Exercises lecture 2/JL Roch - Complexity

Exercise 1. Merkle-Hellman

1. *size = n = 1000 bits.*
2. *Algorithm to build its secret integers a_i and m :*
   (a) *Choose a_0 a random c bit number (c for instance 64). Let $s_0 = a_0$.*
   (b) *for $i = 1, \dots, n-1$: let a_i =rand($\{s_{i-1} + 1, \dots, 2.s_{i-1}\}$)*
   (c) *let m =rand($\{s_n, \dots, 2.s_n\}$)*
3. *We have $a_i \leq 2\sum_{j=0}^{i-1} a_j = 2.a_{i-1} + 2\sum_{j=0}^{i?2} a_j \leq 4a_{i-1} \leq c.4^i$.*
4. *The size of the cipher text C is $\log_2 m$ bits : but $m \leq 2\sum_{j=0}^{n?1} a_j \leq \frac{2c4^n}{3}$. Thus $\log_2 m \leq 2n + \log_2 c$. The size of the cipher text is about twice the size of the plain text.*
5. *Algorithm Encode(X[0..n[) {*
   for (S=0, i=0; i<n; ++i) if (x[i]==1) S =(S+b[i]) mod m;
   return S; }

   Decode(S) {
   S = u.S mod m; // (since $u = t^{-1}$ mod m)
   for (i=n-1; i$\leq$ 0; –i)
   if (S $\geq$ a[i]) { x[i]= 1; S = S-a[i];} else x[i]=0;
   if (S $\neq 0$) return ERROR-DECIPHERING else return (x[0, ..., n[);
   }
   Cost analysis : coding = $O(n^2)$ and decoding = $\tilde{O}(n^2)$.
6. *If SubSetSum(S, b_0, ..., b_{n-1}) modulo m is assumed intractable, then DH($b_0, \dots, b_{n-1}, m$) is provably secure.*
   On a general point of view, the crypto system is theoretically founded on the SUBSET-SUM problem which is NP-complete. But it exists bad choices of the b_i that makes it weak.

Exercise 2. Primes, big factor and factorization

1. *By definition, IS-COMPOSED(n)= not(IS-PRIME(n)). This gives the polynomial-time reduction : AlgoReduction-IS-COMPOSED(n) { r := Oracle-IS-PRIME(n); return (not r); } that proves IS-COMPOSED $\leq_P$ IS-PRIME.*
   Similarly, we obtain IS-PRIME $\leq_P$ IS-COMPOSED.
2. *IS-COMPOSED(n) = YES iff there exists a divisor d of n, with $1 < d < n$. Checking that d divides n is performed in time $\tilde{O}(\log n)$, thus polynomial in the input size. So, IS-COMPOSED $\in$ NP.*
3. *To what complexity class belongs IS-PRIME ? You have to give a proof of your answer. Since IS-PRIME = not (IS-COMPOSED) and IS-COMPOSED $\in$ NP, we clearly have IS-PRIME $\in$ co-NP.*
4. *The AKS algorithm for IS-PRIME rune in time $\tilde{O}(\log^6 n) \leq \log^{O(1)} n$ so polynomial in the input size $\log n$; yet PRIME $\in$ P. Besides, from the first question, IS-COMPOSED $\leq_P$ IS-PRIME; since P is closed by $\leq_P$, we have IS-COMPOSED $\in$ P.*
5. *Let $L(n) = (p_1, \dots, p_k)$ the list of prime factors of n such that $n = \prod i = 1^k p_i$ (the p_i may not be distinct). For any i : $2 \leq p_i \leq n$. Then $k \leq \log_2 n$ and the size of L(n) is $O(\log^2 n)$. Thus it can be checked in polynomial time $\log^{O(1)} n$ that $n = \prod i = 1^k p_i$ and that each of the k integers p_i is prime (using AKS algorithm for IS-PRIME). Now, L(n) is provided as an input certificate to check HAS-BIG-FACTOR(n, m) : once L(n) verified, we compute the maximum element p_j in L(n) : if $p_j \geq m$ (resp. $p_j < m$), then HAS-BIG-FACTOR(n, m)=YES (resp. NO) is proved in polynomial time $(\log n + \log m)^{O(1)}$; this states that HAS-BIG-FACTOR is in NP (resp. co-NP). So, finally, HAS-BIG-FACTOR $\in$ NP $\cap$ co-NP.*
6. *By binary search in $\{2, \dots, n\}$, $\log_2 n$ queries to Oracle HAS-BIG-FACTOR are sufficient to compute the largest prime factor p_1 of n. Then, this operation is performed recursively on n/p_1 until $n = 1$. Then at most $\log_2 n$ prime factors of n are computed, each computed in $\tilde{O}(\log n)$. Thus FACTORIZE*

$\leq_P$ HAS-BIG-FACTOR.
Conversely, obviously, HAS-BIG-FACTOR $\leq_P$ FACTORIZE (by computing the largest prime factor p_1 of n from a call to Oracle-FACTORIZE and returning YES iff $p_1 \geq m$).
So, HAS-BIG-FACTOR $=_P$ FACTORIZE and HAS-BIG-FACTOR $\in$ NP $\cap$ co-NP. If we have P= NP $\cap$ co-NP, then a consequence would be a polynomial time algorithm for FACTORIZE, thus a proof of the existence of a polynomial time algorithm for RSA.
Of course, this would have an impact on the provable security of RSA : however, the impact may be limited by the cost of the algorithm. A polynomial time algorithm with complexity t^{100} in the input size t would require for an instance of size larger than 10 more operations than the number of known particles in the known universe.

Exercise 3. RSA Provable security ; Factorization of n from d

1. *One knows that $\varphi(n)$ divides $ed-1$. There exists $k \in \mathbb{Z}$, such that $ed-1 = k\varphi(n)$. Besides : $t = \frac{ed-1}{2^s}$. Then, let us consider $a \in \mathbb{Z}$ coprime with n.*
One has $1 = a^{k\varphi(n)} = \left(a^t\right)^{2^s} \pmod{n}$. Hence, the order of a^t in $\mathbb{Z}_n$ is necessarily in $\{2^j\ ;\ 0 \leq j \leq s\}$. In other words :

$$\text{there exists } i \in [0,s]\ /\ \{\ a^{2^{i-1}t} \neq \pm 1 \pmod{n} a^{2^i t} = 1 \pmod{n}$$

Sets $u = a^{2^{i-1}t}$. Therefore, one obtains u such that

$$\{\ u^2 - 1 = (u-1)(u+1) = 0 \pmod{n}(u-1) \neq 0 \pmod{n}(u+1) \neq 0 \pmod{n}$$

Hence, $\gcd(u-1,n) = \gcd(a^{2^{i-1}t}-1,n)$ is a non trivial factor of n. One gets the second factor with a simple division.

```
Require: $(n, e, d)$.
Ensure: ($p$ and $q$ such that $n = pq$) or ERROR.
  Let $s$ and $t$ such that $ed - 1 = t2^s$ ($t$ odd)
  Let $a$ be a random number between $0$ and $n-1$.
  $\alpha \leftarrow a^t \pmod{n}$
  if $\alpha = 1$ or $\alpha = n-1$ then
    Return ERROR;
  end if
  for $i \leftarrow 1$ to $s$ do
    $tmp = \alpha^2 \pmod{n}$
    if $tmp = 1 \pmod{n}$ then
      Return $p = \gcd(\alpha, n)$ and $q = \frac{n}{p}$
    end if
    $\alpha \leftarrow tmp$
  end for
  Return ERROR;
```

2. *The average number of draws for a is 2 (the algorithm is immediate !) because if n is composite, at least half of the invertible elements does not satisfy the given relation. The proof of this statement can be divided into two parts. In the first part, one must show that there exists at least one value for a which satisfies these conditions. Then, one must prove that it is also the case for at least half of the invertible elements. Let us begin with the second part, which is easier.*
Sets $m = 2^i t$. Let us assume that there exists a such that $a^m = 1$ and $a^{m/2} \neq \pm 1$. Let $\{b_1, \ldots, b_k\}$ be the set of the b_i such that $b_i^{m/2} = \pm 1$ and $b_i^m = 1$. Then, necessarily, one has $(b_i a)^{m/2} \neq \pm 1$. Therefore, if there exists such value for a, there exists at least $\frac{\varphi(n)}{2}$ of them, hence one over two invertible element. Now, one only has to prove that there exists at least one of them. This is not the case for any composite number, but we shall see that this is true if n is the product of two distinct prime numbers. One recalls that t is odd. Thus, $(-1)^t = -1 \neq 1 \pmod{n}$. There exists one value for a (at worst, some power of -1) which satisfies $u = a^{m/2} \neq 1 \pmod{n}$ and $a^m = 1 \pmod{n}$. According to the Chinese Remainder theorem $u^2 = 1 \pmod{p}$ and $u^2 = 1 \pmod{q}$. Hence, u is congruent to 1 or -1 modulo p and q. It can not be equal to 1 modulo the two prime numbers because it would also be congruent to 1 modulo n. Therefore, one may assume that it is congruent to -1 modulo p. Then, if u is congruent

to 1 modulo q, one has found a valid element.
Otherwise, u is congruent to -1 modulo n. But, in this case, one is able to build another element which satisfies the relation since $m/2$ does not divide either $p-1$, nor $q-1$ (otherwise u would be congruent to 1 modulo p or q).
Indeed, one has $\frac{m}{2} = \frac{ed-1}{2^{s-i+1}} = \frac{k\varphi(n)}{2^{s-i+1}}$. Since $m/2$ does not divide $p-1$, one is able to set $\frac{m}{2} = v\frac{p-1}{2^j}$ with v odd and j between 0 and s. Then, one considers g, a primitive root modulo p, and one sets $\alpha = g^{2^j} \pmod p$ and $\beta = a \pmod q$. Then $\alpha^{m/2} = 1 \pmod p$ (by construction) and $\beta^{m/2} = a^{m/2} = -1 \pmod q$. One only has to apply the Chinese Remainder theorem in order to recover the element γ corresponding respectively to the remainders α and β modulo p and q. This element γ satisfies $\gamma^m = 1 \pmod n$ as well as $\gamma^{m/2} \neq \pm 1 \pmod n$. Since there exists at least one such element, at least of invertible element over two also satisfies the property and breaking RSA can be performed with 2 draws on the average.

Exercise 4. Discrete Log and highest significant bit *Not given. Home exercise.*

Exercise 5. El Gamal protocol. Let G a cyclic group of order $p-1$ generated by α. Let d be an integer, and $\beta = \alpha^d$. In the following asymmetric protocol (El Gamal), α and β are public while d is kept secret and known only by Bob.
The encoding function encodes x using a random integer k which is kept secret by the encoder. It is given by :

$$\begin{array}{lcl} E_{\alpha,\beta} : G & \rightarrow & G \times G \\ x & \mapsto & (\alpha^k, x.\beta^k) \end{array}$$

1. Bob receives a message (y_1, y_2). How will he decode it ?
2. Prove that a required condition for the protocol to be a one-way trap-door function is that the discrete logarithm is computationally impossible.

Not given. Home exercise.

Exercise 6. Asymmetric decryption is in NP Consider an asymmetric crypto-system : the public encryption method is denoted E ; the private decryption method is D. For any plain text p, the size of the corresponding cipher text $c = E(p)$ verifies : $|p| \leq |c| \leq |p| + O(1)$.
It is assumed that E is computed in deterministic polynomial (in practice almost linear) time.
Let "COMPUTE-D" be the following problem (note it is not a decision problem) :
– input : an arbitrary cipher text c ;
– output : the plain text $p = D(c)$.

1. Prove that "COMPUTE-D" can be solved in non-deterministic polynomial time.
2. Formulate decision problems related to "COMPUTE-D".
3. How would you preferably choose "D" (w.r.t. previous questions) ?

Not given. Home exercise.

Exercise 7. Non-deterministic algorithm and certification algorithm *Not given. Home exercise.*

Additional exercises. *Not given. Home exercise.*