Exercise Sheet 12

Cryptography and Security 2025

Exercise 1 Generating Prime Numbers

We recall that if we pick a random number in $\{1, 2, \ldots, N\}$, the probability that it is prime is approximately $\frac{1}{\ln N}$.

We want to generate prime numbers p and q for an RSA modulus with exponent $e = 3$. To generate one ℓ-bit prime number, we iteratively pick a random number between $2^{\ell-1}$ and $2^\ell - 1$ until we find a prime number:

GenPrime(ℓ)
1: **repeat**
2: pick $p \in \{2^{\ell-1}, 2^{\ell-1}+1, \ldots, 2^\ell - 2, 2^\ell - 1\}$ at random
3: **until** p is prime
4: output p

Then, to generate a 2ℓ-bit RSA key, we proceed as follows:

GenRSA(ℓ)
1: **repeat**
2: $p = \mathsf{GenPrime}(\ell)$
3: $q = \mathsf{GenPrime}(\ell)$
4: **until** $e = 3$ is a valid exponent with the RSA modulus pq
5: output p, q

In this exercise, we assume that ℓ is large enough for the RSA security.

1. Estimate the probability that a randomly selected element from $\{2^{\ell-1}, 2^{\ell-1}+1, \ldots, 2^\ell - 2, 2^\ell - 1\}$ is prime.

2. Show that the GenPrime algorithm can be speeded up by a factor 2 by selecting random elements in $\{2^{\ell-1} + 1, 2^{\ell-1} + 3, \ldots, 2^\ell - 3, 2^\ell - 1\}$.

3. Show that $e = 3$ is a valid RSA exponent if and only if p and q are equal to 2 modulo 3.

4. Consider the following algorithm:

   GenRSA$'$(ℓ)
   1: **repeat**
   2: $p = \mathsf{GenPrime}(\ell)$
   3: **until** $p \bmod 3 = 2$
   4: **repeat**
   5: $q = \mathsf{GenPrime}(\ell)$
   6: **until** $q \bmod 3 = 2$
   7: output p, q

   Show that it produces equivalent outputs to GenRSA but with a twice lower expected complexity.

5. The previous way to generate prime numbers is equivalent to using the following new algorithm:

   GenPrime$'$(ℓ)
   1: **repeat**

2: pick $p \in \{2^{\ell-1}, 2^{\ell-1}+1, \ldots, 2^{\ell}-2, 2^{\ell}-1\}$ at random
3: **until** p is prime and $p \bmod 3 = 2$
4: output p

Propose another algorithm $\mathsf{GenPrime}''$ (we expect a full description of the algorithm in the same style as $\mathsf{GenPrime}'$) to generate the prime numbers which is about 6 times faster than $\mathsf{GenPrime}'$. Conclude that GenRSA with this new algorithm instead of $\mathsf{GenPrime}$ is speeded up by a factor of about 12.

HINT: a Chinese proverb says that if you have two requirements at the same time, maybe you should combine them into a single requirement.

6. Propose a way to speed up $\mathsf{GenPrime}''$ by a factor $\frac{4}{5} \times \frac{6}{7} \times \frac{10}{11} \times \frac{12}{13} \times \frac{16}{17} \times \frac{18}{19} \times \frac{22}{23} \times \cdots$

Exercise 2 Security Issues in ECDSA

In Sony PS3, the bootup code can be changed when it comes from a valid signature from the manufacturer. The signature scheme is ECDSA. We briefly recall the scheme here.

The public key consists of a prime number n, a finite field $\mathsf{GF}(q)$, an elliptic curve over this field, a generator G of order n, and another point Q. The secret key is an integer $d \in \mathbf{Z}_n^*$ such that $Q = dG$. To sign a message M, the signer picks $k \in \mathbf{Z}_n^*$, computes the point $(x_1, y_1) = kG$, then $r = \bar{x}_1 \bmod n$ given a function $x \mapsto \bar{x}$ from $\mathsf{GF}(q)$ to $\mathbf{Z}$, and finally $s = \frac{H(M)+dr}{k} \bmod n$ given a hash function H. If $r = 0$ or $s = 0$, the signer restarts the computation until $r \neq 0$ and $s \neq 0$. The signature is the pair (r, s). To verify a signature (r, s) for a message M, the verifier checks that $Q \neq \mathcal{O}$, that Q lies on the curve, that $nQ = \mathcal{O}$, and that $r \in \mathbf{Z}_n^*$. Then, he computes $u_1 = \frac{H(M)}{s} \bmod n$, $u_2 = \frac{r}{s} \bmod n$, and $(x_1, y_1) = u_1 G + u_2 Q$, and finally checks that $r = \bar{x}_1 \bmod n$.

1. ECDSA manipulates values of different *types* such as *points*, *field elements*, *integers*, etc. What are the types of k, r, s, y_1, $H(M)$? What is $\mathcal{O}$?

2. What kind of finite fields can we use in practice? Cite at least two and briefly explain how to perform computations in these structures.

3. If a key is valid and a signature is produced by the signing algorithm, show that the verification algorithm will accept the signature.

4. Why is it hard to recover the secret key given the public key?

5. For some reasons, the manufacturer produced signatures for different codes using the same random k. Given two codes M and M' and their signatures (r, s) and (r', s'), respectively, show that an adversary can recover d.