Cryptography and Security — Midterm Exam

Serge Vaudenay

26.11.2014

- duration: 1h45
- no documents allowed, except one 2-sided sheet of handwritten notes
- a pocket calculator is allowed
- communication devices are not allowed
- the exam invigilators will **not** answer any technical question during the exam
- readability and style of writing will be part of the grade

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.

Q.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.

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

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

Q.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.

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

$\mathsf{GenPrime}'(\ell)$
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 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.

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

2 Encoding Messages in Elliptic Curves

We consider the ElGamal cryptosystem over an elliptic curve. I.e., we work over a field $\mathbf{Z}_p$, use parameters a, b to define the curve $y^2 = x^3 + ax + b$, and use a generator P of the curve, who has a prime order n. (We recall that n is close to p, due to the Hasse Theorem.) Given a secret key d, the public key is $Q = dP$. Normally, we encrypt group elements. To encrypt a point M in the curve, we compute $R = rP$ for $r \in_U \mathbf{Z}_n$ and $S = M + rQ$. The ciphertext is (R, S).

We want to encrypt bitstrings (of fixed length which is less than $\log_2 n$). To encrypt a bitstring m, we map it to a point on the elliptic curve $M = \mathsf{map}(m)$ then encrypt M. We assume that map is efficiently invertible so that after decrypting (R, S) we can apply map^{-1} to obtain m. In this exercise, we consider the problem of defining map.

Q.1 Given the secret d and the parameters (p, a, b, n, P) recall how the above ElGamal cryptosystem is constructed from the semi-static Diffie-Hellman protocol. Then, give the method to decrypt the ciphertext (R, S).

Q.2 One convenient way to map an element of $\mathbf{Z}_n$ to the elliptic curve is to multiple the integer by P. We define a function integer to convert a bitstring into an integer. I.e., $\mathsf{integer}(m) = \sum_{i=1}^{|m|} m_i 2^{|m|-i}$, where $|m|$ is the length of the bitstring m and m_i is the ith bit of m.

List the requirements on the map function to make the cryptosystem usable.

Say if the function $\mathsf{map}(m) = \mathsf{integer}(m)P$ satisfies them.

Q.3 We now consider $\mathsf{map}(m) = (x, y)$ where $x = \mathsf{integer}(m)$, y is the smallest square root of $x^3 + ax + b$, and integer converts a bitstring into an integer. By reviewing the requirements on map, what do you think of this function?

Q.4 Let k be a small (public) constant. We change the previous construction by taking x be the smallest integer at least equal to $2^k \mathsf{integer}(m)$ such that $x^3 + ax + b$ is a quadratic

residue. Review again the required properties on map and provide algorithms to compute map and map^{-1}.

Q.5 Assuming that p has 256 bits, propose a value (as small as possible) for k so that the previous construction should work with probability at least $1 - 2^{-80}$.
HINT: for this question, assume that $x \mapsto x^3 + ax + b$ maps intervals of size 2^k to "random values" in $\mathbf{Z}_p$.