Today we will see some concrete one-way function candidates that arise from number theory, and abstract out some of their other special properties that will be useful when we proceed to investigate pseudorandomness.

1 Collections of OWFs

Our generic definition of a one-way function is concise, and very useful for complexity-theoretic crypto. However, it tends not to be as appropriate for the kinds of hard functions that we use in "real-life" crypto; below we give a more flexible definition. (In your homework, you will show that the generic OWF definition is equivalent to this one.)

Definition 1.1. A *collection of one-way functions* is a family $F = \{f_s \colon D_s \to R_s\}_{s \in S}$ satisfying the following conditions:

1. *Easy to sample a function:* there is a PPT algorithm S such that $\mathsf{S}()$ outputs some $s \in S$ (according to some arbitrary distribution).
2. *Easy to sample from domain:* there is a PPT algorithm D such that $\mathsf{D}(s)$ outputs some $x \in D_s$ (according to some arbitrary distribution).
3. *Easy to evaluate function:* there is a PPT algorithm F such that $\mathsf{F}(s, x) = f_s(x)$ for all $s \in S, x \in D_s$.
4. *Hard to invert:* for any non-uniform PPT algorithm $\mathcal{I}$,
$$\Pr_{s \leftarrow \mathsf{S}(1^n), x \leftarrow \mathsf{D}(s)} \left[\mathcal{I}(s, f_s(x)) \in f_s^{-1}(f_s(x))\right] = \operatorname{negl}(n).$$

For example, the subset-sum function f_{ss} is more naturally defined as a collection, as follows. Let $S_n = (\mathbb{Z}_N)^n$ where $N = 2^n$, and let the full index set $S = \cup_{n=1}^{\infty} S_n$. Define the domain $D_{\vec{a}} = \{0,1\}^n$ and the range $R_{\vec{a}} = \mathbb{Z}_N$, for all $\vec{a} = (a_1, \ldots, a_n) \in S_n$. The corresponding function is defined as
$$f_{\vec{a}}(x) = \sum_{i=1}^{n} a_i \cdot x_i \bmod N.$$
The algorithms S (function sampler), D (domain sampler), and F (function evaluator) are all straightforward.

In the remainder of the lecture, we will see other examples of OWF collections (some with other special properties) that arise from number theory.

2 Number Theory Background

Definition 2.1. For positive integers $a, b \in \mathbb{N}$, their *greatest common divisor* $d = \gcd(a, b)$ is the largest integer d such that $d \mid a$ and $d \mid b$.

As a consequence of Algorithm 1 below, there always exist integers $x, y \in \mathbb{Z}$ such that $ax + by = \gcd(a, b)$. We say that a and b are *co-prime* (or *relatively prime*) if $\gcd(a, b) = 1$, i.e., $ax = 1 \bmod b$. From this, x is the multiplicative inverse of a modulo b, and likewise y is the multiplicative inverse of b modulo a. The following deterministic algorithm shows that $\gcd(a, b)$ (and additionally, the integers x and y) can be computed efficiently.

Algorithm 1 Algorithm $\mathsf{ExtendedEuclid}(a, b)$ for computing the greatest common divisor of a and b.

Input: Positive integers $a \geq b > 0$.
Output: $(x, y) \in \mathbb{Z}^2$ such that $ax + by = \gcd(a, b)$.

if $b \mid a$ **then**
    return $(0, 1)$
else
    Let $a = b \cdot q + r$ for $r \in \{1, \ldots, b - 1\}$
    $(x', y') \leftarrow \mathsf{ExtendedEuclid}(b, r)$
    return $(y', x' - q \cdot y')$
end if

Theorem 2.2. $\mathsf{ExtendedEuclid}$ *is correct and runs in polynomial time in the* lengths *of a and b, i.e., in $poly(\log a + \log b)$ time.*

Proof. For correctness, we argue by induction on the second argument b. Clearly the algorithm is correct when $b = 1$. If $b \mid a$, then $\gcd(a, b) = b$, hence $\mathsf{ExtendedEuclid}$ correctly returns $(0, 1)$. If $b \nmid a$ then by the inductive hypothesis (using $b > r$), the recursive call correctly returns (x', y') such that $bx' + ry' = \gcd(b, r)$. It can be checked that $\gcd(a, b) = \gcd(b, r)$, because any common divisor of a and b is also a divisor of r. Finally, observe that

$$\gcd(b, r) = bx' + ry' = bx' + (a - b \cdot q)y' = ay' + (x' - q \cdot y')b.$$

Hence $\mathsf{ExtendedEuclid}$ correctly returns $(y', x' - q \cdot y')$.

For the running time, observe that all the basic operations (not including the recursive call) can be implemented in polynomial time. The following claim establishes the overall efficiency.

Claim 2.3. *For $2^n > a \geq b > 0$,* $\mathsf{ExtendedEuclid}$ *makes at most $2n$ recursive calls.*

We use induction. The claim is true when $a < 2^1$. Suppose the claim is true for all $a < 2^n$, and suppose $a < 2^{n+1}$. Two cases arise:

- If $b < 2^n$, the first recursive call is on (b, r). Since $b < 2^n$, by the inductive hypothesis we make at most $2n$ more recursive calls. Hence the total number of recursive calls is at most $1 + 2n < 2(n + 1)$.
- If $b \geq 2^n$, i.e., $2^{n+1} > a \geq b \geq 2^n$, we have $a = b \cdot 1 + r$ for $r = a - b < 2^n < b$. The first recursive call is on $(b \geq 2^n, r < 2^n)$. In turn, its recursive call uses $r < 2^n$ as its first parameter. By the inductive hypothesis, the number of recursive calls following the second one is at most $2n$. Hence the total number of recursive calls is at most $2 + 2n \leq 2(n + 1)$. $\square$

We frequently work with the ring $(\mathbb{Z}_N, +, \cdot)$ of integers modulo a positive integer N.

Lemma 2.4 (Chinese remainder theorem, special case)**.** *Let $N = p \cdot q$ for distinct primes p, q. The ring $\mathbb{Z}_N$ is isomorphic to the product ring $\mathbb{Z}_p \times \mathbb{Z}_q$, via the isomorphism $h(x) = (x \bmod p, x \bmod q)$.*

A few remarks about the above lemma:

- In the product ring $\mathbb{Z}_p \times \mathbb{Z}_q$, addition and multiplication are coordinate-wise.

- Clearly the isomorphism h is efficiently computable. Less obvious is that it is also efficiently *invertible*. Suppose we know some elements $c_p, c_q \in \mathbb{Z}_N$ such that $h(c_p) = (1, 0)$ and $h(c_q) = (0, 1)$; such a pair is sometimes called a *CRT basis*. Then given $(x, y) \in \mathbb{Z}_p \times \mathbb{Z}_q$, it is easy to see that $h^{-1}(x, y) = x \cdot c_p + y \cdot c_q$. Exercise: show how to compute c_p, c_q efficiently (hint: use ExtendedEuclid on p, q).

Definition 2.5. The multiplicative group $\mathbb{Z}_N^* := \{x \in \mathbb{Z}_N : x \text{ is invertible mod } N, \text{ i.e., } \gcd(x, N) = 1\}$.

Here are some useful facts about the multiplicative group $\mathbb{Z}_N^*$:

- For a prime p, $\mathbb{Z}_p^* = \{1, \ldots, p - 1\}$.
- When $N = pq$ for distinct primes p, q, we have $\mathbb{Z}_N^* \cong \mathbb{Z}_p^* \times \mathbb{Z}_q^*$.

Definition 2.6. For $N \in \mathbb{Z}^+$, Euler's *totient function* $\varphi(n)$ is defined to be $|\mathbb{Z}_N^*|$, i.e., the number of positive integers $a \leq n$ relatively prime to n.

Here are some useful facts about the totient function:

- For a prime p, we have $\varphi(p) = p - 1$.
- For a prime p and positive integer a, we have $\varphi(p^a) = (p - 1)p^{a-1} = p^a - p^{a-1}$.
- If $gcd(a, b) = 1$, then $\varphi(a \cdot b) = \varphi(a) \cdot \varphi(b)$.

Definition 2.7. The subgroup of *quadratic residues* is defined as

$$\mathbb{QR}_N^* = \{y \in \mathbb{Z}_N^* : \exists\, x \in \mathbb{Z}_N^* \text{ s.t. } y = x^2 \bmod N\} \subseteq \mathbb{Z}_N^*.$$

Here are some useful facts about $\mathbb{QR}_N^*$:

- For an odd prime p, $|\mathbb{QR}_p^*| = \frac{p-1}{2}$, because $x \mapsto x^2$ is 2-to-1 over $\mathbb{Z}_p^*$. (Exercise: prove this.)
- When $N = pq$ for distinct odd primes p, q, we have $\mathbb{QR}_N^* \cong \mathbb{QR}_p^* \times \mathbb{QR}_q^*$, hence $|\mathbb{QR}_N^*| = \frac{p-1}{2} \cdot \frac{q-1}{2}$.
- For an odd prime p, we have $-1 \in \mathbb{QR}_p^*$ if and only if $p = 1 \bmod 4$.

Question 1. Verify that the CRT isomorphism works in $\mathbb{Z}_{15}$ by checking that $h(7 \cdot 9) = h(7) \cdot h(9)$.

Question 2. Show that $h^{-1}(x, y) = x \cdot c_p + y \cdot c_q$.

3 Factoring-Related Functions

We can abstract out a modulus generation algorithm S, which given the security parameter 1^n outputs the product N of two primes p, q. For example, S might choose p and q to be uniformly random and independent n-bit primes.

Rabin's function $f_N : \mathbb{Z}_N^* \to \mathbb{QR}_N^*$ is defined as follows:

$$f_N(x) = x^2 \bmod N.$$

Precisely defining the collection according to Definition 1.1 is a simple exercise. Note that f_N is 4-to-1, because each $y \in \mathbb{QR}_N^*$ has two square roots modulo p, and two modulo q.

Theorem 3.1. *If factoring is hard (with respect to* S*), then the Rabin collection (with function generator* S*) is one-way.*

Proof. First, as already discussed it is easy to generate a function, sample its domain, and evaluate the function. The main fact we use to prove one-wayness is the following.

Claim 3.2. *Let* $N = pq$ *be the product of distinct odd primes. Given any* $x_1, x_2 \in \mathbb{Z}_N^*$ *such that* $x_1^2 = x_2^2 \bmod N$ *but* $x_1 \neq \pm x_2 \bmod N$*, the factors of* N *can be computed efficiently.*

Proof of Claim. We have $x_1^2 = x_2^2 \bmod p$ and $x_1^2 = x_2^2 \bmod q$, which implies $x_1 = \pm x_2 \bmod p$ and $x_1 = \pm x_2 \bmod q$. But we cannot have both $+$ or both $-$, by assumption. Wlog, we have $x_1 = +x_2 \bmod p$ and $x_1 = -x_2 \bmod q$. Then $p \mid (x_1 - x_2)$ but $q \nmid (x_1 - x_2)$, otherwise we'd have $q \mid (2x_2) \Rightarrow q \mid x_2 \Rightarrow x_2 \notin \mathbb{Z}_N^*$. Then $\gcd(x_1 - x_2, N) = p$, which we can compute efficiently. ☐

Continuing with the proof of Theorem 3.1, we prove one-wayness by contrapositive, via a reduction. Assuming we have an inverter for the Rabin function, the idea is to choose our own $x_1 \in \mathbb{Z}_N^*$ and invoke the inverter on $y = f_N(x_1) = x_1^2 \bmod N$. The square root x_2 it returns will be $\neq \pm x_1$, with probability $1/2$. In such a case, we get the prime factorization of N by Claim 3.2. We now proceed more formally.

Assume a non-uniform PPT inverter $\mathcal{I}$ violating the one-wayness of the Rabin collection, i.e.,

$$\Pr_{N \leftarrow \mathsf{S}(1^n), x \leftarrow \mathbb{Z}_N^*}[\mathcal{I}(N, y = x^2 \bmod N) \in \sqrt{y} \bmod N] = \delta(n)$$

is non-negligible.

Our factoring algorithm $\mathcal{A}(N)$ works as follows: first, generate a uniform $x_1 \leftarrow \mathbb{Z}_N^*$. Let $y = x_1^2 \bmod N$ and let $x_2 \leftarrow \mathcal{I}(N, y)$. If $x_2^2 = y \bmod N$ but $x_1 \neq \pm x_2 \bmod N$, then compute the factorization of N by Claim 3.2.

We now analyze the reduction. First, N and y are distributed as expected, so $\mathcal{I}$ outputs x_2 such that $x_2^2 = y \bmod N$ with probability δ. Conditioned on the fixed value of y, there are four possible values for x_1, each equally likely by construction. So we have $x_2^2 = y \bmod N$ and $x_2 \neq \pm x_1 \bmod N$ with prob $\delta/2$, which is non-negligible by assumption. ☐

Suppose $p, q = 3 \bmod 4$. Then -1 is not a square modulo p (respectively, q). So for any $x \in \mathbb{Z}_p^*$ (resp., $\mathbb{Z}_q^*$), exactly one of $\pm x$ is a square modulo p (resp., q). From this it can be seen that if we restrict the Rabin function to have domain $\mathbb{QR}_N^*$, i.e., $f_N \colon \mathbb{QR}_N^* \to \mathbb{QR}_N^*$, it becomes a *permutation* (bijection).

Question: Our proof that f_N is one-way used (quite essentially) the fact that f_N is 4-to-1. Now that we have changed its domain to make f_N a permutation, is the proof still valid?

Definition 3.3 (One-Way Permutation)**.** A collection $F = \{f_s \colon D_s \to D_s\}_{s \in S}$ is a collection of *one-way permutations* if it is a collection of one-way functions f_s under the *uniform* distribution over D_s, and each f_s is a *permutation* of D_s (i.e., a bijection).

Answers

Question 1. Verify that the CRT isomorphism works in $\mathbb{Z}_{15}$ by checking that $h(7 \cdot 9) = h(7) \cdot h(9)$.

Answer. Note that 15 is the product of two primes: 3 and 5, so $\mathbb{Z}_{15} \cong \mathbb{Z}_3 \times \mathbb{Z}_5$. First, we consider $h(7 \cdot 9)$. $7 \cdot 9 \equiv 3 \pmod{15}$ and $h(3) = (3 \bmod 3, 3 \bmod 5) = (0, 3)$. Next, $h(7) = (7 \bmod 3, 7 \bmod 5) = (1, 2)$ and $h(9) = (9 \bmod 3, 9 \bmod 5) = (0, 4)$. Finally we multiply the pairs elementwise, recalling that the first elements of each pair are from $\mathbb{Z}_3$ and the second elements are from $\mathbb{Z}_5$. $(1, 2) \cdot (0, 4) = (0, 3)$ as expected.

Question 2. Show that $h^{-1}(x, y) = x \cdot c_p + y \cdot c_q$.

Answer. Consider,

$$\begin{aligned}
(x, y) &= (x, 0) + (0, y) \\
&= h(x) \cdot (1, 0) + h(y) \cdot (0, 1) \\
&= h(x) \cdot h(c_p) + h(y) \cdot h(c_q) \\
&= h(x \cdot c_p + y \cdot c_q) \\
h^{-1}(x, y) &= x \cdot c_p + y \cdot c_q
\end{aligned}$$