9. One-way Functions and the RSA Trapdoor Permutation

In the last chapter we have seen that exponentiation in certain groups is conjectured hard to invert, i.e., the discrete logarithm in these groups is conjectured hard to compute. We now generalize this concept to an abstract concept called *one-way functions*. Closely related are so-called *trapdoor permutations*, which constitute an important foundation of many modern cryptographic systems. The most common trapdoor permutation is the RSA trapdoor permutation, which we will introduce in the following.

9.1 One-way Functions (OWFs) and Trapdoor Permutations

A *one-way function* is a deterministic function that is easy to compute, but (conjectured) hard to invert, i.e., an adversary gets a value $\mathsf{F}(x) = y$ for a randomly drawn x and tries to find a value x' such that $\mathsf{F}(x') = y$. The success probability of every efficient adversary should be negligible in $|x|$.

Definition 9.1 (One-way Functions) *An efficiently computable, deterministic function* $\mathsf{F}\colon \{0,1\}^* \to \{0,1\}^*$ *is a* one-way function *iff it is* hard to invert*, i.e., for any efficient* A*, we have that*

$$\mathsf{Pr}\left[\mathsf{F}(x) = \mathsf{F}(x');\ x \leftarrow_{\mathcal{R}} \{0,1\}^n,\ y := \mathsf{F}(x),\ x' \leftarrow \mathsf{A}(n,y)\right]$$

is negligible in n*. Here and in the following,* A *is given the parameter* n *in unary representation as usual, for reasons of complexity.* ◇

One could define one-wayness also for function ensembles indexed by a key-generation algorithm as follows: Let $\mathsf{Gen}\colon \mathbb{N} \to \mathcal{PK}$ be an efficiently computable key-generation algorithm and let $F = (\mathsf{F}(pk,\cdot))_{pk \in [\mathsf{Gen}(n)]}$ be a family of deterministic functions $\mathsf{F}(pk,\cdot)\colon \mathcal{X}_{pk} \to \mathcal{Y}_{pk}$ for finite domains and ranges $\mathcal{X}_{pk}$ and $\mathcal{Y}_{pk}$, respectively. Then F is said to be a *family of one-way functions*, or short *one-way*, iff for any efficient A, we have that

$$\mathsf{Pr}\left[\mathsf{F}(pk,x) = \mathsf{F}(pk,x');\ pk \leftarrow \mathsf{Gen}(n),\ x \leftarrow_{\mathcal{R}} \mathcal{X}_{pk},\ y := \mathsf{F}(pk,x), x' \leftarrow \mathsf{A}(n,pk,y)\right]$$

is negligible in n.

For building an encryption scheme, however, we often have to rely on some possibility to invert F, i.e., to recover the message again. This is accomplished by providing some extra information called a *trapdoor* (denoted by *sk* in the following), that enables people knowing the trapdoor to invert the function F. This is captured by the notion of *(keyed) trapdoor permutations*.

Definition 9.2 ((Keyed) Trapdoor Permutations) *Let* $\mathsf{Gen}\colon \mathbb{N} \to \mathcal{PK} \times \mathcal{SK}$ *be an efficiently computable key-generation algorithm and* $F = (\mathsf{F}(pk,\cdot)_{(pk,*) \in [\mathsf{Gen}(n)]}$ *be a family of efficiently computable, deterministic functions* $\mathsf{F}(pk,\cdot)\colon \mathcal{X}_{pk} \to \mathcal{Y}_{pk}$*. F is a* family of trapdoor permutations *iff*

- F *is one-way, i.e., for any efficient adversary* A*, we have that*

$$\mathsf{Pr}\left[\mathsf{F}(pk,x)=\mathsf{F}(pk,x');\ (pk,sk)\leftarrow\mathsf{Gen}(n),\ x\leftarrow_{\mathcal{R}}\mathcal{X}_{pk},\ y:=\mathsf{F}(pk,x),x'\leftarrow\mathsf{A}(n,pk,y)\right]$$

  is negligible, and
- F *has a* trap-door *sk, i.e., there exists an efficiently computable algorithm* G *that* inverts F *given sk: For all* $(pk,sk)\in[\mathsf{Gen}(n)]$ *and for all* $x\in\mathcal{X}_{pk}$*:*

$$\mathsf{G}(n,pk,sk,\mathsf{F}(pk,x))=x.$$

◇

Trapdoor permutations almost look like encryption schemes; however, the following example shows that they might fail completely in providing secrecy when applied in a straightforward manner. The reason is that one-way functions might leak substantial parts of the message, which is clearly not acceptable for encryption, and which does not even yield security against chosen-plaintext attack, i.e, against passive adversaries.

Proposition 9.1 *Let* F *be a OWF. Then* $\mathsf{H}(x\parallel y):=\mathsf{F}(x)\parallel y$ *(where* $|x|=\lceil\frac{|(x\|y)|}{2}\rceil$*) is a OWF as well.*

Proof. Assume that A constitutes a successful adversary against the one-wayness of H, i.e., upon receiving an input $\mathsf{F}(x)\parallel y$, it outputs $x'\parallel y'$ such that $\mathsf{F}(x)\parallel y=\mathsf{F}(x')\parallel y'$ holds with not-negligible probability. We then construct an adversary B against F as follows: It receives $\mathsf{F}(x)$ for a randomly chosen (and unknown) x, chooses a random y, and sends $\mathsf{F}(x)\parallel y$ to A. Upon receiving $x'\parallel y'$, it outputs x'.

B is obviously efficiently computable. Thus it is sufficient to calculate the success probability of B to invert $\mathsf{F}(x)$:

$$\begin{aligned}
&\mathsf{Pr}\left[\mathsf{F}(x)=\mathsf{F}(x');x\leftarrow_{\mathcal{R}}\{0,1\}^n,z:=\mathsf{F}(x),x'\leftarrow\mathsf{B}(n,z)\right] &(9.1)\\
=\ &\mathsf{Pr}\left[\mathsf{F}(x)=\mathsf{F}(x');x,y\leftarrow_{\mathcal{R}}\{0,1\}^n,\ z:=\mathsf{F}(x),\ x'\parallel y'\leftarrow\mathsf{A}(n,(z\parallel y))\right]\\
\geq\ &\mathsf{Pr}\left[\mathsf{F}(x)\parallel y=\mathsf{F}(x')\parallel y';x,y\leftarrow_{\mathcal{R}}\{0,1\}^n,\ z:=\mathsf{F}(x),\ x'\parallel y'\leftarrow\mathsf{A}(n,(z\parallel y))\right]\\
=\ &\mathsf{Pr}\left[\mathsf{H}(x\parallel y)=\mathsf{H}(x'\parallel y');x,y\leftarrow_{\mathcal{R}}\{0,1\}^n,\ z^*:=\mathsf{H}(x\parallel y),\ x'\parallel y'\leftarrow\mathsf{A}(n,z^*)\right] &(9.2)
\end{aligned}$$

We know by assumption that the probability given in Equation 9.2 is not negligible. Consequently the probability given in Equation 9.1 is not negligible which concludes the proof. ∎

Definition 9.3 (Hardcore Predicate) *A* hardcore predicate *of a function* $\mathsf{F}\colon\{0,1\}^*\to\{0,1\}^*$ *is an efficiently computable boolean predicate* $\pi\colon\{0,1\}^*\to\mathcal{BOOL}$ *such that for all efficient* A*, we have that*

$$\left|\mathsf{Pr}\left[z=\pi(x);\ x\leftarrow_{\mathcal{R}}\{0,1\}^n,\ y:=\mathsf{F}(x),\ z\leftarrow\mathsf{A}(n,y)\right]-\frac{1}{2}\right|$$

is negligible in n*.*

◇

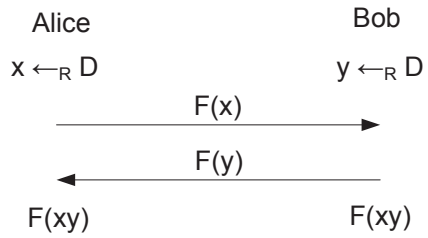

Figure 9.1: Generalized Diffie-Hellman Key Exchange

Goldreich and Levin showed the following theorem, which we give without proof: Let F be a length-preserving OWF (length-preserving means that $|\mathsf{F}(x)| = |x|$), and let $\mathsf{F}'(x \parallel r) := \mathsf{F}(x) \parallel r$ where $|x| = |r| = k$. Then

$$\pi(x \parallel r) := \sum_{i=1}^{k} x_i r_i \bmod 2$$

constitutes a hardcore predicate of F'.

Many functions currently exist that are conjectured one-way, e.g., based on assumptions such as the hardness of factoring, of the Diffie-Hellman function, or of computing the discrete logarithm, one can construct OWFs. Proving the existence of OWFs without such assumptions would however immediately imply the existence of secure encryption schemes, secure signature schemes, and would in particular prove $P \neq NP$.

We conclude by giving two simple constructions of OWFs.

Proposition 9.2 (OWFs from PRPs) *Assume $\mathsf{E} = (\mathsf{E}_n(K,\cdot))_{n\in\mathbb{N}}$ is a family of PRPs , where E_n has domain and range $\{0,1\}^n$. Then $\mathsf{F}^{\mathsf{E}}(x) = \mathsf{E}(x,0)$ for $x \in \{0,1\}^n$ is a one-way function.*

Proposition 9.3 (DLog is an OWF) *Let p be a prime and $g \in \mathbb{Z}_p^*$ an element of (large) prime order q. Then $\mathsf{F}^{\mathsf{Dlog}}(x) := g^x \bmod p$ is conjectured to be one-way. (Its inverse is $\mathsf{DLog}_g(\cdot)$ which is conjectured hard in $\mathbb{Z}_p^*$.)*

Note that $\mathsf{F}^{\mathsf{DLog}}$ has an additional property: Given $a \in \mathbb{Z}$ and $\mathsf{F}^{\mathsf{DLog}}(x)$, $\mathsf{F}^{\mathsf{DLog}}(y)$ one can easily compute $\mathsf{F}^{\mathsf{DLog}}(a\cdot x)$ and $\mathsf{F}^{\mathsf{DLog}}(x+y)$. One says that $\mathsf{F}^{\mathsf{DLog}}$ is an *additive function*. Due to additivity it can be used for Diffie-Hellman key exchange as well as for the ElGamal public-key encryption scheme, whereas one-way functions without such properties, e.g., F^{E} as constructed above, cannot be used for this purpose.

9.1.1 Generalized ELGamal and Diffie-Hellman

Diffie-Hellman key exchange and ElGamal encryption can be generalized to arbitrary additive one-way functions with domain D in an obvious manner. We only show the generalization of Diffie-Hellman key exchange in Figure 9.1 for simplicity. Note that given x and $\mathsf{F}(y)$ Alice can compute $\mathsf{F}(xy)$ by the additivity of F, similarly for Bob who holds y and $\mathsf{F}(x)$.

9.2 Arithmetic Modulo a Composite

In the following we are working in groups $\mathbb{Z}_N$ where $N \in \mathbb{N}$ is a composite. Unless otherwise stated N is the product $N = p \cdot q$ of two large (≈ 512 bit) primes that are of the same size. For a composite N let $\mathbb{Z}_N = \{0, 1, 2, \ldots, N-1\}$; addition and multiplication are defined modulo N as usual.

Extended Euclidean Algorithm Given two natural numbers a and b one can find the *greatest common divisor* $r = \gcd(a, b)$ by means of the *Euclidean algorithm*. In addition, integers x, y exist that satisfy $ax + by = r$, and these integers can be found by the *extended Euclidean algorithm* as follows:

1. Start with numbers a and b, as well as corresponding vectors $(1, 0)$ and $(0, 1)$.
2. Divide the larger of the two numbers a and b by the smaller using division with remainder. Call this quotient q.
3. Subtract q times the smaller from the larger number.
4. Subtract q times the vector corresponding to the smaller number from the vector corresponding to the larger number.
5. Repeat steps 2 through 4 until one of the numbers equals zero. The vector that was changed in the last step consist of the two desired numbers x, y.

The following examples shows the algorithm running on numbers $a = 53$ and $b = 30$:

53	30	$(1, 0)$	$(0, 1)$
$53 - 1 \cdot 30 = 23$	30	$(1, 0) - 1 \cdot (0, 1) = (1, -1)$	$(0, 1)$
23	$30 - 1 \cdot 23 = 7$	$(1, -1)$	$(0, 1) - 1 \cdot (1, -1) = (-1, 2)$
$23 - 3 \cdot 7 = 2$	7	$(1, -1) - 3 \cdot (-1, 2) = (4, -7)$	$(-1, 2)$
2	$7 - 3 \cdot 2 = 1$	$(4, -7)$	$(-1, 2) - 3 \cdot (4, -7) = (-13, 23)$
$2 - 2 \cdot 1 = 0$	1	$(4, -7) - 2 \cdot (-13, 23) = (30, -53)$	$(-13, 23)$

Thus we see that $-13 \cdot 53 + 23 \cdot 30 = -689 + 690 = 1$.

The *inverse* of $x \in \mathbb{Z}_N$ is an element $y \in \mathbb{Z}_N$ such that $x \cdot y = 1 \bmod N$. We write x^{-1} instead of y provided that the inverse exists.

Proposition 9.4 (Inverse) *An element $x \in \mathbb{Z}_N$ has an inverse if and only if* $\gcd(x, N) = 1$.

Proof. If $\gcd(x, N) = 1$ then there exist two integers $a, b \in \mathbb{Z}$ such that $ax + bN = 1$. Reducing this relation modulo N leads to $ax = 1 \bmod N$, hence $a = x^{-1} \bmod N$.

If, on the other hand, there exists a such that $ax = 1 \bmod N$, then there exists k such that $ax = kN + 1$ or equivalently $ax - kN = 1$. By definition $\gcd(x, N) | ax$ and $\gcd(x, N) | kN$, hence we have that $\gcd(x, N) | 1$, which is only possible if $\gcd(x, N) = 1$. ∎

This proof also yields an efficient method for constructing the inverse since the extended Euclidean algorithm can be used to efficiently construct the value a (which is the inverse of x modulo N). This inversion algorithm also works in $\mathbb{Z}_p$ for a prime p and is more efficient than inverting x by computing $x^{p-2} \bmod p$. The ability to compute an inverse in particular entails an algorithm for solving linear equations $a \cdot x = b \bmod N$. The solution is $x = b \cdot a^{-1} \bmod N$ where a^{-1} can now be computed using the extended Euclidean algorithm (provided that a^{-1} exists).

We note that if we know an element $x \in \mathbb{Z}_N$ with $0 \neq x \notin \mathbb{Z}_N^*$ then we can factor N, since $\gcd(x, N)$ constitutes a non-trivial factor of N.

We now investigate how many elements in $\mathbb{Z}_N$ are invertible. While in $\mathbb{Z}_p$ all elements except for 0 are invertible, this is no longer true for $\mathbb{Z}_N$.

Definition 9.4 (Eulers Totient Function) *Let $\varphi(N) := |\mathbb{Z}_N^*|$ the number of invertible elements in $\mathbb{Z}_N^*$.* ◇

The function $\varphi(N)$ turns out to be easily computable, provided that the factorization of N is known:

1. We already know that $\varphi(p) = p - 1$ for any prime p.
2. We have $\varphi(pq) = (p-1)(q-1)$ if p and q are relatively prime.
3. We have $\varphi(p^e) = p^e - p^{e-1}$ for prime p and $e \in \mathbb{N}$.

These rules can be summarized to the following compact form for computing $\varphi(N)$.

Proposition 9.5 (Computing $\varphi(N)$) *Let $N = p_1^{e_1} \cdot \ldots \cdot p_m^{e_m}$ denote the prime factorization of N. Then*

$$\varphi(N) = N \cdot \prod_{i=1}^{m} (1 - \frac{1}{p_i}).$$

The following theorem is a generalization of Fermat's Little Theorem to groups of arbitrary order:

Theorem 9.1 (Euler) *For any $a \in \mathbb{Z}_N^*$ we have that $a^{\varphi(N)} = 1 \bmod N$.* □

Theorem 9.2 (Chinese Remainder Theorem (CRT) for $N = pq$) *Let $p \neq q$ be primes and let $N = pq$. Given $x_p \in \mathbb{Z}_p$ and $x_q \in \mathbb{Z}_q$, there exists a unique element $s \in \mathbb{Z}_N$ such that $s = x_p \bmod p$ and $s = x_q \bmod q$. Furthermore, s can be computed efficiently by the following algorithm: On input (p, q, x_p, x_q) with p, q prime, $p \neq q$, and $x_p \in \mathbb{Z}_p, x_q \in \mathbb{Z}_q$:*

- *Use the extended Euclidean algorithm to compute $u, v \in \mathbb{Z}$ with $up + vq = 1$.*
- *Output $x := upx_q + vqx_p \bmod N$.*

□

The CRT shows that each element $s \in \mathbb{Z}_N$ can be viewed as a unique pair (x_p, x_q) where $x_p = s \bmod p$ and $x_q = s \bmod q$. The uniqueness guarantee shows that each pair $(x_p, x_q) \in \mathbb{Z}_p \times \mathbb{Z}_q$ corresponds to exactly one element of $\mathbb{Z}_N$. For example, the pair $(1, 1)$ corresponds to $1 \in \mathbb{Z}_N$.

We conclude with the following simple proposition; a simplified version thereof was already given on one of the exercise sheets.

Proposition 9.6 *Let G be a group and $g \in G$. For all $x, y \in \mathbb{Z}$ it holds that*

$$g^x = g^y \quad \textit{iff} \quad x = y \bmod \mathsf{ord}(g)$$

Proof. Let $\Omega := \mathsf{ord}(g)$.
"⇐": Let $x = y \bmod \Omega$. Then there exists $k \in \mathbb{N}$ such that $y = x + k \cdot \Omega$. Consequently,

$$g^y = g^{x+k\cdot\Omega} = g^x \cdot (g^\Omega)^k = g^x \cdot 1^k = g^x.$$

"⇒": Let $x - y = q \cdot \Omega + r$ where $0 \leq r < \Omega$. Then

$$1 = g^{x-y} = g^{q\cdot\Omega+r} = g^r.$$

Consequently $r = 0$, otherwise we had a contradiction to the minimality of Ω. Thus $x = y \bmod \Omega$.
■

9.3 The RSA Trapdoor Permutation

In this section we will investigate the RSA trapdoor permutation, one of the most famous and most widely deployed one-way functions. The RSA trapdoor permutation was invented by Rivest, Shamir, Adleman in 1977, and can be used to construct various cryptographic schemes. However, we will see that one has to be careful when using this function for constructing encryption schemes since naive constructions result in insecure encryption schemes (even if they are sometimes claimed to be good in several books). We will see later in this chapter how RSA can be used to construct good encryption schemes.

Definition 9.5 (RSA Trapdoor Permutation) *Pick random n-bit primes p and q (in practice often 512 bits) and let $N = pq$. Choose an arbitrary value e such that $\gcd(e, \varphi(N)) = 1$ and compute d such that $ed = 1 \bmod \varphi(N)$.*

- *The public information is $pk = (N, e)$, and the RSA trapdoor permutation $\mathsf{F}^{\mathsf{RSA}} \colon \mathbb{Z}_N \to \mathbb{Z}_N$ for this pk is defined as $\mathsf{F}^{\mathsf{RSA}}(x) := x^e \bmod N$.*
- *The trapdoor information is $sk = d$, and $\mathsf{D}^{\mathsf{RSA}}(y) := y^d \bmod N$ is the inverse of $\mathsf{E}^{\mathsf{RSA}}$.*

◇

Before proving that $\mathsf{D}^{\mathsf{RSA}}$ is the inverse of $\mathsf{F}^{\mathsf{RSA}}$ we prove the following lemma:

Lemma 9.1 *For N, e, d as constructed in Definition 9.5 and for arbitrary $m \in \mathbb{Z}_N$ it holds that*

$$m^{ed} = m \bmod N.$$

□

Proof. By the Chinese Remainder Theorem, it is sufficient to show $m^{ed} = m \bmod p$ and $m^{ed} = m \bmod q$. For symmetry reasons, we only show this for p.

- If $m = 0 \bmod p$, then obviously $0^{ed} = 0 \bmod p$, thus nothing needs to be proved.
- If $m \neq 0 \bmod p$, then $m \in \mathbb{Z}_p^*$. Since $ed = 1 \bmod \varphi(N)$ and $\varphi(N) = (p-1)(q-1)$, there exists $k \in \mathbb{Z}$ such that $ed = k(p-1)(q-1) + 1$. Thus we obtain

$$m^{ed} = m^{1+k(p-1)(q-1)} = m \cdot (m^{p-1})^{k(q-1)} = m \cdot 1^{k(q-1)} = m \bmod p.$$

This finishes the proof. ■

Now simple calculation shows that $\mathsf{D}^{\mathsf{RSA}}$ is the inverse of $\mathsf{F}^{\mathsf{RSA}}$: For any $m \in \mathbb{Z}_N$

$$\mathsf{D}^{\mathsf{RSA}}(\mathsf{F}^{\mathsf{RSA}}(m)) = (m^e)^d = m^{ed} = m \bmod N.$$

As we did for discrete logarithms, we explicitly state the assumption that the RSA trapdoor permutation is one-way. This is known as the *RSA assumption*.

Definition 9.6 (RSA Assumption) *Given random n-bit primes p and q, $N := pq$, a value e such that $\gcd(e, \varphi(N)) = 1$, and a random number $c \in \mathbb{Z}_N$, there does not exist any efficient algorithm that computes the e'th root of c modulo N up to negligible probability.* ◇

One of the main open questions in cryptography is the following: Given an algorithm for computing e'th roots modulo N, i.e., for inverting RSA, does it imply a factoring algorithm? There is some evidence that this might not be the case, i.e., that breaking RSA is indeed easier than factoring (Boneh-Venkatesan 1998). However, the best algorithm which is known to invert the RSA trapdoor permutation without knowing d is to first factor $N = p \cdot q$ (this is conjectured hard) and subsequently to compute e-th roots modulo p and q (this is easily doable).

9.3.1 Naive Encryption with RSA

It is tempting to use the following construction for encryption: One generates (pk, sk) as in Definition 9.5, publishes pk, and keeps sk secret. Then encryption and decryption are realized by applying and inverting the RSA trapdoor permutation, respectively: Given a message $m \in \mathbb{Z}_N$ or a ciphertext $c \in \mathbb{Z}_N$, one computes $\mathsf{E}(pk, m) := m^e \bmod N$ or $\mathsf{D}(sk, c) := c^d \bmod N$, respectively. Naive RSA encryption is obviously not CPA-secure, as it is deterministic. Furthermore, it leaks concrete bits, e.g., the so-called Jacobi symbol of the message. It is also highly vulnerable to active attacks, e.g., chosen-ciphertext attacks. There are simple attacks that completely retrieve the message for a given ciphertext.

For the first such attack the attacker asks for the decryption of two ciphertexts that are constructed as follows. Denote the given ciphertext the adversary is trying to decrypt by c.

- Choose $c_1 \in \mathbb{Z}_N^*$ arbitrary and ask for its decryption, yielding a message m_1.
- Let $c_2 := c \cdot c_1^{-1}$ and ask for its decryption, yielding a message m_2.
- Output $m_1 \cdot m_2$.

An easy calculation shows that this attack retrieves the message $m = c^d \bmod N$ from the ciphertext c:

$$m_1 \cdot m_2 = c_1^d \cdot c_2^d = c_1^d \cdot c^d \cdot c_1^{-d} = 1 \cdot c^d = m^{ed} = m.$$

One can do even better: Even with only one decryption query one can decrypt an arbitrary ciphertext as the following attack shows. Again the adversary is given a ciphertext c.

- Choose $m_1 \in \mathbb{Z}_N^*$ arbitrary and let $c_1 := m_1^e \bmod N$.
- Let $c_2 := c \cdot c_1^{-1}$ and ask for its decryption, yielding a message m_2.
- Compute $m_1 \cdot m_2$.

Again a simple calculation shows that this attack retrieves the message $m = c^d \bmod N$ from the ciphertext c:

$$m_1 \cdot m_2 = m_1 \cdot c_2^d = m_1 \cdot c^d \cdot c_1^{-d} = m_1 \cdot c^d \cdot m_1^{-1} = m^{ed} = m.$$