Université Libre de Bruxelles
Faculté des Sciences

January 26, 2022

INFO-F-405: Introduction to cryptography

Principles

Question 1 (8%): Suppose that (Gen, E, D) is an IND-CPA secure symmetric-key encryption scheme with diversification, i.e., there is no known way of winning the IND-CPA game other than with a probability negligibly close to $\frac{1}{2}$ or with over-astronomical resources. Let the key space be $\{0,1\}^n$ with $n \geqslant 128$, the plaintext space to be arbitrary and the diversifier space be the set of non-negative integers $\mathbb{N}$. Would the following scheme E' be IND-CPA secure?

$$E'_k(d, m) = E_k(d, m) \parallel E_k(d + 2^{32}, m)$$

If it is IND-CPA secure, please justify briefly. If it is *not* IND-CPA secure, please specify how an adversary can win the IND-CPA game, and how much effort is necessary.

Correction :

Question 1 : The new scheme E' is not IND-CPA-secure anymore. For instance, an adversary can win the game with the following strategy :

1. The adversary chooses a random message m_0 and a random nonce d and queries the encryption $E'_k(d, m_0)$. The challenger answers with a ciphertext $q||q'$.

2. The adversary then submits the two messages m_0 and m_1 where and m_1 a random message such that $|m_0| = |m_1|$, along with the diversifier $d + 2^{32}$. The challenger answers with two ciphertexts $c_0||c'_0$ and $c_1||c'_1$.

3. The adversary checks whether or not q' equals c_0. If it is the case, it means that m_0 was the message encrypted. Otherwise, it's m_1.

Indeed, one can check that considering how E'_k is defined, we have the equalities

$$\begin{cases} E'_k(d, m_0) & = & E_k(d, m_0) \,||\, E_k(d + 2^{32}, m_0) & = & q||q' \\ E'_k(d + 2^{32}, m_0) & = & E_k(d + 2^{32}, m_0) \,||\, E_k(d + 2^{33}, m_0) & = & c_0||c'_0 \end{cases}$$

and therefore, if the message chosen by the challenger is m_0, we must have $q' = c_0$.
Note that the two diversifier d and $d + 2^{32}$ were used only once, as required.

Secret-key cryptography

A company is selling videogames. Next to the game on a physical media, they offer the users to register their copy online. Registered users can then enjoy playing over the network and benefit from other advantages. We assume that each copy of a videogame has a unique serial number S and comes with an authentication code C computed as a function of S and of K, a secret key known only to the company and used for all the games they sell. When registering, the user then has to give the pair (S, C) of his/her copy to the registration server.

The company built their authentication function on a block cipher $\mathcal{B}$ with the following signature :

$$\mathcal{B} : \{0,1\}^{128} \times \{0,1\}^{256} \to \{0,1\}^{128} : \mathcal{B}_K(X) \mapsto Y$$

where the X is the input block, Y is the output block and K is the secret key. They also chose a security parameter $\alpha \in [0, 128]$.

For each copy of the game, the company

- chooses a new serial number $S \in \{0,1\}^{128}$,
- computes the authentication code C by first applying the block cipher to S then keeping only the first α bits of the output, i.e., $C = \lfloor \mathcal{B}_K(S) \rfloor_\alpha$, and
- prints (S, C) on a sticker and attaches it to the videogame.

Question 2 (8%): How does the registration server check that the pair (S, C) is legitimate?

Question 3 (8%): A hacker would like to generate a valid pair (S, C) with a serial number different from that of his own copy. What is the name of this type of attack? If he makes random guesses, how many attempts do you expect him to submit before finding a valid one depending on the security parameter α?

Question 4 (8%): If we assume that the best known attack on $\mathcal{B}$ that retrieves the secret key is an exhaustive search, what is easier between finding a valid code by random-guessing and retrieving the secret key? Why?

In order to increase the security of the authentication, the company decides to use bigger parameters, more specifically, a serial number of 256 bits, $S \in \{0,1\}^{256}$. As the input is now larger than the block size of $\mathcal{B}$, the company defines a new mode of operation inspired from the *Electronic Code Book* (ECB) mode: C is obtained by

- applying the block cipher to S_1, the first 128 bits of S, keeping only the first α bits of the output,
- applying the block cipher to S_2, the last 128 bits of S, keeping only the first α bits of the output, and
- concatenating the results of these last two steps, i.e., $C = \lfloor \mathcal{B}_K(S_1) \rfloor_\alpha \parallel \lfloor \mathcal{B}_K(S_2) \rfloor_\alpha$.

Question 5 (8%): Instead of increasing the security, this variant completely breaks the system's security. Assuming he has seen one or more valid valid pairs (S_i, C_i), explain how a hacker could generate a valid (S, C) with a different serial number without knowing the secret key K. If the hacker has seen N valid pairs (S_i, C_i), how many new valid pairs can he create? (You can make assumptions on the way the serial numbers S_i are generated.)

> Correction :
>
> *Question 2 :* The registration server simply applies the block cipher $\mathcal{B}$ to S then truncate it to keep the first α first bits and checks if the output correspond to the value C sends by the user.
> In other words, it checks that the equality $C = \lfloor \mathcal{B}_K(S) \rfloor_\alpha$ is true.
>
> *Question 3 :* This kind of attack is called a forgery or a *random tag-guessing*. In order to find a valid pair (S, C) by random-guessing, the hacker can choose a random value S and then sends several pairs (S, C_i) to

the server until one is accepted. Since there are 2^α possibilities for C (and since there is exactly one possible corect value), the hacker would need 2^α attempts to be guaranteed to success. With about $2^{\alpha-1}$ attempt he would already have a reasonable chance to succeed.

Question 4 : The key is 256-bits long so an exhaustive search would require up to 2^{256} attempts to retrieve it. Meanwhile, we just saw that forge a valid pair (S, C) requires at most 2^α attempts where α is smaller than 128.
We deduce that forging a valid pair (S, C) is easier than finding the key.
However, finding the key would allow to forge as many valid codes as desired.

Question 5 : They are two ways for the hacker to generate new valid codes :

- Assuming he only knows one pair $(S, C) = (S||S', C||C')$, the hacker could submit the pair $(S'||S, C'||C)$ to the server. Indeed, because the variant is based on ECB, the two blocks forming S and C are checked independently so rearranging the order yields another valid pair.
  It is also possible to simply reuse twice one of the two blocks to get the valid pairs $(S||S, C||C)$ and $(S'||S', C'||C')$.

- Assuming he knows several pairs $(S_i, C_i) = (S_i||S'_i, C_i||C'_i)$, the hacker could swaps blocks between pairs to obtain new ones. For instance, the pairs $((S_1||S_2, C_1||C_2)$, $(S_2||S'_1, C_2||C'_1)$ or $(S'_1||S'_2, C'_1||C'_2)$ are all valid.

Now let us assume that the hacker has access to N different valid pairs (S_i, C_i). If N is small in front of the number of possible blocks $(= 2^{128})$, we can expect that every block (S_i, C_i) or (S'_i, C'_i) is unique. That would mean that with his N valid pairs, the hacker has access to $2N$ different valid blocks (S_i, C_i) or (S'_i, C'_i).
If we combine the two methods above, we deduce that in order to form a valid pair, we can pick any of our $2N$ block for the first block of our pair then pick any of the same $2N$ blocks for the second block of our pair. The total number is therefore $(2N)^2 = 4N^2$. If we substract the N pairs already available to the hacker we deduce that he can forge up to $4N^2 - N$ new valid pairs.

Hashing

A team is setting up a new blockchain. They are designing a new hash function H for it and considering to use the sponge construction.

Question 6 (8%): For their application, they estimate that preimage resistance is more important than collision resistance. They would like to have at least 160 bits of preimage resistance and at least 100 bits of collision resistance. What is the minimum number of bits that the hash function must output? Please justify! Also, if they use the sponge construction, what capacity can they choose, and what is the smallest permutation they can use?

We can view this blockchain abstractly as a set of transactions $\{T_i\}$. For a transaction T_i to be valid, it has to come with a solution x_i to the following a puzzle: The bit string value x_i must be chosen such that the digest $H(T_i\|x_i)$ starts with 0^k, i.e., the first k bits are all 0. (Here, we do not fix k as it will typically vary with the life of the blockchain.)

Question 7 (8%): Modeling H as a random oracle, what is the probability, for a fixed T and randomly-chosen x, that the output of $\mathcal{RO}(T\|x)$ starts with 0^k? What about after t attempts with different candidates x, approximately? As a function of k, how many attempts do we need before we can solve this puzzle, approximately? If we want the puzzle to take a time equivalent to about 2^{30} attempts, what is the corresponding value k?

Correction :

Question 6 : Let c and n be the capacity and the size of the output, respectively.
We first determine the minimal value for n :

- For some fixed output y, the probability that a random input x is hashed onto y is $\frac{1}{2^n}$ and so, an exhaustive search (which is the only possible attack if we assume H acts like a random oracle) would find a preimage for y after typically 2^n attempts. This means our hash function has a preimage resistance of n bits. In order to reach the 160 bits required, we need to take $n \geqslant 160$.
- Because of the birthday paradox, finding an (exernal) collision requires typically $\sqrt{2^n}$ attempts, hence providing $\frac{n}{2}$ bits of collision security. In order to reach the 100 bits required, we need to take $n \geqslant 200$.

Wrapping up those two constraints we deduce that n must be at least 200.

We apply the same reasoning for the capacity :

- A hash using the sponge construction provides $\frac{c}{2}$ bits of preimage resistance. In order to reach the 160 bits required, we need to take $c \geqslant 320$.
- It provides $\frac{c}{2}$ bits of (internal) collision resistance. In order to reach the 100 bits required, we need to take $c \geqslant 200$.

Wrapping us those two constraints we deduce that c must be at least 320.

The permutation f used is a function from $\{0,1\}^{c+r}$ to $\{0,1\}^{c+r}$ where r is the rate of the sponge function, *i.e.* the number of bits outputted after each application of f, during the squeezing phase.
We just saw that c is at least 320 bits and since r cannot be 0, we deduce that $c + r$ is at least 321, which means the smallest f we can use is a permutation of 321-bits bitstrings.

Question 7 :

- If we assume that H acts as a random oracle, the probability that the $H(T||x)$ starts with 0^k is $\frac{1}{2^k}$.
- After t attempts, the probability is $\frac{t}{2^k}$ if t is small in front of 2^k.
- In order to find a valid input x, we would need about 2^k attempts (more precisely, after 2^k attempts, the chances we find at least one valid input is about 63.2% when k goes to infinity).
- If we want to puzzle to take about 2^{30} to be solved, we would set $k = 30$.

Public-key cryptography

Consider the following variant of the ElGamal encryption scheme. Let p be a large prime and g is a generator of $\mathbb{Z}_p^*$. Let XOF be a secure extendable output function. Let $a \in \mathbb{Z}_{p-1}$ be Alice's private key and $A = g^a \bmod p$ her public key.

Encryption of $m \in \{0,1\}^*$ with Alice's public key A:

- Randomly choose a secret integer $k \in [1, p-2]$
- Compute

$$K = g^k \bmod p$$
$$T = A^k \bmod p$$
$$c = m \oplus \mathsf{XOF}(T)$$

- Send the ciphertext (K, c) to Alice

Decryption of (K, c) by Alice with her private key a:

$$T' = K^a \bmod p$$
$$m' = c \oplus \mathsf{XOF}(T')$$

Note that "$\oplus$" denotes the bitwise modulo-2 addition between two bit strings. We assume that $\mathsf{XOF}(\cdot)$ returns as many output bits as the length of the string it is added to.

Question 8 (6%): Please explain why the decryption procedure correctly recovers the plaintext m.

Question 9 (6%): Consider an adversary attempting a known plaintext attack. If it observes a known plaintext-ciphertext pair $(m, (K, c))$, can it recover the value T that was used during the encryption? Please justify.

Question 10 (8%): If for a given encryption, k is not kept secret, what are the consequences (if any) for the security of this ElGamal variant? Please justify.

Question 11 (8%): If for the encryption of two plaintexts m_1 and m_2 to Alice, the same value k is used, what are the consequences (if any) for the security of this ElGamal variant? Please justify.

Question 12 (8%): Bob wants to send many messages to Alice, but he is a bit lazy. So, instead of choosing an independent random k every time, he randomly chooses a secret integer k_0 once for all and increments it for every new encryption, i.e., he uses $k = k_0 + i \bmod (p-1)$ for the encryption of message number i. What are the consequences (if any) for the security of this ElGamal variant? Please justify.

Question 13 (8%): Let $\mathcal{E}$ be an elliptic curve over GF(p). Let $G \in \mathcal{E}$ be a generator of order q. Rewrite this encryption scheme to use the elliptic curve $\mathcal{E}$.

CORRECTION :
In this protocol,

- p,g, A and XOF are public
- a is know by Alice only
- k is picked randomly by Bob for each encryption

Question 8 :
Upon receiving the ciphertext (K, c), Alice computes $T' = K_a \bmod p$.
If the ciphertext is well-formed, then $K = g^k \bmod p$ and thus

$$T' = K^a = g^{ka} = g^{ak} = A^k \bmod p$$

so, under this assumption, we obtain the equality $T' = T$.

Alice then compute $m' = c \oplus \mathsf{XOF}(T')$.
Again, if the ciphertext if well-formed, $c = m \oplus \mathsf{XOF}(T)$ and thus

$$\begin{aligned} m' &= c \oplus \mathsf{XOF}(T') \\ &= m \oplus \mathsf{XOF}(T) \oplus \mathsf{XOF}(T') \\ &= m \oplus \mathsf{XOF}(T) \oplus \mathsf{XOF}(T) \\ &= m \end{aligned}$$

This proves that as long as the ciphertext (K, c) is well-formed, the decrpytion procedure retrieves the actual plaintext. For the following, we will assume the ciphertext will always be well-formed.

Question 9 :
We assume that an attacker (let us call her Eve from now on) knows a ciphertext (K, c) and the corresponding plaintext m.

She can compute $\mathsf{XOF}(T) = c \oplus m$ but since XOF is a secure extendable output function, one cannot compute its inverse to retrieve T.
She also knows that T is defined as $T = A^k \bmod p$. To retrieve T from this equality, she needs to solve the Diffie-Hellman Problem which is not feasible in polynomial time.
In the end, Eve cannot retrieve T knowing only K, c, m and the public parameters.

Question 10 :
If Eve intercepts a ciphertext (K, c) and knows the integer k used for the encryption, she can

1. compute $T = A^k \bmod p$,
2. compute $\mathsf{XOF}(T)$,
3. compute $m = c \oplus \mathsf{XOF}(T)$

and so, she can retrieve the plaintext m associated with the particular ciphertext (K, c). Note that Eve cannot decrpyt any other ciphertext and in particular, still ignore Alice's secret key a.

Question 11 :
Let two messages m_1 and m_2 be encrypted as (K_1, c_1) and (K_2, c_2) respectively and let us assume that both ciphertexts were obtained using the same nonce $k := k_1 = k_2$. In this setting, we can show that T_1 and T_2 are equal. We therefore write $T := T_1 = T_2$.
Knowing the ciphertexts, we can write

$$\begin{cases} c_1 &= m_1 \oplus \mathsf{XOF}(T) \\ c_2 &= m_2 \oplus \mathsf{XOF}(T) \end{cases}$$

`XOR`ing the two lines yields

$$\begin{aligned} c_1 \oplus c_2 &= (m_1 \oplus \mathsf{XOF}(T)) \oplus (m_2 \oplus \mathsf{XOF}(T)) \\ &= m_1 \oplus m_2 \end{aligned}$$

And thus Eve has access to the bitwise difference between the two plaintexts. Note that, again, Alice's secret key a is not leaked at all during te process.

Question 12 :
Let $k_i := k_0 + i$, where k_0 is chosen randomly by Bob before he encrypts anything and is kept secret bt him. From this, we can write

$$T_i = A^{k_i} = A^{k_0+i} = T_0 A^i \bmod p$$

where $T_0 := A^{k_0}$ is kept secret. Note that from Euler's theorem, we can reduce $k_0 + i$ modulo $p - 1$.

For each encryption, each T_i remains secret, even in a known-plaintext setting, as discussed in Question 9. The fact that the T_i's are connected together by a simple relation does not help, as this relation is broken by the XOF and not visible at the level of the keystream.

So, in short, this way of using this ElGamal variant remains secure.

However, one can imagine several other answers to this question, depending on the assumptions made about how the cryptosystem is used.

- First setting :
  We assume that the number of messages sent by Bob is greater than $p - 1$. If Eve manages to intercept two cipertexts (K_i, c_i) and (K_j, c_j) such that $i = j \bmod p - 1$, she can apply the attack describe in Question 11 to retrieve the bitwise difference $m_i \oplus m_j$ and thus the scheme is not completely secure anymore.
  In practice, this attack is not relevant since p is assumed t be a large prime number (at least 2^{128} and more realistically greater than 2^{2048}) and we can assume that Bob will never encrypt that many messages anyway.

- Second setting :
  In the encryption scheme, $\mathsf{XOF}(T_i)$ acts like a one-time pad generated by k_0, the latter is fixed and can therefore be interpreted as a secret key. If for any reason Eve gets to learn the value of k_0, T_0 or any tuple (k_i, i) or (T_i, i), she could deduce the value of $\mathsf{XOF}(T_j)$ for any j and then decrypt any ciphertext (K_j, c_j) as long as she knows the value of j.
  In this case, she can decrypt both past and future messages sent by Bob. Nevertheless, it doesn't reveal anything about Alice's secret key so if Bob notices the issue and changes how his k_i are generated, the new messages would be secure again.

Question 13 :

To convert the cryptosystem to its elliptic curve variant, all we have to do is to replace the group $\mathbb{Z}_p^*$ and it's modular addition by the group of points of an elliptic curve $\mathcal{E}$ and its point addition law.
In concrete terms, all we have to do is to replace every expression of the form $y = x^n \bmod p$ by $y = [n]x$.
In order to respect some conventions, integers are written in lowercase while the points of the curve are written in uppercase.

Let p be a large prime and G is a generator of $\mathcal{E}$. Let XOF be a secure extendable output function. Let $a \in [1, q-1]$ be Alice's private key and $A = [a]G$ her public key.

Encryption of $m \in \{0,1\}^*$ with Alice's public key A:

- Randomly choose a secret integer $k \in [1, q-1]$
- Compute

$$\begin{aligned} K &= [k]G \\ T &= [k]A \\ c &= m \oplus \mathsf{XOF}(T) \end{aligned}$$

- Send the ciphertext (K, c) to Alice

Decryption of (K, c) by Alice with her private key a:

$$\begin{aligned} T' &= [a]K \\ m' &= c \oplus \mathsf{XOF}(T') \end{aligned}$$

Portfolio

IND-CPA (chosen plaintext, chosen diversifier)

A scheme $\mathcal{E} = (\text{Gen}, \text{Enc}, \text{Dec})$ is **IND-CPA-secure** if no adversary can win the following game for more than a negligible advantage.

1. Challenger generates a key (pair) $k \leftarrow \text{Gen}()$
2. Adversary queries Enc_k with (d, m) of his choice
3. Adversary chooses d and two plaintexts $m_0, m_1 \in M$ with $|m_0| = |m_1|$
4. Challenger randomly chooses $b \leftarrow_R \{0, 1\}$, encrypts m_b and sends $c = \text{Enc}_k(d, m_b)$ to the adversary
5. Adversary queries Enc_k with (d, m) of his choice
6. Adversary guesses b' which plaintext was encrypted
7. Adversary wins if $b' = b$ (Advantage: $\epsilon = \left|\Pr[\text{win}] - \frac{1}{2}\right|$.)

The adversary **must** respect that d **is a nonce**! The values of d used in steps 2, 3 and 5 must all be different.

Sponge construction

The sponge construction is a mode on top of a permutation. It calls a b-bit permutation f, with $b = r + c$, i.e., r bits of *rate* and c bits of *capacity*. A sponge function is a concrete instance with a given f, r, c and implements a mapping from $M \in \{0, 1\}^*$ to $\{0, 1\}^\infty$ (truncated at an arbitrary length):

- $s \leftarrow 0^b$
- $M \| 10^*1$ is cut into r-bit blocks
- For each M_i do (absorbing phase)
  - $s \leftarrow s \oplus (M_i \| 0^c)$
  - $s \leftarrow f(s)$
- As long as output is needed do (squeezing phase)
  - Output the first r bits of s
  - $s \leftarrow f(s)$

The differentiating advantage of a random sponge from a random oracle is at most $t^2/2^{c+1}$, with t the time complexity in number of calls to f.

Random oracle

A random oracle is a non-deterministic algorithm that returns uniformly and independently distributed random bits for any input value. The same output bits are returned when the same value is input.

Original ElGamal encryption

Encryption of $m \in \mathbb{Z}_p^*$ with Alice's public key A:

- Randomly choose a secret integer $k \in [1, p-2]$
- Compute

$$K = g^k \bmod p$$
$$c = mA^k \bmod p$$

- Send the ciphertext (K, c) to Alice

Decryption of (K, c) by Alice with her private key a:

$$m = K^{-a} c \bmod p$$