, knowing only n, it is impossible *in practice* to find p and q since, with existing algorithms, it would take much longer than the lifetime of the universe to do it.

The RSA cipher is an example of a *public-key* cipher system.

The basic idea of RSA. Alice publishes two numbers n and e, which form the public key. They are not secret.

A message m (of length less than n) is then sent (publicly as well) by enciphering it as

$$c \equiv m^e \bmod n.$$

The strength of the system lies in the fact that it is impossible in practice to recover m from c.

More precisely, RSA works as follows.

How it works.

1. Alice (at the receiving end) firstly does the following:

   (a) Chooses two large primes p and q (typically of the order of 10^{100}), and calculates $n = pq$ and $\phi(n) = (p-1)(q-1)$.

   (b) Chooses e (the encryption key) so that $\gcd(e, \phi(n)) = 1$.

   (Here, e need not be large. Less than 10^4 seems okay. Choosing e is quick; a guess at a value followed by a check using Euclid's Algorithm that it is relatively prime to $\phi(n)$.)

   (c) Calculates d (the decryption key), where $ed \equiv 1 \bmod \phi(n)$. The private key is d. Again this is quickly done using Euclid's Algorithm.

   (d) Lastly, Alice makes n and e public.

2. Bob (at the sending end, and who knows the public key n and e) does the following:

   (a) Converts the message to a string of digits.

   (b) Breaks up the message into blocks of numbers $m_1, m_2, \ldots, m_k$ each less than n.

   (c) Bob then encrypts these blocks as

   $$c_i \equiv m_i^e \bmod n$$

   and sends the encrypted blocks to Alice.

   (Recall the fast-exponentiation algorithm for efficiently calculating these powers mod$n$.)

3. Alice now

   (a) calculates $m_i \equiv c_i^d \bmod n$ and

   (b) recombines the messages $m_1, m_2, \ldots, m_k$ to get m.

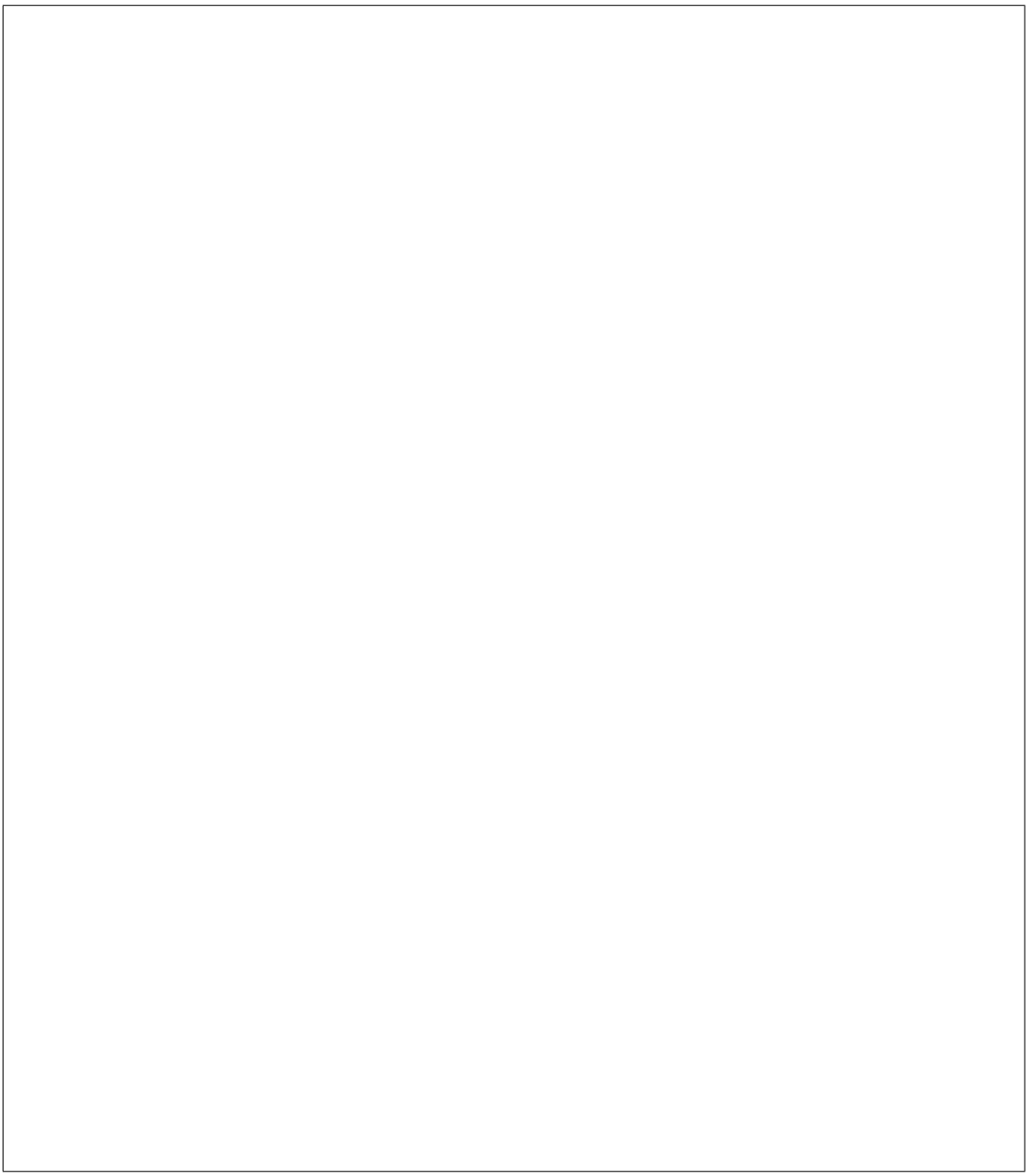
Example 4.3. An easy example to illustrate the central ideas.

Theorem 4.2. Decryption in RSA works.

Proof.

- In fact, if $\gcd(m_i, n) \neq 1$, then $\gcd(m_i, n) \in \{p, q, pq\}$ and it can be shown in each of these instances that
$$c_i^d \equiv m_i \bmod n.$$

Example 4.4. A bit more complicated but work your way through it carefully.



Example 4.5. Eve already knows n. If she additionally knows $\phi(n)$, does she know p and q?

Comments. Existing factorisation algorithms means that it is wise to choose the two primes of roughly equal bit length and this bit length should be more than 512 which means that the two primes are about 150 digits long.

It turns out that finding d is equivalent to factorising n. Thus if factoring n is truely hard, then calculating d is also hard. Hence many attacks on the RSA cryptosystem often attempt to decrypt ciphertext without trying to find d.

Historical note. Life is never fair. Clifford Cocks, a young recruit to the Government Communications Headquarters (GCHQ) at Cheltenham, UK, first discovered what is essentially now known as the RSA cryptosystem in 1973, five years before it was rediscovered by Rivest, Shamir, and Adleman. For security reasons, at the time, it wasn't allowed to be announced publicly, so Cocks never received the adulations he deserved.