Cryptography and Security — Midterm Exam
Solution

Serge Vaudenay

1.11.2023

- 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
- answers should not be written with a pencil

The exam grade follows a linear scale in which each question has the same weight.

1 Perfect Secrecy with Enigma

This exercise is about Enigma, and the possibility to obtain perfect secrecy when we limit the length of the plaintext. We assume that the Enigma key is uniformly distributed among the Enigma keyspace. We recall that a key is defined by four alphabet permutations $\alpha_0, \beta_0, \gamma_0, \sigma$ and an offset a. The permutations $\alpha_0, \beta_0, \gamma_0$ are elements of a choice of 5 permutations and must be different (they are the rotors). The permutation σ is an involution with 14 fixed points (this is the plugboard). The offset a is an integer such that $0 \leq a < 26^3$. To encrypt a bitstring $x = x_1 \cdots x_m$, we obtain $y = y_1 \cdots y_m$ with

$$y_i = \sigma^{-1} \circ \alpha_{i_1}^{-1} \circ \beta_{i_2}^{-1} \circ \gamma_{i_3}^{-1} \circ \pi \circ \gamma_{i_3} \circ \beta_{i_2} \circ \alpha_{i_1} \circ \sigma(x_i)$$

where π is a fixed involution with no fixed point (this is the reflector), $i+a = 26^2 i_3 + 26 i_2 + i_1$, with $0 \leq i_j < 26$, and $\alpha_i = \rho^i \circ \alpha_0 \circ \rho^{-i}$ (the first rotor in position i), the same for β_i and γ_i, and ρ is the circular rotation of the alphabet (of 26 letters).

Q.1 Recall what is the necessary constraint regarding the message space and the key space to obtain perfect secrecy.

Due to the Shannon theorem, the keypace must have at least as many elements as the support of the message space, i.e. the number of messages which can occur with nonzero probability.

Q.2 What is the minimal message length beyond which this constraint is not respected? HINT: we have seen in class that the Enigma keyspace has cardinality 2^k with $k \approx 57$.

The message space is based on an alphabet of 26 letters. If $26^\ell > 2^k$, there is no way to obtain perfect secrecy with arbitrary plaintext distributions. This gives as a limit

$$\ell = k \frac{\log 2}{\log 26} = k \times 0.213$$

For $k = 57$, this is $\ell = 12.1$. Beyond 12 characters, there is no perfect secrecy.

Q.3 Prove that when we press a key c, the lamp which shows on is never equal to c.

Given the formula to compute y_i in terms of x_i, there exists a permutation φ_i which depends on i such that

$$y_i = \varphi_i^{-1} \circ \pi \circ \varphi_i(x_i)$$

Actually, $\varphi_i = \gamma_{i_3} \circ \beta_{i_2} \circ \alpha_{i_1} \circ \sigma$. We know that π is an involution with no fixed point by design. If we had $y_i = x_i$, we would have $\pi(\varphi_i(x_i)) = \varphi_i(x_i)$ and $\varphi_i(x_i)$ would be a fixed point of π, which is not possible. Hence, we have $y_i \neq x_i$ for every i and x: whenever we press x_i, the lamp y_i is different from x_i.

Q.4 Prove that Enigma offers no perfect secrecy even when we limit the message space to one character and the plaintext can be any element of that space with a nonzero probability.

Let X and Y denote the plaintext and the ciphertext, respectively. We note that X and Y are in the same message space, due to the Enigma structure. Let c be such that $\Pr[Y = c] \neq 0$. We have $\Pr[X = c] \neq 0$ due to the assumption. However, we have $\Pr[X = Y = c] = 0$ due to the previous question. Hence, $\Pr[X = Y = c] \neq \Pr[X = c] \Pr[Y = c]$. Therefore, X and Y are not independent and we do not have perfect secrecy.

2 Diffie-Hellman Forever

We consider a group G (with multiplicative notations) generated by some g. The textbook Diffie-Hellman protocol is as follows:

- Alice picks a random x, computes $X = g^x$, and sends X to Bob.
- Bob picks a random y, computes $Y = g^y$, and sends Y to Alice.
- Alice computes $K = Y^x$, Bob computes $K = X^y$, and they both use it as a secret key.

Q.1 There are 4 important details missing in the textbook Diffie-Hellman protocol (4 for Alice and 4 for Bob). Spot at least 3.

First of all, the distribution to sample x or y is missing. It is uniform, in a set $\mathbf{Z}_q^*$, where q is the order of the group.
Second, they must verify that what they receive is not the neutral element to avoid the trivial man-in-the.middle attack making $K = 1$.
Third, they must verify that what they receive is a group element. This avoids attacks seen in class and other problems.
Finally, they must transform K using a KDF function before using it, because the distribution of a bitstring representing a group element may be weird.

Q.2 Let p and q be large prime numbers and g be an element of $\mathbf{Z}_p^*$ of order q. Hence, we have $G = \langle g \rangle$. Given a random element $z \in \mathbf{Z}_p^*$, how do we check efficiently if $z \in G$? Analyze the time complexity.

Since g has order q in $\mathbf{Z}_p^*$ of order $p-1$, we know that q divides $p-1$, due to the Lagrange theorem. We have seen in class that for any prime number q dividing $p-1$ there is a unique subgroup of $\mathbf{Z}_p^*$ of order q, and it is the group of the roots of the equation $z^q - 1 = 0$ in the $\mathbf{Z}_p$ field. So, we can check membership by computing $z^q \bmod p$ using the square-and-multiply algorithm and comparing the result to 1. The complexity is $\mathcal{O}(\ell^3)$ where ℓ is the bitlength of p.

Q.3 Let p, q, r be pairwise different large prime numbers and g be an element of $\mathbf{Z}_p^*$ of order $n = qr$. Hence, we have $G = \langle g \rangle$. Given a random element $z \in \mathbf{Z}_p^*$, how do we check efficiently if $z \in G$? Analyze the time complexity.
HINT: thanks to Bezout, we can write $z = (z^q)^u (z^r)^v$ and use the previous result.

It works like in the previous question: for any $z \in \mathbf{Z}_p^*$, we have $z \in G$ if and only if $z^n \bmod p = 1$. The $\Rightarrow$ sense is due to the Lagrange theorem. To prove the $\Leftarrow$ sense, we can proceed like in the proof from the course or use the following trick.
We assume that $z^n \bmod p = 1$. z^r satisfies $(z^r)^q = 1$ so is the unique subgroup of order q, which is generated by g^r. Hence, $z^r = g^{ar}$. Similarly, $z^q = g^{bq}$. We can use a Bezout relation $uq + vr = 1$ (due to q and r being coprime) and get $z = (z^q)^u (z^r)^v = g^{ubq+var}$ to show that z is generated by g.
The complexity is $\mathcal{O}(\ell^3)$ where ℓ is the bitlength of p.

Q.4 Is there any advantage of using a subgroup of $\mathbf{Z}_p^*$ of order qr? Discuss when q and r are known. Discuss when n is known but neither q nor r. Discuss when n is unknown. [This is an open question. Any answer with a detailed analysis is welcome.]

First of all, we have seen in the class that it is not recommended to use a group G of composite order. If we are open-minded, we can try to see the consequences of that. Normally, in a group of order q, we use a q of 256 bits and have exponentials with 256-bit exponents. We could wonder if there is an advantage.
If the discrete logarithm is easy in either $\langle g^q \rangle$ or $\langle g^r \rangle$, there is the special man-in-the-middle attack raising the public keys to the power either q or r then making the adversary able to compute K. So, we should not consider reducing the size of q and r compared to the required size if they were alone. Hence, both q and r should have at least 256 bits, making n of 512 bits.
If only n is known and hard to factor, one problem is that n should be really big, i.e. at least 2048 bits. There is no complexity advantage either.
If n is unknown, we have problems to sample x and y in $\mathbf{Z}_n$ so we would need x and y bigger than n to reduce the possible bias. In addition to this, n is a factor of $p-1$ and should not be too small. The ECM algorithm could recover factors less than 256 bits.
Maybe there could be an advantage for compatibility with other applications, but right now, except to make a midterm exercise, there is no real advantage.

3 Modulo 99 991

We let $n = 99\,991$ and we want to develop arithmetics modulo n with pen-and-paper.

Q.1 Given a decimal number x of 10 digits, develop an algorithm to reduce it modulo n by using only 2 subtractions and 3 additions of numbers up to 6-digit long. (We take multiplication by 10 as free.)
HINT: $10^5 \bmod 99\,991$.

We write $x = 10^5 a + b$ with a and b being two 5-digit numbers. We know that $10^5 \bmod n = 9$ so $x \equiv 9a + b \pmod{n}$. To multiply by 9, we just shift a with a 0 and we subtract a. That is a subtraction of a 6-digit number to a 5-digit number. We obtain a 6-digit number that we can add to the 5-digit number b. We obtain a new 6-digit number that we can write $10^5 a' + b'$ with a' having a single digit and b' being a 5-digit number. We can multiply a' by 9 to obtain a 2-digit number that we can add to b'. In rare cases, this will still be larger than n and we would have to subtract n. To subtract n, we can just add 9 and drop the carry which appears in the most significant digit.

1: *write $x = 10^5 a + b$ with a and b of length 5 digits*
2: *set $c = 10a$* ▷ *shift left and insert a right zero*
3: $d \leftarrow c - a$ ▷ *6-digit minus 5-digit → 6-digit*
4: $y \leftarrow b + d$ ▷ *5-digit plus 6-digit → 6-digit*
5: *write $y = 10^5 a' + b'$ with b' of 5 digits* ▷ *a' is a single digit*
6: *set $c' = 10a'$* ▷ *shift left and insert a right zero*
7: $d' \leftarrow c' - a'$ ▷ *2-digit minus 1-digit → 2-digit*
8: $z \leftarrow b' + d'$ ▷ *5-digit plus 2-digit → 5-digit*
9: ***if*** $z < n$ ***then return*** z
10: $w \leftarrow z + 9$ ▷ *5-digit plus 1-digit → 6-digit*
11: *drop the most significant bit of w to get v*
12: ***return*** v

Q.2 Factor $n - 1$ as a product of prime numbers.

We have $n - 1 = 2 \cdot 3^2 \cdot 5 \cdot 11 \cdot 101$.

Q.3 How would you verify that n is prime? Estimate the complexity.

We can run the Miller-Rabin test: pick a random $b \in \{1, \ldots, n-1\}$, raise it to the power $\frac{n-1}{2} = 49\,995$, and check that it is neither 1 nor -1 modulo n. In binary, 49 995 is 1100 0011 0100 1011. Hence, this means 15 squares and 7 multiplications using the square-and-multiply algorithm. Assuming we reduce modulo n after each square/multiplication, we obtain 22 square/multiplications of 5-digit numbers and 22 modulo n reductions of a 10-digit number for each b trial.
Note: the square-and-multiply algorithm which was seen in class uses 16 squares and 8 multiplications. However, the first square is squaring 1 and the first multiplication is multiplying by 1. So we can optimize the algorithm to save one quare and one multiplication.
Enumerating all prime numbers less than 316 (the floor of $\sqrt{n}$) and trying to divide n by them could also be a solution. There are only 65 prime numbers to try...

Q.4 How to check if x has a square root modulo n, and how to find one when it exists? Estimate the complexity.

If $x \bmod n = 0$, it has a square root and it is 0. Otherwise, x is in $\mathbf{Z}_n^*$ and we know that it has a square root if and only if $x^{\frac{n-1}{2}} \bmod n = 1$. That is because n is prime. To find one when it exists, since $n \bmod 4 = 3$, we can compute $x^{\frac{n+1}{4}} \bmod n$. We have $\frac{n+1}{4} = 24998$ which is $110\,0001\,1010\,0110$ in binary.
To check quadratic residuosity takes 22 square/multiplications and modulo operations. To compute the square root takes 21 square/multiplications and modulo operations. If we need to do both, we would rather directly apply the square root algorithm and verifies that it gives a square root in the end. This verification is only one multiplication and modulo operation.