Advanced Cryptography — Midterm Exam
Solution

Serge Vaudenay

17.4.2012

- duration: 3h00
- any document is allowed
- a pocket calculator is allowed
- communication devices are not allowed
- the exam invigilators will not answer any technical question during the exam
- the answers to each exercise must be provided on separate sheets
- readability and style of writing will be part of the grade
- do not forget to put your name on every sheet!

1 Circular RSA Encryption

Let $n = pq$ and $d = e^{-1} \bmod \varphi(n)$ define an RSA key pair. For some reason, we need to encrypt p with the plain RSA cryptosystem.

Q.1 If y decrypts to p, show that an adversary who has only the public key at disposal can decrypt y.
Hint: think modulo p.

If $y = p^e \bmod n$, then $y \bmod p = 0$ and $y \bmod q$ is in $\mathbf{Z}_q^*$ (since p and q are different prime numbers, p is coprime with q so p is invertible modulo q, so y as well). Hence, $\mathsf{gcd}(y,n) = p$ so the adversary recovers p easily.

2 The Goldwasser-Micali Cryptosystem

Consider the group $\mathbf{Z}_n^*$. We recall that if m is an odd factor of n, then the Jacobi symbol $x \mapsto \left(\frac{x}{m}\right)$ is a group homomorphism from $\mathbf{Z}_n^*$ to $\{-1,+1\}$. I.e., $\left(\frac{xy \bmod n}{m}\right) = \left(\frac{x}{m}\right)\left(\frac{y}{m}\right)$. It further has the property that $\left(\frac{x}{mm'}\right) = \left(\frac{x}{m}\right)\left(\frac{x}{m'}\right)$. We consider that multiplication in $\mathbf{Z}_n$ and the computation of the above Jacobi symbol can each be done in $O((\log n)^2)$.

Let s be a security parameter. We consider the following public-key cryptosystem.

Key Generation. Generate two different odd prime numbers p and q of bit size s, compute $n = pq$, and find some $z \in \mathbf{Z}_n^*$ such that $\left(\frac{z}{p}\right) = \left(\frac{z}{q}\right) = -1$. The public key is (n,z) and the secret key is p.

Encryption. To encrypt a bit $b \in \{0,1\}$, pick $r \in_U \mathbf{Z}_n^*$ and compute $c = r^2 z^b \bmod n$. The ciphertext is c.

Decryption. To decrypt c, compute $\left(\frac{c}{p}\right)$ and find b such that it equals $(-1)^b$. The plaintext is b.

This cryptosystem is known as the Goldwasser-Micali cryptosystem.

Q.1 Show that the cryptosystem is correct. I.e., if the key generation gives (n,z) and p, if b is any bit, if the encryption of b with the key (n,z) produces c, then the decryption of c with the key p produces b.

By construction, we have $n = pq$, $\left(\frac{z}{p}\right) = -1$, and $c \equiv r^2 z^b \pmod{n}$. We have $\left(\frac{c}{p}\right) = \left(\frac{r^2 z^b}{p}\right)$ since p divides n. Thus,

$$\left(\frac{c}{p}\right) = \left(\frac{r^2 z^b}{p}\right) = \left(\frac{z}{p}\right)^b = (-1)^b$$

So, the decryption of c produces b.

Q.2 Analyze the complexity of the three algorithms in terms of s.

Key generation: to generate the primes p and q of bit size s requires $O(s^4)$ by using Miller-Rabin primality testing, square-and-multiply exponentiation, and schoolbook multiplication. The Legendre symbol requires $O(s^2)$ which is negligible, as well as computing $n = pq$. So, key generation works in $O(s^4)$.
Encryption: this requires a constant number of multiplications which are $O(s^2)$.
Decryption: this requires a Legendre symbol, so $O(s^2)$ as well.

Q.3 Let $\mathcal{N}$ be the set of all n's which could be generated by the key generation algorithm. Let Fact be the problem in which an instance is specified by $n \in \mathcal{N}$ and the solution is the factoring of n.

Q.3a Define the key recovery problem KR related to the cryptosystem. For this, specify clearly what is its set of instances and what is the solution of a given instance.

In the KR *problem, an instance is a pair (n,z) such that $n \in \mathcal{N}$ and $\left(\frac{z}{p}\right) = \left(\frac{z}{q}\right) = -1$ where $n = pq$ is the factoring of n. The solution to the problem is p. Or, equivalently, q which plays a symmetric role.*

Q.3b Show that the KR problem is equivalent to the Fact problem. Give the actual Turing reduction in both directions.

Clearly, factoring n solves the problem: by submitting n to an oracle solving Fact*, we get p and q so we can yield p.*
Conversely, with an oracle solving the KR *problem, we can define an algorithm to factor n. For this, we just need to find one z satisfying $\left(\frac{z}{p}\right) = \left(\frac{z}{q}\right) = -1$ and feed (n,z) to the oracle solving* KR*. By construction, we have*

$$\left(\frac{z}{n}\right) = \left(\frac{z}{p}\right)\left(\frac{z}{q}\right) = 1$$

If we pick a random z satisfying $\left(\frac{z}{n}\right) = 1$, we have $\left(\frac{z}{p}\right) = \left(\frac{z}{q}\right)$ but this can be 1 or -1. If this is -1 (which happens with probability $\frac{1}{2}$), feeding (n,z) to the KR *oracle yield p. We can check that p solve the* Fact *problem and stop. If it is $+1$, it is bad luck as we have a bad z and we don't know. Thus, feeding (n,z) to the* KR *oracle may give anything. However, if it gives something which solves the* Fact *oracle, we are happy anyway and we can stop. Otherwise, we can start again with a new z. Eventually, we find a good z and the solution to* Fact.
So, KR *and* Fact *are equivalent.*

Q.4 Let QR be the problem in which an instance is specified by a pair (n,c) in which $n \in \mathcal{N}$ and $\left(\frac{c}{n}\right) = 1$. The problem is to decide whether or not c is a quadratic residue in $\mathbf{Z}_n^*$.

Q.4a Define the decryption problem DP related to the cryptosystem. For this, specify clearly what is its set of instances and what is the solution of a given instance.

In the DP *problem, an instance is defined by a triplet (n,z,c) where $n \in \mathcal{N}$ (let write $n = pq$), $z \in \mathbf{Z}_n^*$ is a non-quadratic residue with $\left(\frac{z}{n}\right) = 1$, and $c = r^2 z^b \bmod n$ for some $r \in \mathbf{Z}_n^*$ and a bit b. The problem is to find b.*

Q.4b Show that the DP problem is equivalent to the QR problem. Give the actual Turing reduction in both directions.

Clearly, with an oracle solving QR*, we can solve* DP*: we just submit (n,c) to the* QR *oracle and obtain b. Indeed, $r^2 z^b \bmod n$ is a quadratic residue if and only if $b = 0$.*
To show the converse, we assume an oracle O solving the DP *problem and construct an algorithm to solve the* QR *one. Given a* QR *instance (n,c), we pick $z \in \mathbf{Z}_n^*$ such that $\left(\frac{z}{n}\right) = 1$ and consider the function $f_z : y \mapsto O(n,z,y)$.*
If z is a quadratic residue, we observe that for any b, $r^2 z^b \bmod n$ is uniformly distributed in the set of quadratic residues modulo n. So, this is independent from b. Thus, $f_z(r^2 z^b \bmod n)$ is a random bit independent from b. If now z is a non-quadratic residue, $f_z(r^2 z^b \bmod n) = b$. By taking b uniformly distributed, we can easily identify in which case we are. We can thus iterate until we have a good z which is a non-quadratic residue. Then, we can compute $f_z(c)$ and get the solution to the QR *problem.*
So, DP *and* QR *are equivalent.*

3 Faulty Multiplier

Let B be a basis. Given some integers $x_0, \ldots, x_{n-1}$, we say that the sequence $[x_{n-1}, \ldots, x_0]$ represents x if

$$x = \sum_{i=0}^{n-1} x_i B^i$$

We say that $[x_{n-1}, \ldots, x_0]$ is a reduced sequence if $0 \le x_i \le B-1$ for all $i = 0, \ldots, n-1$. We say that a number x contains a block a if there exists n and a reduced sequence $[x_{n-1}, \ldots, x_0]$ representing x, and some i such that $a = x_i$. We consider the schoolbook algorithms for addition and multiplication. These are the methods that children learn at school for $B = 10$ and reduced sequences. We extend them to any B value.

We work with a microprocessor using a built-in 32×32-bit to 64-bit hardware multiplication. Each 32×32-bit to 64-bit multiplication is called an elementary multiplication. So, in the next we let $B = 2^{32}$. We assume that there is a bug such that the result is always correct except when the first operand is a special a_0 value and the second one is a special b_0 value in which case the result is a constant c_0 which is not equal to $a_0 b_0$.

Q.1 Let a,b,c,u,v be five 32-bit blocks. Let x be represented by $[a,b,c]$ and y be represented by $[u,v]$. Using the schoolbook multiplication algorithm in basis B to multiply x by y, give the list of elementary multiplications which are required to compute xy.

The schoolbook algorithm makes $u \times [a,b,c,0] + v \times [a,b,c]$. So, it performs av, bv, cv as in xv and also au, bu, cu as in xu. It obtains $[au,bu,cu,0] + [av,bv,cv] = [au, bu+av, cu+bv, cv]$. It then performs a reduction to obtain a reduced sequence representing xy.

Q.2 Let $w = \left\lceil \frac{\sqrt{b_0 B^3} - a_0}{B} \right\rceil$ and y be represented by $[w, a_0]$. Assume that $b_0 \leq \frac{B}{4} - 1$. Deduce that y contains the block a_0 and that y^2 contains the block b_0.
Hint: first show that

$$\sqrt{(b_0+1)B} - \sqrt{b_0 B} \geq 1$$

then show that

$$\frac{\sqrt{(b_0+1)B^3} - a_0}{B} > w \geq \frac{\sqrt{b_0 B^3} - a_0}{B}$$

and deduce that $\sqrt{(b_0+1)B^3} > y \geq \sqrt{b_0 B^3}$.

Since $[w, a_0]$ is a reduced sequence representing y, a_0 is trivially in y.
We have

$$\sqrt{(b_0+1)B} - \sqrt{b_0 B} = \frac{B}{\sqrt{(b_0+1)B} + \sqrt{b_0 B}}$$

If $b_0 \leq \frac{B}{4} - 1$, the denominator is upper bounded by B. So,

$$\sqrt{(b_0+1)B} - \sqrt{b_0 B} = \frac{\sqrt{(b_0+1)B^3} - a_0}{B} - \frac{\sqrt{b_0 B^3} - a_0}{B} \geq 1$$

Since w is the ceiling of $\frac{\sqrt{b_0 B^3} - a_0}{B}$, we obtain

$$\frac{\sqrt{(b_0+1)B^3} - a_0}{B} > w \geq \frac{\sqrt{b_0 B^3} - a_0}{B}$$

Now, $y = wB + a_0$. So, $b_0 B^3 \leq y^2 < (b_0+1)B^3$ from which we deduce that y^2 starts with the 32-bit block b_0. Clearly, y ends with the 32-bit block a_0. It is unlikely that b_0 appears in y, nor that a_0 appears in y^2.

In what follows, we assume that y does not contain the block b_0 and that y^2 does not contain the block a_0.

Q.3 Assume we want to raise y to some power k modulo n using the square-and-multiply with scanning of the bits of the exponent from left to right. The leading bit of the exponent k being 1, let b denote the second leading bit of k.

Q.3a Give the list of all multiplications this algorithm does when scanning these two bits in the two cases: i.e., for $b = 0$ and $b = 1$.

When scanning the first bit, it multiplies y by 1. The accumulator become equal to y. Then, it squares the accumulator and looks at the second bit. If it is 0, it does nothing more. Otherwise, it multiplies the accumulator by y. So, for $b = 0$, it computes $1 \times y$, y^2, and that's it. For $b = 1$, it computes $1 \times y$, y^2, and $y^2 \times y$.

Q.3b Show that for the y from Q.2, this algorithm is likely to compute $y^k \bmod n$ correctly when $b = 0$ whereas it does a computation error when $b = 1$.

In the $b = 1$ case, it multiplies y containing a_0 by y^2 containing b_0. Due to the schoolbook algorithm, this requires the bogus $a_0 b_0$ elementary operation so it makes an error. In the $b = 0$ case, it never needs to multiply y by y^2. So, it is unlikely that the bogus $a_0 b_0$ operation occurs.

Q.4 We assume a tamper-proof device implementing the RSA decryption with CRT acceleration, square-and-multiply with scanning of the bits of the exponent from left to right, and the school-book multiplication algorithm.

Q.4a Assuming that the second leading bits of $d \bmod (p-1)$ and $d \bmod (q-1)$ are different, using the y of Q.2, give an algorithm producing x such that $x^e \bmod n$ is equal to y modulo either p or q but not modulo both.

The CRT exponentiation computes $(y \bmod p)^{d \bmod (p-1)} \bmod p$ *and* $(y \bmod q)^{d \bmod (q-1)} \bmod q$. *Since* y *is small,* $y \bmod p = y \bmod q = y$. *So, it computes* $y^{d \bmod (p-1)} \bmod p$ *and* $y^{d \bmod (q-1)} \bmod q$. *If the second leading bits of* $d \bmod (p-1)$ *and* $d \bmod (q-1)$ *are different, one error will occur in exactly one of these operations. So, after CRT reconstruction, the result* x *will be equal to* y^d *modulo either* p *or* q *but not both. So,* $x^e \bmod n$ *will be equal to* y *modulo either* p *or* q *but not both.*

Q.4b Deduce a factoring attack on RSA using this device.

After getting x*, we compute* $\mathsf{gcd}(x^e - y \bmod n, n)$ *which is a non-trivial factor of* n*.*

4 Trapdoor Sbox

Let n be an integer. We consider the set $\mathbf{Z}_2^n$ as a vector space. Given a vector x, x_k denotes its k-th component (which is a bit). Additions are implicitly takes modulo 2. Product of bits are also implicitly taken modulo 2. The dot product $\alpha \cdot x$ between two vectors means $\sum_{k=1}^n \alpha_k x_k$. We also multiply a bit by a vector by multiplying the bit to each component.

Let $\alpha, \beta, \gamma \in \mathbf{Z}_2^n$. Let i and j be two fixed indices such that $\alpha_i = \beta_j = 1$ and $\gamma_j = 0$. Let w be the total number of bits set to 1 in γ. Let A be the subset of $\mathbf{Z}_2^n$ of all tuples in which the i-th component is zero. Let B be the subset of $\mathbf{Z}_2^n$ of all tuples in which the j-th component is zero. Let φ be a bijection from A to B.

Let p be a function from $\mathbf{Z}_2^n$ to A defined by $p(x)_k = x_k$ for all $k \neq i$ and $p(x)_i = 0$.

Let $v = (0, \ldots, 0, 1, 0, \ldots, 0) \in \mathbf{Z}_2^n$ be a constant vector, where $v_j = 1$.

We construct a function S on $\mathbf{Z}_2^n$ as follows.

$$S(x) = \varphi(p(x)) + \left((\alpha \cdot x) + (\beta \cdot \varphi(p(x))) + \prod_{k:\gamma_k=1} \varphi(p(x))_k \right) v$$

Q.1 Show that S is a permutation.
Hint: show that $S(x) = S(x')$ implies $p(x) = p(x')$ for any x and x' and show that $S(x+u) = S(x) + v$ for a constant vector u and any x.

Let q *be a function from* $\mathbf{Z}_2^n$ *to* B *defined by* $q(v)_k = v_k$ *for all* $k \neq j$ *and* $q(v)_j = 0$. *Since* q *is linear, since* $q(v) = 0$*, and since* $q(v) = v$ *for* $v \in B$*, we have* $q(S(x)) = \varphi(p(x))$. *So,* $S(x) = S(x')$ *implies* $\varphi(p(x)) = \varphi(p(x'))$. *Since* φ *is a bijection, this implies* $p(x) = p(x')$. *So, either* $x = x'$*, or* x *and* x' *only differ by their* i*-th bit.*
Let $u \in \mathbf{Z}_2^n$ *such that* $u_i = 1$ *and* $p(u)$ *is the null vector. Since* $p(x) = p(x+u)$*, we have* $S(x+u) = S(x) + v$. *So,* x *and* $x+u$ *do not have the same* S*-image. Finally,* $S(x) = S(x')$ *implies* $x = x'$. *That is,* S *is a permutation.*

Q.2 Compute $\mathsf{LP}_S(\alpha,\beta)$.
Hint: first give a simple expression of $(\alpha\cdot x)+(\beta\cdot S(x))$.

We have

$$\beta\cdot S(x)=\beta\cdot\varphi(p(x))+\left((\alpha\cdot x)+(\beta\cdot\varphi(p(x)))+\prod_{k:\gamma_k=1}\varphi(p(x))_k\right)\beta\cdot v$$

Since $\beta_j=1$ *and* v_j *is the only component of v set to 1, we have* $\beta\cdot v=1$. *So,*

$$\beta\cdot S(x)=\beta\cdot\varphi(p(x))+(\alpha\cdot x)+(\beta\cdot\varphi(p(x)))+\prod_{k:\gamma_k=1}\varphi(p(x))_k=(\alpha\cdot x)+\prod_{k:\gamma_k=1}\varphi(p(x))_k$$

Thus,

$$(\alpha\cdot x)+(\beta\cdot S(x))=\prod_{k:\gamma_k=1}\varphi(p(x))_k$$

Since $\varphi(p(x))$ *is uniformly distributed in B when x is uniformly distributed in* $\mathbf{Z}_2^n$, *and since* $\gamma_j=0$, *we have* $\Pr[(\alpha\cdot x)+(\beta\cdot S(x))]=2^{-w}$ *where w is the number of components of* γ *set to 1. Finally, we obtain*

$$\mathsf{LP}_S(\alpha,\beta)=(1-2^{1-w})^2$$

Q.3 Deduce a way to construct an Sbox with a given high $\mathsf{LP}_S(\alpha,\beta)$.

We select i, j such that $\alpha_i=\beta_j=1$. *Then, we pick* γ *such that* $\gamma_j=0$ *and with many components set to 1 (the more 1's, the larger* LP*). Then, we pick a permutation* φ *from A to B. The proposed construction for S is a permutation over* $\mathbf{Z}_2^n$ *which has a large* $\mathsf{LP}_S(\alpha,\beta)$.