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:

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

Université Libre de Bruxelles
Faculté des Sciences

January 25, 2023

INFO-F-405: Introduction to cryptography

This assessment may be done with the use of written or printed personal notes, slides, books, etc., but without any electronic devices (e.g., laptop, tablet, phone) and without internet access. Also, the assessment is strictly personal and you are not allowed to communicate with other students or other people during this assessment.

This exam is made of 11 questions and includes a portfolio at the end. Please ensure that we can split your answers to questions 1-7 from those of questions 8-11 by using separate sheets of paper.

Principles

■ **Question 1** (10%): With your own words, please explain what are confidentiality and authenticity in cryptography. Then, does an encryption scheme provide authenticity and why (not)? And finally, does an authentication scheme provide confidentiality and why (not)?

> Those security notions admit several variants depending of the actual use case they are needed for. As high-level and intuitive definitions, one could say that :
>
> - A scheme E_k using a secret k to transmit a ciphertext $c := E_k(m)$ achieves confidentiality if it is impossible to retrieve the message m from c without knowing the secret k.
> - A scheme S_k using a secret k to transmit a message m achieves authenticity if it is impossible to create a forgery, i.e., a valid pair $(m, s) := S_k(m)$ without knowing the secret k. Authenticity includes integrity in the sense that aftering observing a legitimate valid pair (m, s), it must be impossible to create a different valid pair $(m', s') \neq (m, s)$ without knowing the secret k.
>
> In general, an encryption scheme provides no authenticity. It ensures confidentiality to its users but does not prevent malicious attackers from impersonating them through a *meet-in-the-midle* attack, for instance. As another example, with a stream cipher, an attacker can flip bits of the ciphertext to flip the corresponding bits of the plaintext, even if not knowing their value in the plaintext.
>
> In general, an authentication scheme does not provide confidentiality as it does not transform the message but instead adds a tag (MAC or signature) to it. Usually, the original message is transmitted along with the tag.

■ **Question 2** (12%): 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 $n \geq 128$, the plaintext space to be arbitrary and the diversifier space be the set of non-negative integers $\mathbb{N}$. Let H be a cryptographic hash function. Would the following schemes $E^{(1)}$, $E^{(2)}$ and $E^{(3)}$ be IND-CPA secure?

$$E_k^{(1)}(d, m) = E_k(H(d), m)$$
$$E_k^{(2)}(d, m) = E_k(d, m) \,\|\, H(m)$$
$$E_k^{(3)}(d, m) = E_k(H(k\|m), m) \,\|\, H(k\|m)$$

For each scheme, if it is IND-CPA secure, please justify briefly, and specify if there are properties that H needs to satisfy for this. Otherwise, please specify how an adversary can win the IND-CPA game, and how much effort is necessary.

- $E_k^{(1)}$ is still IND-CPA-secure. Indeed, H is assumed to be a cryptographic hash function and thus, it should be collision resistant. This implies that if d is indeed a nonce, $H(d)$ also acts as a nonce and doesn't lead to any vulnerability.
  Note that H must be collision resistant, otherwise the adversary could find $d_1 \neq d_2$ such that $H(d_1) = H(d_2)$, and then win the IND-CPA game by reusing the same diversifier in E_k while respecting the diversifier uniqueness at the level of $E_k^{(1)}$.
- $E_k^{(2)}$ is not IND-CPA-secure. Appending the hash of the message gives us a way to win the IND-CPA game. The strategy is the following :
  1. The adversary randomly chooses two messages m_0 and m_1 and a diversifier d and transmits (m_0, m_1, d) to the challenger.
  2. The challenger responds with $c = E_k^{(2)}(d, m_b) = E_k(d, m_b)||H(M_b)$, with $b \in \{0, 1\}$. Because the specification of E and H are publicly known, the adversary can extract the last bits of c to obtain $H(m_b)$.
  3. The adversary computes $H(m_0)$ and compares it with $H(m_b)$. If there are the same, they outputs $b' = 0$. Otherwise they outputs $b' = 1$.

  Because H is assumed secured, the equality $H(m_b) = H(m_0)$ implies $m_b = m_0$ with overwhelming probably and so we are almost certain to win the game after the first attempt. Note that no query is required, here.
- $E_k^{(3)}$ cannot be IND-CPA-secure because the diversifier has no effect. In order to win the IND-CPA game, we can apply one strategy that works for every scheme without diversification :
  1. The adversary randomly chooses a message m' and a diversifier d' and sends (m', d') as a query. The challenger answers with the ciphertext c'.
  2. The adversary chooses two messages : m_0 equals m' and m_1 is random. They then chooses a diversifier $d \neq d'$ and transmits (m_0, m_1, d) to the challenger.
  3. The challenger responds with a ciphertext $c = E_k(H(k|m_b), m_b) \mid H(k|m_b)$.
  4. The adversary checks whether or not the equality $c' = c$ holds. If it does, they output $b' = 0$. Otherwise they output $b = 1$.

  This strategy respects the rules of the game : the adversary never uses the same diversifier twice. Yet, because the encryption only depends on the message and the key (which doesn't change), the equality $E_k^{(3)}(d, m_b) = E_k^{(3)}(d', m')$ implies (with overwhelming probability) $m_b = m'$. And again, we expect to win the game after the first attempt, using a single query.

■ **Question 3** (12%): A hypothetical encryption scheme, called RC404, has been standardized with only the following security claim: *"RC404 resists against key recovery attacks with a security strength of 128 bits in the known plaintext model."*

For each of the following attacks, please state whether it contradicts the claim and why (not).

a. A method that recovers the key with a success probability of one in a billionth ($\approx 2^{-30}$), requires 2^{100} known plaintext-ciphertext pairs and runs in about 2^{100} times the execution time of RC404.

b. A method that recovers with certainty the first byte of the plaintext (although not the next ones) from the ciphertext only, with a running time of a few hours on a modern PC.

c. A method that recovers the key with certainty after seeing the ciphertexts corresponding to the $10! \approx 2^{22}$ plaintexts composed of the 10 letters "ALGORITHMS" in any order, with a running time of a few hours on a modern PC.

What criticism can you make on RC404 and on its security claim?

Let us start with reading the security claim and identify what is *left unclaimed*:

- other goals than key recovery,
- any attack with higher security strength level than 128 bits, i.e., with running time t, data complexity d and success probability ϵ such that $\log_2(t+d) - \log_2(\epsilon) > 128$, and
- other data models than the known plaintext model.

a. First note that $\log_2(t+d) - \log_2(\epsilon) > 128$ is equivalent to $\frac{t+d}{\epsilon} > 2^{128}$. In this first attack, $t = d = 2^{100}$ and $\epsilon = 2^{-30}$, so $\frac{2^{101}}{2^{-30}} = 2^{131}$, which is greater than 2^{128}. Hence, this attack *does not contradict* the claim as it is above the 128-bit security strength level.

b. The second attack does not recover the key but only some information about the plaintext. While this attack shows that RC404 does not ensure confidentiality properly, it *does not contradict* the claim as only the key recovery goal is claimed.

c. The third attack requires a chosen plaintext data model, as it needs a specific set of plaintext inputs to work. So, it *does not contradict* the claim as only security in the known plaintext model is claimed.

Clearly, this shows that RC404 would be a very bad encryption scheme. First, it does not provide proper confidentiality as the first byte of plaintext can be fairly easily recovered. Second, even the key can be fairly easily recovered, thereby destroying all confidentiality, if the attacker can get access to it for the encryption of a few million specific plaintexts.

The security claim leaves much to be desired as it does not claim security for all goals and all data models. A better claim would be to say that the scheme is indistinguishable from an ideal scheme in the chosen plaintext model (IND-CPA) or in the chosen plaintext and ciphertext model (IND-CCA). The 128-bit security strength level, however, is sufficient for many years to come as it would require from the attacker astronomical resources to go above 128 bits of security.

Secret-key cryptography

Rijndael / AES

■ **Question 4** (6%): What is a block cipher? How do you use it in an encryption or authentication scheme, in general? Even if the best attack against a block cipher is the exhaustive key search, is it enough to guarantee the security of an encryption/authentification scheme that uses this block cipher? Why (not)?

For a block size n and a secret key size m, a block cipher is a mapping $E : \mathbb{Z}_2^n \times \mathbb{Z}_2^m \to \mathbb{Z}_2^n$

- from a secret key $k \in \mathbb{Z}_2^m$ and an input block $x \in \mathbb{Z}_2^n$
- to an output block $y = E_k(x) \in \mathbb{Z}_2^n$.

For each key k, it has an inverse: $x = E_k^{-1}(y)$.

To make an encryption or authentication scheme from a block cipher, it is necessary to specify a *mode of operation* such as CBC, CTR, CBC-MAC, etc.

Even if the best attack against a block cipher is exhaustive key search, an encryption or authentication scheme using it can be broken if the mode of use is bad. For instance, the ECB mode leads to an unsecure encryption scheme despite the quality of the block cipher. One could even imagine pathological modes of operation that would output the secret key as a part of the ciphertext.

■ **Question 5** (10%): Consider two executions of the block cipher AES-128 with the same secret key. The first one has input block X and the second input block X'. If X and X' differ only in *two bytes*, at least how many bytes will differ after two rounds? And at most? Note that the position of the two bytes is not specified, and so you may specify it to be able to reach the minimum or the maximum.

> We must track the bytes that differ between the two executions through the steps of two rounds, namely, SubBytes, ShiftRows, MixColumns, AddRoundKey, SubBytes, ShiftRows, MixColumns, AddRoundKey. First, we remark that SubBytes and AddRoundKey do not modify the location nor the number of bytes that differ, so we can forget them in our reasoning. Second, as ShiftRows only changes the position of the bytes, the initial location of the two differing bytes can be equivalently specified before or after the first ShiftRows. For simplicity, we can specify them after the first ShiftRows. As a result, we can simplify our reasoning and track the differing bytes through the sequence MixColumns, ShiftRows, MixColumns.
>
> After two rounds, the minimum number of differing bytes is 12 and the maximum is 16. Let us see why.
>
> - For the minimum, we consider two cases based on the initial location of the two differing bytes.
>   - If they are in the same column, the MDS property implies that there are at least 3 differing bytes in the column after MixColumns. After ShiftRows, these three bytes are moved to three different columns. Then, MixColumns expands each differing byte in a column to 4 differing bytes, because of the MDS property. As a result, there are 12 differing bytes after two rounds (see case 1 in Figure 1).
>   - If they are in different columns, the MDS property implies that there are 4 differing bytes in these two columns after MixColumns. After ShiftRows, these eight bytes are moved to different columns with two bytes in each. Then, MixColumns expands each two differing bytes in a column to at least 3 differing bytes, because of the MDS property. As a result, there are 12 differing bytes after two rounds (see case 2 in Figure 1).
> - For the maximum, we only need to show one case where 16 bytes can differ. Let us arbitrarily put the two differing bytes in the same column. Then, after the first MixColumns, there can be 4 differing bytes in the same column. These bytes are then moved to different columns after ShiftRows. Finally, MixColumns expands each differing byte in a column to 4 differing bytes, because of the MDS property. As a result, there are 16 differing bytes after two rounds (see case 3 in Figure 1). (Note that we can also reach the maximum of 16 by letting the initial differing bytes be in different columns.)

Hashing

SHA-512/256 is a hash function standardized by the NIST. In a nutshell, it works like SHA-512, except that the output is truncated to 256 bits. In more details, it uses the Merkle-Damgård construction on top of SHA-512's compression function. Hence, it has a chaining value of 512 bits and a message block size of 1024 bits. The output consists of the first 256 bits of the final chaining value. At the time of this writing, there has been no weakness found in its compression function.

■ **Question 6** (10%): In the light of these specifications of SHA-512/256,

a. what is its preimage resistance?

b. what is its second preimage resistance?

c. what is its collision resistance?

d. is producing an internal collision a good strategy to find a collision in this hash function, and why (not)?

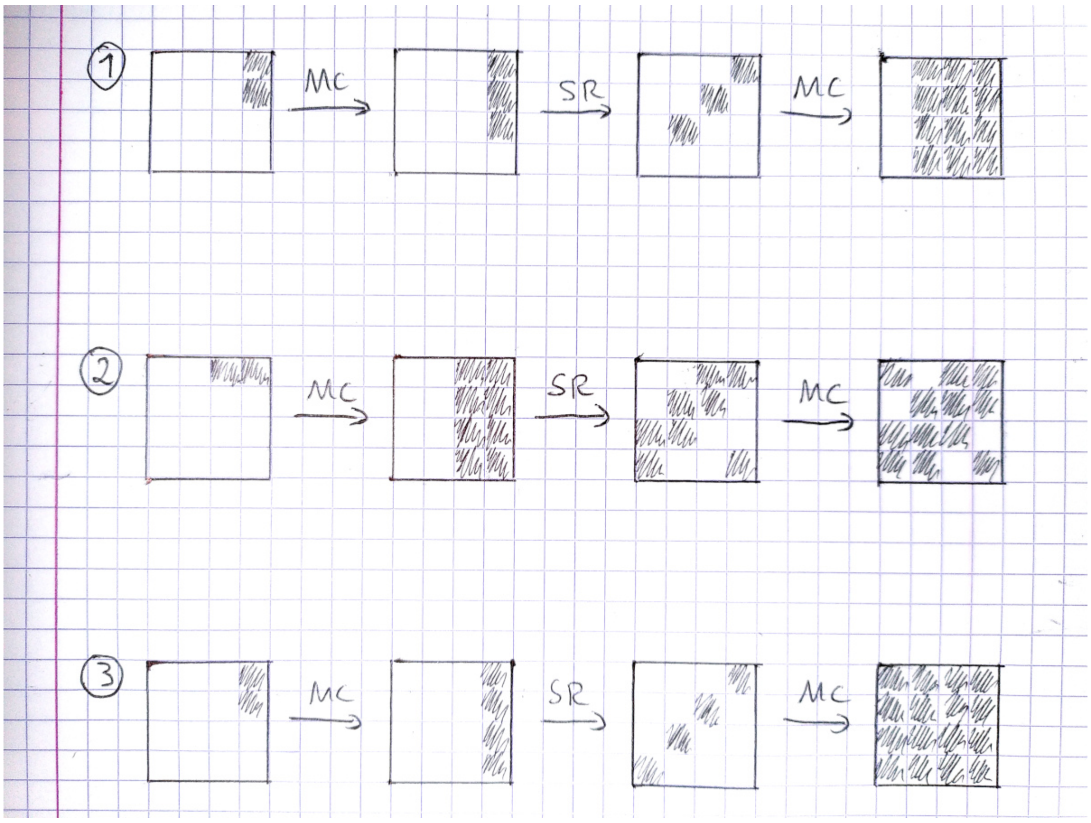


Figure 1

The hash function uses a strong compression function. Also, it has a large enough chaining value, i.e., with 512 bits of chaining value, an internal collision would require 2^{256} attempts. So, in general, it should be suitable for 256-bit security.

a. The preimage resistance is 256 bits as the output size is 256 bits.
b. The second preimage resistance is 256 bits as the output size is 256 bits.
c. Because of the birthday paradox, the collision resistance is 128 bits as the output size is 256 bits.
d. Producing an internal collision is not a good strategy to find a collision in this hash function since the chaining value is longer than the output. So, it would take more effort to search for an internal collision than for an output collision.

■ **Question 7** (10%): For authenticating transactions, we would like to use a message authentication code $\text{MAC}_K(\text{message})$, but we hesitate between different options. Here K is a 128-bit key and the message can be of any size. For each option, please state whether it is secure and briefly justify your answer.

a. $\text{MAC}_K(\text{message}) = \text{SHA-512}(K\|\text{message})$
b. $\text{MAC}_K(\text{message}) = \text{SHA-512/256}(K\|\text{message})$
c. $\text{MAC}_K(\text{message}) = \text{HMAC-SHA-512}_K(\text{message})$
d. $\text{MAC}_K(\text{message}) = \text{HMAC-SHA-512/256}_K(\text{message})$

Which one is the most efficient and secure option in your opinion?

The key point to remember here is length extension attacks (LEAs)!

a. This first option is *not secure* as it is susceptible to LEAs.
b. This second option looks similar to the first one, but it is no longer susceptible to LEAs. SHA-512/256 only outputs the first 256 bits of the chaining value. To apply the length extension attack, the adversary would need to guess the remaining 256 bits, which is more costly than guessing the 128-bit key. So, this option is *secure*.
c. This third option is *secure*, as HMAC is specifically designed to thwart LEAs.
d. The same conclusion goes for this last option, even if the chaining value is shorter (but sufficiently long).

HMAC implies an extra call to the compression function, hence the third and fourth options are slightly less efficient than the second one. So, in our opinion, the second option is the most efficient one while still being sufficiently secure.

Public-key cryptography

A tripartite key-exchange

Anya, Bert and Chloe want to communicate using AES but to do this, they first need to agree upon a common secret that they could use to derive the AES secret key. They want to use the Diffie-Hellman key exchange but this protocol allows to share a secret between two parties only. The goal of this exercise is to create a variant of Diffie-Hellman in which three people can obtain the same secret.

Here, we fix a prime p and a generator g of the group $\mathbb{Z}_p^*$. Anya's, Bert's and Chloe's private/public key pairs (SK, PK) are $(a, A = g^a)$, $(b, B = g^b)$, $(c, C = g^c)$ with a, b, c in $[1, \ldots, p-2]$. The reduction modulo p is implicit as we work in $\mathbb{Z}_p^*$.

■ **Question 8** (8%): Propose a protocol based on Diffie-Hellman that would allow all three people to share the common secret g^{abc}. *Hint:* The parties can first work in pairs. What is the cost of your protocol compared to the original Diffie-Hellman protocol ?

One simple solution is the key exchange described by the following diagram. The variables in parenthesis are the data available by each participant.

$$\begin{tikzcd}
 & \begin{array}{c}\text{Anya}\\(a,A,B,C)\end{array} \arrow[dl, "C^a"'] & \\
\begin{array}{c}\text{Bert}\\(b,A,B,C)\end{array} \arrow[rr, "A^b"'] & & \begin{array}{c}\text{Chloe}\\(c,A,B,C)\end{array} \arrow[ul, "B^c"']
\end{tikzcd}$$

The step-by-step protocol is as follow :

1. - Anya computes C^a and sends it to Bert.
   - Bert computes A^b and sends it to Chloe.
   - Chloe computes B^c and sends it to Anya.

2. - Anya computes $(B^c)^a = g^{abc}$.
   - Bert computes $(C^a)^b = g^{abc}$.
   - Chloe computes $(A^b)^c = g^{abc}$.

Note that everyone's computations involve only data available to them. In this scheme, 3 modular exponentiations are computed in the first round and 3 others in the second round, for a total of 6 exponentiations. Considering a usual Diffie-Hellman protocol requires a total of 2 exponentiations [1] , the computational cost of our proposed protocol is threefold.

Variants of the proposed scheme are also possible. For instance, we could do the following :

1. - Anya computes C^a and sent it to Bert.
   - Anya computes B^a and sent it to Chloe.
   - Bert computes C^b and sent it to Anya (or Chloe computes B^c and send it to Anya)

2. - Anya computes $(C^b)^a = g^{abc}$ (or $(B^c)^a$ if Chloe sent her B^c)
   - Bert computes $(C^a)^b = g^{abc}$.
   - Chloe computes $(B^a)^c = g^{abc}$.

We can represent this by the following diagram

$$\begin{tikzcd}
 & \begin{array}{c}\text{Anya}\\(a,A,B,C)\end{array} \arrow[dl, shift right, "C^a"'] \arrow[dr, "B^a"] & \\
\begin{array}{c}\text{Bert}\\(b,A,B,C)\end{array} \arrow[ur, shift right, "C^b"'] & & \begin{array}{c}\text{Chloe}\\(c,A,B,C)\end{array}
\end{tikzcd}$$

This scheme allows everyone to obtain the shared secret with a total of 6 exponiations just like the previous solution but it is not as elegant as the first one as it introduces an asymmetry [2] between the three parties.

Generic group notation

Let G be a group of order $|G| = q$ and g is a generator of this group.

■ **Question 9** (6%): Rewrite the two-party Diffie-Hellman algorithm in the generic group notation so that it can be applied, e.g., to elliptic curves. What is the expression of the common secret with the generic group notation?

With the generic group notations, the respective key pairs of Anya and Bert are $(a, A = [a]g)$ and $(b, B = [b]g)$ with a generator g of G and $a, b \in [1, q-1]$.
The usual Diffie-Hellman key exchange then consists of the following steps :

1. - Anya sends A to Bert
   - Bert sends B to Anya

2. - Anya computes the shared secret $[a]B = [a][b]g = [ab]g$
   - Bert computes the shared secret $[b]A = [b][a]g = [ab]g$

A one-round tripartite Diffie-Hellman using a pairing

We now want to improve our protocol and improve its efficiency. In order to do this, we will use a special kind of function (often used in elliptic curve cryptography) called a *pairing*.

Let E be an elliptic curve, G the group of points on this curve and P a point that is a generator of the group G. Let $q = |G|$ be the order of G, i.e., the number of elements in G. Anya's, Bert's and Chloe's private/public key pairs $(\mathsf{SK}, \mathsf{PK})$ are now $(a, A = [a]P), (b, B = [b]P), (c, C = [c]P)$ where a, b, c are integers in $\mathbb{Z}_q$.

Finally, we define a map $F : \mathbb{Z}_q \times G \times G \to \mathbb{Z}_q^*$ with the following properties. If P is a generator of the group G, then for any points R and Q and any integer x in $\mathbb{Z}_q$, we have

- $F(1, P, P) = u,$ where u is some fixed and known element in $\mathbb{Z}_q^*$,
- $F(x, R, Q) = F(1, R, Q)^x$,
- $F(1, [y]R, Q) = F(1, R, Q)^y$,
- $F(1, R, [z]Q) = F(1, R, Q)^z$.

Such a map F is what we call a *pairing*. (*Note:* you can safely assume the pairing F to be already defined and use it abstractly.)

In essence, these properties allow us to compute $F(x, Y, Z)$ for any point $Y = [y]P$ and $Z = [z]P$, and the result can be expressed as a power of the known integer u. Note that every point can be expressed as a multiple of P since we assume P is a generator.

■ **Question 10** (6%): Compute the value of $F(1, A, B)$ and $F(a, P, P)$.

We simply apply the formulas :

1.

$$\begin{aligned}
F(1,A,B) &= F(1,[a]P,[b]P) && \text{(by definition of } A \text{ and } B\text{)}\\
&= F(1,P,[b]P)^a && \text{(using the second property of } F\\
&= F(1,P,P)^{ab} && \text{(using the third property of } F\text{)}\\
&= u^{ab} && \text{(by definition of } u\text{)}
\end{aligned}$$

2.

$$\begin{aligned}
F(a,P,P) &= F(1,P,P)^a && \text{(using the first property of } F\text{)}\\
&= u^a && \text{(by definition of } u\text{)}
\end{aligned}$$

■ **Question 11** (10%): Propose a protocol in which each party obtain a common secret by applying the pairing F only once. What is the expression of this common secret?

If Anya, Bert and Chloe have respective key pairs $(a, A = [a]P)$, $(b, B = [b]P)$ and $(c, C = [c]P)$, the tripartite key exchange simply consist of one step :

- Anya computes $F(a,B,C) = F(a,P,P)^{bc} = F(1,P,P)^{abc} = u^{abc}$
- Bert computes $F(b,A,C) = F(b,P,P)^{ac} = F(1,P,P)^{abc} = u^{abc}$
- Chloe computes $F(c,A,B) = F(c,P,P)^{ab} = F(1,P,P)^{abc} = u^{abc}$

Then can therefore use u^{abc} as the shared secret. This solution has several advantages compared to the one proposed for question 8 : it only requires one round of computation instead of two and more importantly, it is *non-interactive*. As long as everyone's public key is made public (which is only done once), any group of three people can compute such a shared secret without having to send and receive intermediate messages from the others.
This idea was presented in 2000 by Antoine Joux, in the paper *A One Round Protocol for Tripartite Diffie–Hellman*.

Portfolio

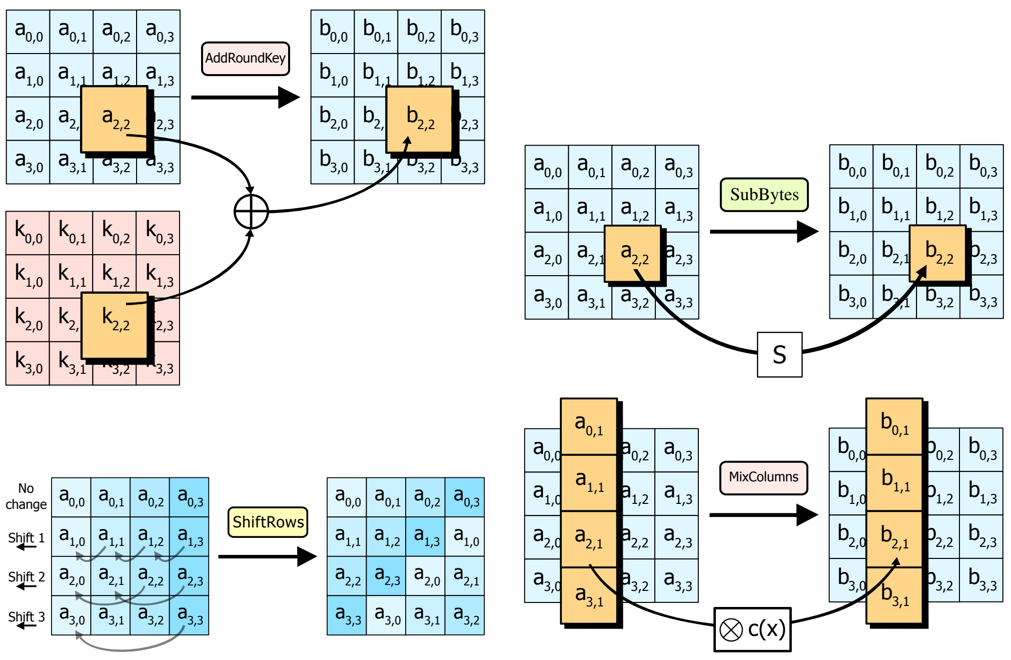
IND-CPA (chosen plaintext, chosen diversifier)

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

1. Challenger generates a key (pair) $k \leftarrow \mathrm{Gen}()$
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 $c = \mathrm{Enc}_k(d, m_b)$ 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: $\epsilon = \left|\Pr[\mathrm{win}] - \frac{1}{2}\right|$.)

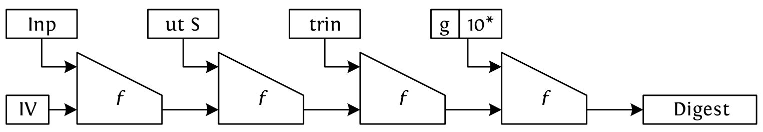
The adversary **must** respect that d **is a nonce**! The values of d used in steps 2, 3 and 5 must all be different.

The step mappings inside Rijndael/AES



MDS property: If N bytes change at the input of MixColumns and as a result M bytes change at its output, then either $N = M = 0$ or $N + M \geq 5$.

The Merkle-Damgård construction



HMAC (simplified)

$$\text{HMAC-}H_K(\text{message}) = H(K \| H(K \| \text{message}))$$

The Diffie-Hellman protocol

In the multiplicative group setting, the public parameters are the group $G = \mathbb{Z}_p^*$ (with p prime) and a generator g. The keypairs (SK, PK) of Alice and Bob are respectively $(a, A = g^a)$ and $(b, B = g^b)$, for a, b in $\mathbb{Z}_p^*$.
The key exchange is then described by the following algorithm :

1. Alice sends A to Bob ; Bob sends B to Alice
2. Alice computes B^a ; Bob computes A^b
3. Alice outputs B^a as the shared secret ; Bob outputs A^b as the shared secret

Université Libre de Bruxelles
Faculté des Sciences

August 25, 2020

INFO-F-405 Introduction to cryptography

This assessment may be done with the use of personal notes, slides, books, web pages, etc. However, the assessment is strictly personal and you are not allowed to communicate with other students or other people during this assessment.

Question 1

(5.6/20) Public-key cryptography In the following, we describe a key agreement protocol, based on ephemeral Diffie-Hellman and on elliptic curves, that Alice and Bob use to obtain a common secret key k_{AB}. However, we voluntarily added some mistakes (including omissions) to its definition. Some of them are functional, that is, they prevent the protocol from working correctly, while others introduce security flaws.

Please *list all the mistakes* you find, and for each, *justify* why it is incorrect and *propose a correction.*

Let $\mathcal{E}$ be an elliptic curve over $\mathbb{Z}_p$ (for some prime p) that two parties, called Alice and Bob, agreed upon. They also agreed on a base point $G \in \mathcal{E}$ with prime order q, i.e., $[q]G$ is the neutral element.

Note that we describe the protocol only from Alice's point of view. The protocol from Bob's point of view is exactly the same, but with reverted roles and variables (i.e., $a, A, e, E, n_{\mathrm{A}}$ become $b, B, f, F, n_{\mathrm{B}}$, respectively, and vice-versa). Also, note that the uppercase letters refer to points on $\mathcal{E}$, while lowercase letters are integers.

Alice's key generation:

1. Alice secretly generates her secret key a randomly and uniformly from $[1 \ldots p-1]$.
2. Alice computes her public key $A = [a]G$, and publishes it via some PKI process. (This aspect is out of scope of this question. In the sequel, we assume that all the public keys are correctly authenticated and can be trusted.)
3. Alice secretly generates her secret value n_{A} randomly and uniformly from $[1 \ldots p-1]$.

Key agreement protocol, from Alice's point of view

1. Alice generates e randomly and uniformly from $[1 \ldots p-1]$.
2. Alice computes $E = [e]G$.
3. Alice signs E by calling $\mathrm{Sign}_a(E)$.
4. Alice sends both E and $\mathrm{Sign}_a(E)$ to Bob.
5. Alice receives $F = [f]G$ from Bob.

6. Alice computes $K_{\text{AB}} = [e]F$.
7. Alice computes the common secret key $k_{\text{AB}} = \text{hash}(K_{\text{AB}} \| a \| e)$.

Procedure $\text{Sign}_a(m)$

1. Hash m: $h = \text{hash}(m)$.
2. Compute $R = [n_{\text{A}}]G$ and set r to the x-coordinate of R modulo p.
3. If $r = 0$, randomly generate a new n_{A} and restart from line 2.
4. Compute $s = n_{\text{A}}^{-1}(h + ar) \bmod p$.
5. Return (R, s).

Procedure $\text{VerifySignature}_a(m, R, s)$

1. Check that $R \in \mathcal{E}$, otherwise the signature is invalid.
2. Hash m: $h = \text{hash}(m)$.
3. Set r to the x-coordinate of R modulo p.
4. Compute $A = [a]G$.
5. Compute $[r]A - [s]R$ and $[h]G$.
6. The signature is valid iff $[r]A - [s]R = [h]G$.

Question 2

(3.2/20) Hashing Let $f : \{0,1\}^* \to \{0,1\}^n$ be a one-way collision-resistant compressing function and $\mathsf{E}_k : \{0,1\}^n \to \{0,1\}^n$ a block cipher with key $k \in \{0,1\}^n$. For a $2n$-bit integer x, we write x_H to denote the n first (most significant) bits of x and x_L to denote the n last (least significant) bits of x such that $x = x_H || x_L$. We construct a compressing function $g : \{0,1\}^* \to \{0,1\}^{2n}$ as follows:

$$g(x) = \begin{cases} \mathsf{E}_{x_L}(x_H) || x_L & \text{if } |x| = 2n \\ f(x) || f(x) & \text{otherwise.} \end{cases}$$

where $|x|$ denote the bit-size of x.

Is $g(x)$ a collision resistant function? What about one-wayness?

Be careful to formally motivate your answer !

Question 3

(5.6/20) Practical application You have been appointed as the new project manager of the COVID-19 tracing team. The goal of your project is to prevent the spread of the virus by letting people know when they have been in contact with an infected person. A trivial solution would be to force every person receiving a positive result for their test to publicly announce where they have been and who they met in the last 14 days. Naturally, this would be unacceptable as it would be a huge infringement of privacy. As an ethic technology enthusiast, you realize that cryptography can be used to construct a less invasive solution, more respectful of privacy.
We ask you to depict such a solution by giving a high-level explanation of your idea followed by a formal description of the cryptographic primitives used during the communications. In addition to that, we ask you to explain what are the different privacy and security threats that you identified in your analysis of the situation and how your solution is actually preventing them. The question is rather open, but we will make the following assumptions to simply the situation:

- Every person possess a smartphone that can be loaded with code from a trusted source.
- Those smartphones can communicate over the Internet as well as in short-range in a peer-to-peer fashion.
- Tests are performed by honest medical doctors who have access to the Internet and can retrieve information from the phone of their patient.
- A global database is available online.

Question 4

(5.6/20) Inside symmetric primitives Below are the specifications of a block cipher under development, called ABCD. We give the specifications of the evaluation of the block cipher in the forward direction, ask you some questions about it, then we ask you to write the specifications of the inverse block cipher.

ABCD is a 256-bit block cipher with a variable-length key. The authors were not very much inspired and gave it its name because the 256-bit input block and running state are represented as 4 words (or rows) of 64 bits each, denoted a, b, c and d. We denote the individual bits in a rows using subscripts, e.g., row a is composed of the bits $a_0, a_1, \ldots, a_{63}$. The four bits (a_i, b_i, c_i, d_i) at the same position in each row is called a column. In other words, the state can be viewed as a grid of 4 rows by 64 columns.

The evaluation of the block cipher consists in 13 rounds. We do not give the specifications of the key schedule, but we assume that, from the input secret key K, one can derive 14 round keys $K_0, K_1, \ldots, K_{13}$ of 256 bits each. The evaluation of block cipher goes as follows:

```
AddRoundKey($K_0$)
for each round $i = 1$ to 13 do
  ShakeColumns
  PreShiftRows
  StirColumns
  PostShiftRows
```

```
    AddRoundKey($K_i$)
  end for
```

We now describe the five step mappings AddRoundKey, ShakeColumns, PreShiftRows, StirColumns and PostShiftRows, all specified at the bit level, or more formally, in the Galois field $\mathrm{GF}(2) = \{0, 1\}$. In this field, $x + y$ (resp. xy) denotes the modulo-2 addition (resp. multiplication) of $x, y \in \mathrm{GF}(2)$. We use the operator $\oplus$ to express the component-wise addition of vectors of elements in GF(2), such as rows, columns or states.

Definition of AddRoundKey(K_i)

```
  $(a, b, c, d) \leftarrow (a, b, c, d) \oplus K_i$
```

Definition of ShakeColumns

```
  for each column $i = 0$ to 63 do
    $p \leftarrow a_i + b_i + c_i + d_i$
    $a_i \leftarrow a_i + p$
    $b_i \leftarrow b_i + p$
    $c_i \leftarrow c_i + p$
    $d_i \leftarrow d_i + p$
  end for
```

Definition of PreShiftRows

```
  $a \leftarrow a$
  $b \leftarrow \mathrm{rot}^1(b)$
  $c \leftarrow \mathrm{rot}^3(c)$
  $d \leftarrow \mathrm{rot}^9(d)$
```

For a row x, $\mathrm{rot}(x)$ denotes the operation that consists in cyclically shifting all the bits by one position, i.e.,

$$(x_0, x_1, x_2, \ldots, x_{63}) \to (x_1, x_2, \ldots, x_{63}, x_0).$$

Definition of StirColumns

```
  for each column $i = 0$ to 63 do
    $a_i \leftarrow a_i + b_i c_i$
    $b_i \leftarrow b_i + c_i d_i$
    $c_i \leftarrow c_i + d_i a_i$
    $d_i \leftarrow d_i + a_i b_i$
  end for
```

Definition of PostShiftRows

```
  $a \leftarrow a$
  $b \leftarrow \mathrm{rot}^1(b)$
  $c \leftarrow \mathrm{rot}^5(c)$
  $d \leftarrow \mathrm{rot}^{25}(d)$
```

Sub-question A: Classify each step mapping as linear or non-linear. As a reminder, a mapping λ is linear iff $\forall x, y : \lambda(x \oplus y) = \lambda(x) \oplus \lambda(y)$. For linear mappings, please provide a short justification (one line). For non-linear mappings, please provide an example of x, y that violates the definition of linearity.

Sub-question B: Point out the step mapping(s) that provide diffusion, with a short justification. As a reminder, a step mapping provides diffusion if at least one input bit influences more than one output bit.

Sub-question C: For the step mapping ShakeColumns, please explain what happens at the output when:

- one flips 1 bit at the input;
- one flips 2 bits of the same column at the input;
- one flips 2 bits of different columns at the input.

Sub-question D: Write the specifications of the inverse of ABCD.

Université Libre de Bruxelles
Faculté des Sciences

August 19, 2021

INFO-F-405 Introduction to cryptography

This assessment may be done with the use of personal notes, slides, books, web pages, etc. However, the assessment is strictly personal and you are not allowed to communicate with other students or other people during this assessment.

Optionally, you may skip one sub-question of your choice. If you decide to do so, you have to write clearly and explicitly which sub-question you skip, e.g., "I am skipping sub-question 1A." The skipped question will then not be graded and the total of the points will be rescaled accordingly.

Question 1

(5/20) Practical aspects of cryptography Alice has recently landed in Belgium for a one-year Erasmus at ULB/VUB. Just like any other student, she has been granted access to the university supercomputer, *Hydra*, to run experiments for her Master's Thesis.

We will assume that the connection protocol with Hydra consists in a Diffie-Hellman key exchange followed by the symmetric encryption of the communications. Upon the first connection, Alice's computer requests Hydra's long-term public key and stores it for the subsequent connections. Then for each connection, Alice's computer generates an ephemeral key pair and sends the public key to Hydra. Finally, each party generates a secret key from its private key and the public key received from the other party.

Carelessly, Alice connects to Hydra with this protocol from her student residence.

Sub-question 1A. Let us assume that an attacker (Charles) has control over the network of Alice's student residence. Charles knows that many students will try to connect to Hydra. How can Charles intercept and read all communications between Alice's computer and Hydra in the clear without being noticed?

Sub-question 1B. The next day, Charles does not intercept connections to Hydra anymore. What will Alice notice when she connects to Hydra? Why?

Sub-question 1C. In SSH, the first time you connect to a host, you get prompted with a *fingerprint* and asked whether that *fingerprint* matches what you expect from the remote host you are trying to reach. What is this fingerprint? To what is it applied? What kind of attack does it prevent?

Sub-question 1D. Let us assume that there is a trusted authority such as the Belgian government in the loop. Could the ULB/VUB prevent attacks such as the one Charles is able to perform via this trusted authority ?

Question 2

(5/20) Secret-key cryptography An automated teller machine (ATM) distributes cash to the customers of a bank. It is connected to the bank's server, which is in charge of authorizing and tracking transactions. The bank and the ATM share a k-bit secret key K. When authorizing, the bank sends the ATM a triplet $(u, m, \texttt{tag})$ allowing user u to receive an amount m in cash, and tag is a n-bit message authentication code (MAC) computed under key K over the first two components of the triplet. (This is of course a very simplified view for the sake of this question.)

Sub-question 2A. What is the bank trying to protect against, assuming an adversary who has full access to the network connecting the bank and the ATM?

Sub-question 2B. Let us assume for now that the bank and the ATM use a secure MAC function. What is the minimum key length k that is required to have a security strength of 128 bits, and why?

Sub-question 2C. Let us further assume that the ATM processes not more than one triplet per second and that the secret key K is securely renewed every 24 hours. What is the minimum length n of the MAC such that the probability of fraudulent transaction is less than 10^{-12} per day, and why?

Sub-question 2D. Let us now assume that $n = k$ and that the MAC function is constructed as follows:

$$\text{MAC}_K(\text{message}) = \text{hash}(\text{message}) \oplus K,$$

where hash$(\cdot)$ is a secure n-bit hash function and $\oplus$ denotes the bitwise mod-2 addition (or XOR). Is this MAC function secure? If so, please justify briefly. If not, please describe an attack (preferably in the light of the EU-CMA game).

Sub-question 2E. Let us consider another MAC function. We assume again that $n = k$, but the MAC function is now constructed as follows:

$$\text{MAC}_K(\text{message}) = \text{hash}(K_1 \| \text{message}) \oplus K,$$

where hash$(\cdot)$ is a secure n-bit hash function, $\|$ denotes the concatenation and $K = K_1 \| K_2$ with $|K_1| = |K_2| = k/2$. What is the security strength of this MAC function, and why?

Question 3

(4/20) Hashing Various sub-questions.

Sub-question 3A. For a given hash function, does second preimage resistance imply collision resistance, or vice-versa, and why?

Sub-question 3B. A friend of yours designed his/her own hash function CATSHA320, with 320 bits of output. It is an iterated hash function that cuts the input message into blocks of 192 bits, processes them sequentially and has a chaining value of 224 bits. Without knowing more about this hash function, what is its expected collision resistance (in number of attempts), and why?

Sub-questions 3C and 3D. Consider SHAKE128, a standard sponge function with permutation f, capacity $c = 256$ bits and rate $r = 1344$ bits. Let us assume that it is used in an application that hashes only small input messages that fit in a single bock of r bits (even after padding[1]), and where the output size is $n = 256$ bits.

- Sub-question 3C. First, write symbolically (i.e., as a mathematical expression) the output as a function of the input message for the restricted use case of this application.
- Sub-question 3D. Then, knowing that f and its inverse f^{-1} can both be evaluated efficiently, is this hash function preimage-resistant? If so, please justify briefly. If not, please describe an attack.

[1]The details of the padding have little relevance in this question, so we may assume that the message M is simply padded as $M||10^*$, i.e., with a single bit 1 followed by the minimum number of bits 0 such that the padded message is r-bit long.

Question 4

(6/20) Public-key cryptography Alice frequently needs to upload files to Bob's server, and they use RSA and hybrid encryption to keep their files confidential. Please find below the specifications of their protocol, similar to RSA-KEM, to which we added a few mistakes.

One-time setup

1. Bob sets $e = 3$.
2. Bob randomly chooses a private prime number p of 256 bits. He checks that $\gcd(e, p) = 1$; otherwise, he repeats with a new prime p.
3. Similarly, Bob randomly chooses another private prime number q of 256 bits. He checks that $\gcd(e, q) = 1$ and $p \neq q$; otherwise, he repeats with a new prime q.
4. Bob computes $d = e^{-1} \bmod \phi(pq)$ and keeps d private.
5. Bob computes $n = pq$ and sends (e, n) to Alice. They check together that Alice received Bob's public key correctly.

When Alice needs to upload a file F

6. Alice chooses a random string m of 512 bits.
7. Alice computes the 160 bits of output of $\text{SHA1}(m)$ and interprets them as the integer k with $0 \leq k \leq 2^{160} - 1$.
8. Alice computes $c = k^e \bmod n$, and she encrypts her file F with a good symmetric encryption scheme using the secret key k resulting in ciphertext G.
9. Alice uploads (c, G).

When Bob receives an encrypted file (c, G)

10. Bob recovers k by computing $k = c^d \bmod n$.
11. Bob chooses a random string m of 512 bits and computes $k' = \text{SHA1}(m)$. If $k' \neq k$, it outputs an error message and aborts.
12. Bob decrypts G with the same good symmetric encryption scheme, using k as secret key, to recover F.
13. Bob stores F on the server.

Sub-question 4A. What is the size in bits of Bob's modulus n that will result from the setup? Does that give sufficient security in 2021? If so, please justify briefly. If not, please recommend which size the primes should have to achieve about 128 bits of security.

Sub-question 4B. On line 4, what is $\phi(pq)$? Can you give a simpler expression? What would happen if $\phi(pq)$ was part of Bob's public key?

Sub-question 4C. There is a mistake in the one-time setup procedure that can cause the computation to fail sometimes. What is it? How can you fix it, and why? Before it is fixed, what is approximately the probability that this procedure fails?

Sub-question 4D. There is a mistake in the way RSA is used that causes a major security issue. What is it? How can you fix it, and why?

Sub-question 4E. There is a silly mistake in Bob's procedure that causes it to fail almost always. What is it? How can you fix it, and why?

Sub-question 4F. Can someone other than Alice upload a file onto Bob's server? If so, please sketch briefly how to modify the protocol so that only Alice can upload a file. Otherwise, please explain briefly how the current protocol achieve this restriction.

Université Libre de Bruxelles
Faculté des Sciences

January 22, 2020

INFO-F-405 Introduction to cryptography

This assessment has to be done without the use of personal notes, books or any other support. You may use a calculator (but it is not required). You have to justify and detail each of your answers. Indicate your name, first name and year of study on every answer sheet. The answers to the questions should be on different answers sheets (on sheet per question; the multiple-choice questions have to be together on one sheet).

Question 1

(3.8/20) Secret-key encryption The Electronic Codebook (ECB) mode is a flawed encryption mode on top of a block cipher.

a) What makes it an insecure mode?

b) What could an adversary make as queries and choice of (m_0, m_1) to win the IND-CPA game?

c) Describe a mode that turns a block cipher into a stream cipher. How can this mode securely handle the encryption of multiple messages using the same secret key?

Note: both ECB and IND-CPA are given in the portfolio.

Question 2

(3.8/20) Hashing Let $\mathcal{RO}(x)$ be a random oracle that outputs an infinite sequence of uniformly and independently distributed bits for each input $x \in \{0,1\}^*$. If the random oracle is queried twice with the same input, it always returns the same sequence. We denote with $\mathcal{RO}(x, \ell)$ the output of $\mathcal{RO}(x)$ truncated after the first ℓ output bits, so $\mathcal{RO}(x, \ell) \in \{0,1\}^\ell$.

a) *Preimage attack.* Given some value $y \in \{0,1\}^\ell$, an adversary would like to find a preimage, i.e., an input x such that $\mathcal{RO}(x, \ell) = y$. What is the probability of success after t random attempts?

b) *Multi-target preimage attack.* Given a set of m distinct values $\{y_1, \dots, y_m\}$, with $y_i \in \{0,1\}^\ell$, an adversary would be happy to find a preimage of any one of these images. What is the probability of success after t random attempts?

Let $h(x)$ be a concrete extendable output function (XOF) that outputs a potentially infinite sequence of bits, and we similarly denote $h(x, \ell)$ its truncation to ℓ output bits. As of today, the only known attack against $h(x)$ has a probability of success $\epsilon(t) = t/2^{c/2}$, for some security parameter c, and where t is the time spent by the adversary expressed in the number of attempts. This means that, except for this probability $\epsilon(t)$, we can model $h(x)$ as a random oracle.

a) What is the range of output sizes ℓ and parameter values c such that $h(x)$ offers 128 bits of preimage resistance?

b) What is the range of output sizes ℓ and parameter values c such that $h(x)$ offers 128 bits of multi-target preimage resistance with $m = 2^{32}$?

You may use the approximation $\log(a + b) \approx \max(\log a, \log b)$.

Question 3

(3.8/20) Public-key signature with Schnorr Consider the Schnorr-type signature scheme based on elliptic curves in the portfolio.

a) What does the notation $X = [y]Z$ mean?

b) Please explain why the verification succeeds on a genuinly generated signature, and expand the computations annotated with the properties used.

c) The value k must be randomly generated and unique per signature. What happens if k is repeated?

d) To make k random and unique, the signer generates a random k upon the first signature, and then increments it ($k \leftarrow k + 1$) for each new signature. Is that secure? If so, please justify. If not, what is the problem and how can you fix it?

Question 4

(3.8/20) Practical question A company that installs and maintains an open-source operating system is creating an automatic update mechanism for its customers. Each computer connected to the internet can fetch an *update pack* containing the latest changes to be made to the operating system. The company would like to guarantee the integrity of these update packs so as to avoid the installation of malicious software on their customers' computers.

Please propose a concrete solution based on schemes such as encryption, either public-key or secret-key, authentication, authenticated encryption, signature and hashing. You also need to describe how the keys are generated and/or distributed.

Question 5

(4.8/20) Multiple choices on various subjects For each of the following sub-questions, the relative rating is: 0.8 points for a correct answer, 0.0 for a wrong answer and 0.3 for "I don't know".

A. After how many attempts can one typically find a collision on a secure hash function with an output length of 384 bits?

(0) I don't know.
(1) 128
(2) 192
(3) 384
(4) 2^{128}
(5) 2^{192}
(6) 2^{384}

B. What happens if the one-time pad is incorrectly used and that two distinct plaintexts are encrypted with the same key?

(0) I don't know.
(1) The key is compromised.
(2) The two plaintexts are revealed.
(3) The difference between the two plaintexts is revealed.
(4) The authenticity of the plaintext is compromised.

C. What is the relationship between the discrete logarithm (DL) and the Diffie-Hellman (DH) problems?

(0) I don't know.
(1) An adversary who can solve the DL problem can also solve the DH problem, but not necessarily vice-versa.
(2) An adversary who can solve the DH problem can also solve the DL problem, but not necessarily vice-versa.
(3) Both problems are equivalent.

D. How does hybrid encryption work?

(0) I don't know.
(1) Alice sends a secret key to Bob encrypted with his public key, and she encrypts the plaintext with the secret key.
(2) Alice sends a public key to Bob encrypted with their secret key, and she encrypts the plaintext with his public key.
(3) Alice sends a public key to Bob encrypted with her private key, and she encrypts the plaintext with her public key.
(4) Alice sends a private key to Bob encrypted with his public key, and she encrypts the plaintext with his public key.

E. Let E be an elliptic curve over $\mathrm{GF}(p)$ for a given prime number p and let $\#E$ be the number of points on E. Which statement is most plausible? (The symbol $\sim$ means "is of the order of".)

   (0) I don't know.

   (1) $p \sim 2^{512}$, $\#E \sim 2^{256}$ and E offers a security of ≈ 128 bits.

   (2) $p \sim 2^{256}$, $\#E \sim 2^{128}$ and E offers a security of ≈ 128 bits.

   (3) $p \sim 2^{256}$, $\#E \sim 2^{256}$ and E offers a security of ≈ 128 bits.

F. Assume an adversary performs an exhaustive key search on a huge network of 10^9 computers, each capable of testing 10^9 keys per second. After about how much time will a 128-bit key typically be found?

   (0) I don't know.

   (1) A few seconds.

   (2) A few days.

   (3) A few years.

   (4) A few centuries.

   (5) A few times the age of the universe.

Portfolio

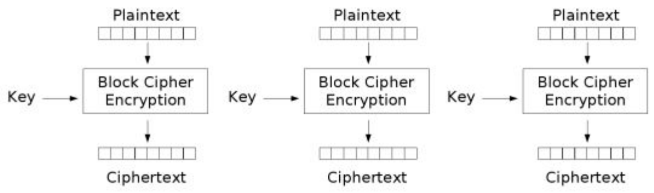
Electronic Codebook (ECB)

Input: secret key $k \in \{0,1\}^m$, plaintext $p \in \{0,1\}^*$, output: ciphertext $c \in \{0,1\}^*$

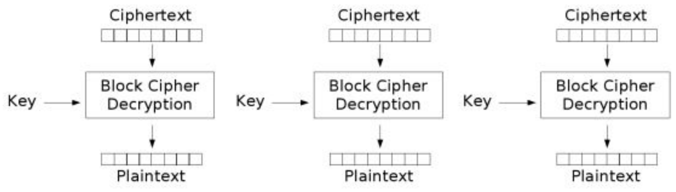
- Pad p with 10^* to reach a multiple of the block size n
- Cut $p \| 10^*$ into blocks p_i of size n
- Process each block independently through E_k

$$c_i = E_k(p_i)$$

And similarly with E_k^{-1} for decryption.



Electronic Codebook (ECB) mode encryption



Electronic Codebook (ECB) mode decryption

Indistinguishability under chosen plaintexts (IND-CPA)

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

- Challenger generates a key (pair) $k \leftarrow \mathrm{Gen}()$
- Adversary queries Enc_k with plaintexts of his choice
- Adversary chooses two plaintexts $m_0, m_1 \in M$ with $|m_0| = |m_1|$
- Challenger randomly chooses $b \leftarrow_R \{0,1\}$, encrypts m_b and sends $c = \mathrm{Enc}_k(m_b)$ to the adversary
- Adversary queries Enc_k with plaintexts of his choice
- Adversary guesses the index b' of the plaintext was encrypted
- Adversary wins if $b' = b$

Schnorr-type signature scheme on elliptic curve

Key pair on curve E and base point G of order q

- Private key $a \in \mathbb{Z}_q$
- Public key $A = [a]G$

Signature of message $m \in \{0,1\}^*$ by Alice with her private key a:

- Randomly choose k in $\mathbb{Z}_q$
- Compute $R = [k]G$
- Compute $e = \mathrm{hash}(R \| A \| m)$
- Compute $s = k + ea \bmod q$
- Send (R, s) along with m

Verification of signature (R, s) on m with Alice's public key A:

- Check $[s]G \overset{?}{=} R + [\mathrm{hash}(R \| A \| m)]A$

Université Libre de Bruxelles
Faculté des Sciences

January 20, 2021

INFO-F-405 Introduction to cryptography

This assessment may be done with the use of personal notes, slides, books, web pages, etc. However, the assessment is strictly personal and you are not allowed to communicate with other students or other people during this assessment.

Question 1

(5/20) Practical aspects of cryptography Thelma and Louise recently met for the first time and would like to do business together. After their meeting, because of the COVID-19, they are forced to stay at home and can communicate only by phone or via the Internet. They wish to start discussing their business project, but want to do so confidentially. Hopefully, they already exchanged their phone numbers and email addresses, and are both knowledgeable about cryptography. However, their project is highly sensitive and they need to protect their conversation against even active adversaries, that is, people who can afford to tamper with Internet connections. We nevertheless assume that the adversaries are not going to cut their communication lines, as this would be too obvious and as they would otherwise miss all opportunities of spying on the two business women.

Sub-question A: Please explain how Thelma and Louise can set up a secure communication channel using cryptography, and how they can use it to communicate securely. If they use keys, specify what kind of keys and which key(s) belong(s) to whom. List clearly the assumptions that allow their communication to be confidential, including potential threats that you think your solution does not (completely) prevent against (if any).

Your solution can make black-box use of cryptographic building blocks such as secret-key encryption, message authentication code, authenticated encryption, hash function, public-key encryption, key agreement or signature. (There is no need to specify a particular choice of algorithms behind these building blocks.)

Sub-question B: The same story happens with Tom and Jerry. Unlike Thelma and Louise, however, the two businsess men have been friends since childhood, and they share lots of common memories. Many of these memories can be considered secret, as they are not known to anyone except themselves (or at least, they are unknown to the adversaries).

Propose a way for Tom and Jerry to set up a secure communication channel that takes advantage of these secret memories.

Question 2

(5/20) Secret-key cryptography Consider the following encryption mode called NXCTR for “Nonce-Xored CounTeR”. It is a block cipher-based mode very similar to the original CTR, but the nonce is XORed into the output of the block cipher instead of being part of its input. More precisely, NXCTR works as in Algorithm 1 and is illustrated in Figure 1. However, NXCTR is flawed.

A. What is the practical advantage of NXCTR over the original CTR? And over CBC?

B. What is/are intuitively the security problem(s) of NXCTR?

C. Describe how an adversary can win the IND-CPA game against NXCTR with only practical data and time complexities.

D. How can you change NXCTR to make it IND-CPA secure? Why does it solve the problem?

Algorithm 1 The NXCTR encryption mode on top of block cipher E with block size n bits and key size m bits.

```
Input: secret key $K \in \mathbb{Z}_2^m$, plaintext $p \in \mathbb{Z}_2^*$ and nonce $N \in \mathbb{N}$
Output: ciphertext $c \in \mathbb{Z}_2^{|p|}$

Cut $p$ into blocks of $n$ bits $(p_0, p_1, p_2, \ldots, p_l)$, except for the last one that can be shorter
Generate the keystream, for $i = 0$ to $l$

        $k_i = E_K(\text{block}(i)) \oplus \text{block}(N)$

(here block($x$) encodes the integer $x$ into a string in $\mathbb{Z}_2^n$)
Truncate the last keystream block $k_l$ to the size of $p_l$
Compute the ciphertext $c_i = k_i \oplus p_i$, for $i = 0$ to $l$
```

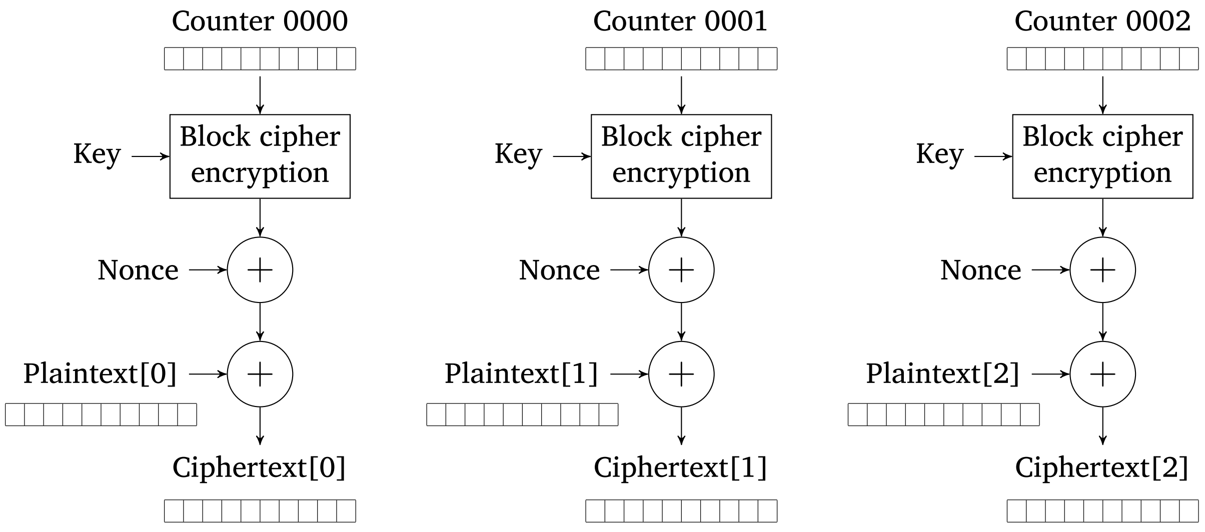


Figure 1: The NXCTR encryption mode.

Question 3

(5/20) Hashing Let π be a cryptographically secure permutation that maps $b = 1600$-bit strings onto b-bit strings. We assume that π and its inverse π^{-1} are both equally efficiently implementable. We build a hash function by applying the sponge construction on top of π. The hash function has a fixed output length of $n = 256$ bits. However, for the sake of this question, we set the capacity to $c = 0$.

In more details, for an input bit string m of any length, we proceed as follows:

- Padding: we append to m a bit 1, followed by the minimum number of bits 0 such that its length is a multiple of b bits.
- Cutting: we cut the padded input string into l blocks of b bits each: $m_1, m_2, \ldots, m_l$.
- Initialization: we set the state s to 0^{1600}, the string of 1600 zero bits.
- For each block m_i, with i running from 1 to l, we do the following:
  - We bitwise modulo-2 add (or XOR) m_i to the state: $s \leftarrow s \oplus m_i$.
  - We apply the permutation to the state: $s \leftarrow \pi(s)$.
- We return as hash the first n bits of s.

Sub-question A: Please explain how to obtain a collision with this hash function.

Sub-question B: Given two prefixes x and y of b bits each, show how to obtain a collision between one input that must start with x and the other one with y.

Sub-question C: Given an output value z of n bits, describe how to find a preimage with this hash function.

Sub-question D: Now let us assume that this hash function is used exclusively in an application that hashes input strings made of ASCII-encoded text, for which each byte has its most significant bit always set to 0. More precisely, we restrict the input strings m such that every 8th bit is 0, and for the sake of simplicity this is the only restriction we consider (i.e., we do not care whether m is actually valid ASCII-encoded text or not). Please explain how to obtain a collision with this hash function such that the colliding inputs have this restriction. What is the complexity of your attack?

Sub-question E: Given an output value z of n bits, describe how to find a preimage with this hash function such that the preimage has the aforementioned restriction. What is the complexity of your attack?

Question 4

(5/20) Public-key cryptography A company that installs and maintains an open-source operating system is creating an automatic update mechanism for its customers. Regularly, the company publishes an *update pack* containing the latest changes to be made to the operating system, and each computer connected to the Internet can fetch it. The company would like to guarantee the integrity of these update packs so as to avoid the installation of malicious software on their customers' computers. Fortunately, confidentiality is not required, as it is open-source software.

In the following, we describe a protocol that the company uses to sign the update packs and for the customers to check that the update packs are genuine. However, we voluntarily added some mistakes (including omissions) to its definition. Some of them are functional, that is, they prevent the protocol from working correctly, while others introduce security flaws.

Please *list all the mistakes* you find, and for each, *justify* why it is incorrect and *propose a correction.*

Let $\mathcal{E}$ be a standard elliptic curve over $\mathrm{GF}(p)$ (for some prime p) that is used by the company and its customers. They also agreed on a base point $G \in \mathcal{E}$ with prime order q, i.e., $[q]G$ is the neutral element. Note that the uppercase letters refer to points on $\mathcal{E}$, while lowercase letters are integers, and `UP` is a string of bits that represents an update pack.

One-time setup at the company:

1. The company secretly generates its secret key c randomly and uniformly in $[1 \dots p-1]$.
2. The company computes its public key $C = [c]G$, and publishes it via some PKI process. (This aspect is out of scope of this question. In the sequel, we assume that all the public keys are correctly authenticated and can be trusted.)
3. The company secretly generates its secret value n_C randomly and uniformly in $[1 \dots p-1]$.

Company's procedure to sign and post an update pack `UP`

1. Compute $k = \mathrm{hash}(n_\mathrm{C})$.
2. Compute $R = [k]G$.
3. Compute $e = \mathrm{hash}(R \| \mathtt{UP})$.
4. Compute $s = k + ec \bmod p$.
5. Post $(\mathtt{UP}, e, s)$ on the company's update repository.

Customer's procedure to retrieve and install an update pack

1. Retrieve $(\mathtt{UP}, e, s)$ from the company's update repository.
2. Compute $C = [c]G$.
3. Compute $R' = [s]G - [e]C$.
4. Compute $e' = \mathrm{hash}(R' \| \mathtt{UP})$.
5. The customer installs `UP`.