Introduction to Cryptography: exam

June 16, 2021

Instructions: You can score a maximum of 100 points and you have 180 minutes to solve all nine problems. Each question indicates how many points it is worth. You are **not** allowed to use any books/slides/notes/*etc.*, nor a smart phone or any device. Please write clearly and **explain your answers**. Each exercise is independent and they can be solved in the order of your choice. The formula sheet can be found attached at the end of the exam. Do not forget to **put your name and student number on each sheet.**

1. (**10 points**) **Symmetric cryptography fill-in question.** In the following text, fill in the missing notions. There is no explanation necessary.

   In symmetric cryptography, there are two types of encryption: __(a)__ and stream encryption. Stream encryption is usually done with a stream cipher. A (modern) stream cipher takes two inputs, namely __(b)__ and __(c)__. It outputs a keystream. The encryption algorithm then consists of adding the keystream to the plaintext. This mimics the only encryption scheme that provides perfect secrecy: __(d)__.

   Another kind of cipher in symmetric cryptography is the __(e)__. Examples of this cipher are DES and __(f)__. There exist modes with which you can turn (e) into a stream cipher, e.g., __(g)__.

   Additionally, we can authenticate messages with __(h)__. We prove the security of a (h) by showing that it is hard to distinguish it from its corresponding ideal counterpart, i.e., __(i)__, with non-negligible advantage. One way to create such a distinguisher is based on __(j)__ attacks, that, in general, threaten the goal of (h). In such attacks, you can use several (online) generation queries and (online) verification queries.

   Solution:

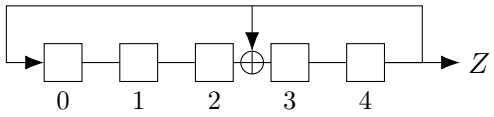
   (a) block encryption;
   (b) diversifier/key;
   (c) key/diversifier;
   (d) one-time pad;
   (e) block cipher;
   (f) AES; (or Triple-DES)
   (g) counter mode (CTR); or, output feedback mode (OFB);
   (h) message authentication function (MAC);
   (i) (truncated) random oracle;
   (j) forgery.

   [[Grading Instruction:

Grading (total 10):	
aspect:	points
each correct term/notion	1

   ]]

2. (**9 points**) **LFSR security** Consider the following linear feedback shift register.



(a) Give the upper bound for the security strength when only considering exhaustive key search. 3 pt

(b) Consider the output stream $Z = \texttt{10001}$. Determine the initial state of the LFSR. 6 pt

Solution:

(a) Since the key (initial state) consists of 5 bits, the security strength of exhaustive key search will be 5 bits.

(b) We iterate the LFSR with a state $\texttt{s}_0\texttt{s}_1\texttt{s}_2\texttt{s}_3\texttt{s}_4$, substituting as we go:

$$\begin{aligned} s_0s_1s_2s_3s_4 &\rightarrow s_4 = 1 \\ 1s_0s_1(s_2+1)s_3 &\rightarrow s_3 = 0 \\ 01s_0s_1(s_2+1) &\rightarrow s_2 = 0+1 = 1 \\ 001s_0s_1 &\rightarrow s_1 = 0 \\ 0001s_0 &\rightarrow s_0 = 1 \\ 10001 & \end{aligned}$$

Hence, the initial state was 10101.

[[Grading Instruction:

Grading (total 9):	
aspect:	points
(a)	**3**
answer	1
explanation	2
(b)	**6**
answer	2
computation	4

There is room to verify at (b), but there is no time for that I think. Although someone who took the time to indicate that something is wrong will get deducted 1 point less. I will take into account "doorrekenfouten", but an error will cost a point.

]]

3. (**9 points**) **Hashing with a XOF.** Keccak can be instantiated with seven possible values for the permutation width b, i.e., $b \in \{25, 50, 100, 200, 400, 800, 1600\}$. It has two parameters, the rate r and the capacity c, that must sum to b, so $r + c = b$. Keccak generates an arbitrary output length, but you have to fix it to some value n. Determine the minimum values b, c, n for Keccak when the following properties are needed. Give your final answer as $(n, b, c, r) = (., ., ., .)$ and give a concise explanation of your answer.

(a) First preimage resistance with security strength 128 bits; 3 pt

(b) Second preimage resistance with security strength 200 bits; 3 pt

(c) Collision-resistance with security strength 288 bits. 3 pt

Solution:

(a) $(n, b, c, r) = (128, 400, 256, 144)$.
Security strength against first preimage attacks is bounded by $\min(n, c/2)$, hence we immediately find $n = 128$ and $c = 256$. For b, we just pick the first value such that $b > c$, that is $b = 400$. Then $r = b - c = 400 - 256$.

(b) $(n, b, c, r) = (200, 800, 400, 400)$.
Security strength against second preimage attacks is bounded by $\min(n, c/2)$ again, hence we immediately find $n = 200$ and $c = 400$. For b, we just pick the first value such that $b > c$, that is $b = 800$. Then $r = b - c = 800 - 400 = 400$.

(c) $(n, b, c, r) = (576, 800, 576, 224)$.
Security strength against collision attacks is bounded by $\min(n/2, c/2)$, hence we immediately find $n = c = 576$. For b we must then take the smallest value for which $b > c$, that is $b = 800$. Then $r = b - c = 800 - 576 = 224$.

[[Grading Instruction:

Grading (total 9):	
(a)/(b)/(c)	**3**
all values correct	1
bound correct	1
explanation for b correct	1

For example, someone who only has n wrong, will get 2 out of three point is the explanation is correct. If $r = 0$ is taken, then this will give 1 of the three points, as the explanation for b is wrong.

]]

4. (**12 points**) **Distinguishing one-round Feistel** In this exercise, we consider a one-round Feistel structure. Let $\mathrm{F}_K \colon \{0,1\}^\ell \to \{0,1\}^\ell$ be indistinguishable from a random permutation. We are going to show that this Feistel structure is not PRP-secure. In order to do this, we want to find a distinguisher $\mathcal{D}$ that has non-negligible advantage in distinguishing this Feistel structure from a random permutation.

(a) Give a distinguisher $\mathcal{D}$ that distinguishes the one-round Feistel structure from a random permutation. [Make sure that it will yield a non-negligible advantage in (c).] 4 pt

(b) What is the probability that your distinguisher guesses that it is talking to a one-round Feistel structure, while it is actually talking to a random permutation? 3 pt

(c) Give the advantage of your distinguisher. 3 pt

(d) Is this one-round Feistel structure SPRP-secure? 2 pt

Solution:

(a) For a one-round Feistel structure, we have $C_R = P_R$. This leads to the following distinguisher $\mathcal{D}$. By making one online encryption query, $P_L \| P_R$, we retrieve $C_L \| C_R$ and decide that we are talking to the Feistel structure if $C_R = P_R$.

(b) The probability that $C_R = P_R$ for a random permutation is $2^{-\ell}$ as there are 2^ℓ possible outcomes for the last ℓ bits. Since they have to be this specific right-half of P_R, the probability is $2^{-\ell}$. In that case we would say we talk to the Feistel structure wrongly.

(c) The formula for advantage tells us:

$$\begin{aligned} \mathrm{Adv}_{\mathcal{D}} &= |\Pr[\mathcal{D} = 1 \mid \mathrm{Fst}] - \Pr[\mathcal{D} = 1 \mid \mathcal{RO}]| \\ &= |1 - 2^{-\ell}| \end{aligned}$$

This advantage is quite large, hence certainly non-negligible.

(d) No, if a block cipher is not PRP-secure, the same distinguisher works against SPRP-security, as an SPRP-distinguisher has even more freedom in queries.

[[Grading Instruction:

Grading (total 12):	
aspect:	points
(a)	**4**
query	1
property	2
conclusion	1
(b)	**3**
answer	1
explanation	2
(c)	**3**
adapting advantage formula	1
computation	1
answer	1
(d)	**2**
answer	1
explanation	1

]]

5. (**10 points**) SHADES**, a novel MAC function! But how secure is it?** We build a novel MAC function called SHADES from SHA-256 and DES. It works as follows. It takes the input (message) and hashes it with SHA-256. Then, it truncates the resulting digest to its first 64 bits. Subsequently, it encrypts the result with DES_K with K a secret key. Then, the output of DES is truncated to its first 48 bits and that will be the tag. This question is about the quantitative security strength of this function. Attackers can make use of offline queries (i.e., computations) and online MAC generation and verification queries. Each DES computation and each SHA-256 computation counts as one offline query. We denote the total number of offline queries by N, and the total number of online queries by M.

Exhaustive key search on SHADES: Make two generation queries with input arbitrary (short) messages m_1 and m_2 and keep the resulting tags T_1 and T_2. Compute (offline) for both messages m_1 and m_2 the SHA-256 digests and truncate the results to 64 bits, that we will call p_1 and p_2. Then, try different random keys K' until one is found that satisfies the following two criteria:

(a) The first 48 bits of p_1 encrypted with $\text{DES}_{K'}$ is T_1.

(b) The first 48 bits of p_2 encrypted with $\text{DES}_{K'}$ is T_2.

These checks can be done with offline computations. A key K' that satisfies both checks is very likely to be the right key: $K' = K$. Using a single pair (m, T) would not be sufficient to determine the key as it is highly likely that, in addition to K, there are many other keys K' for which SHADES maps m to T for any pair m and T.

Answer the following questions and provide for each of them a (short) explanation.

(a) Give the success probability of a single key guess in the exhaustive key search. [Hint: DES has a 56-bit key. You can ignore the complementation property here.] 1 pt

(b) Give the security strength of SHADES against exhaustive key search. [Hint: In exhaustive key search, you do many key guesses.] 2 pt

(c) Give the security strength of SHADES against forgery if an attacker can only (blindly) guess tags. 2 pt

(d) You can use collisions to create a forgery. Explain how that would work. [Hint: You do not require a collision on the full output of SHA-256.] 2 pt

(e) Taking into account the attacks in the previous sub-questions, give the security strength of SHADES against tag forgery. 3 pt

Solution:

(a) Success probability is 2^{-56} because there are 2^{56} keys.

(b) This attack has $M = 2$ and after X key guesses corresponding to offline complexity $N = X + 2$, it has success probability $p = X2^{-56}$. The security strength is given by the $\lg((N + M)/p) = \lg((X + 4)/X2^{-56})$. If the number of key guesses is large this can be simplified to $\lg(2^{56}) = 56$ bits. In the other extreme, if $X = 1$ it becomes $\lg(5 \times 2^{56}) \approx 58$ bits. You can delay the 2nd offline call to SHA-256 until the first key is hit that satisfies the first condition. So then for small X we have $M = 1$ instead and we get $\lg(4 \times 2^{56}) \approx 58$ bits.

(c) The tag is 48 bits, so a MAC verification query with a blindly guessed tag will have success probability 2^{-48}. Such an attack has $N = 0$ and M is the number of verification queries. Trying M times gives success probability approximately $M2^{-48}$. So $(N + M)/p = 2^{48}$ and hence the security strength is 48 bits.

(d) Find two messages m_1 and m_2 that have SHA-256 digests with equal first 64 bits. Then make a MAC generation query with m_1 giving tag T_1. Then the pair m_2, T_1 is a forgery: presenting it to a MAC verification query will return OK.

(e) The collision-based attack has $N \approx 2^{32}$ because of the birthday bound and $M = 1$ to succeed with probability 1, so it has associated security strength 32 bits. This is lower than the bounds obtained from exhaustive key search and tag guessing and likely the security strength of SHADES.

[[Grading Instruction:

Grading (total 10):	
aspect:	points
(a)	**1**
(b)	**2**
explanation	1
value	1
(c)	**2**
explanation	1
value	1
(d)	**2**
collision	1
generation query	1
(e)	**3**
explanation	2
value	1

]]

6. (**11 points**) **Public-key cryptography fill-in question.** In the following text, fill in the missing notions. There is no explanation necessary.

   In this course, we have treated three types of public-key cryptography: key establishment (which includes encryption), __(a)__ and __(b)__. To set up a shared secret key to be used for symmetric-key cryptography, Alice and Bob can best use the RSA or ElGamal __(c)__. The security of these depend on the hardness of __(d)__ and the hardness of the computational Diffie-Hellman problem, respectively. However, note that the textbook versions of cryptosystems such as Diffie-Hellman key agreement are vulnerable to man-in-the-middle attacks, because the public keys are not __(e)__.

   Once Alice has set up a public key that is (e), she can prove to Bob who she is with e.g., the Chaum-Evertse-Van de Graaf protocol. Another protocol that can be used for this is __(f)__, which is both secure and efficient. In general, such protocols need to be __(g)__, __(h)__ and (honest-verifier) zero-knowledge.

   Interestingly, (f) can be converted into a signature scheme using the __(i)__ transform. If the verification of this signature scheme is too computationally costly, one can use another signature scheme, e.g., __(j)__. A drawback of this scheme is however that signing is much more costly, although it can be sped up by a factor four with __(k)__.

   Solution:

   (a) cryptographic signatures / authentication protocols

   (b) authentication protocols / cryptographic signatures

   (c) KEMs (key encapsulation mechanisms)

   (d) factoring / inverting RSA (may include something about the hash function being indistinguishable from a random oracle and the symmetric cryptosystem being secure)

   (e) authenticated

(f) the Schnorr authentication protocol

(g) complete / (special) sound

(h) (special) sound / complete

(i) Fiat-Shamir

(j) the FDH-RSA signature scheme

(k) Chinese Remainder Theorem (CRT)

[[Grading Instruction:

Grading (total 11):	
aspect:	points
for each notion	1

]]

7. (**15 points**) **Elliptic curves.** Consider the elliptic curve $\mathcal{E} : y^2 = x^3 + 5x - 1$ over $\mathbb{F}_{23}$.

   (a) Show that $P := (7, 3)$ and $Q := (22, 4)$ are on the curve $\mathcal{E}$. 2 pt

   (b) Compute $[2]Q$. 4 pt

   (c) Compute $P + Q$. 4 pt

   (d) Let it be given that $\#\mathcal{E}(\mathbb{F}_{23}) = 17$. Give a generator of the group $\mathcal{E}(\mathbb{F}_{23})$. Explain your answer! 3 pt

   (e) Give the compressed point representation of P. 2 pt

   Solution:

   (a) For P: $7^3 + 5 \cdot 7 - 1 \equiv 7 \cdot 3 + 5 \cdot 7 - 1 \equiv 55 \equiv 9 \pmod{23}$ and $3^2 = 9$, so indeed $P \in \mathcal{E}(\mathbb{F}_{23})$.
   For Q: $22^3 + 5 \cdot 22 - 1 \equiv -1 - 5 - 1 = -7 \equiv 16 \pmod{23}$ and $4^2 = 16$, so $Q \in \mathcal{E}(\mathbb{F}_{23})$.

   (b) We have

$$\begin{aligned} \lambda &= \frac{3 \cdot 22^2 + 5}{2 \cdot 4} \equiv \frac{3 + 5}{8} \equiv 1 \pmod{23}; \\ x_{[2]Q} &= 1 - 2 \cdot 22 \equiv 1 + 2 \equiv 3 \pmod{23}; \\ y_{[2]Q} &= -4 + 1(22 - 3) \equiv 19 + 22 - 3 = 38 \equiv 15 \pmod{23}. \end{aligned}$$

   So $[2]Q = (3, 15)$.

   (c) We have

$$\begin{aligned} \lambda &= \frac{3 - 4}{7 - 22} \equiv \frac{-1}{-15} \equiv \frac{-1}{8} \equiv -1 \cdot 3 \equiv 20 \pmod{23}; \\ x_{P+Q} &= 20^2 - 7 - 22 \equiv 9 - 7 + 1 \equiv 3 \pmod{23}; \\ y_{P+Q} &= -3 + 20(7 - 3) \equiv -3 - 3 \cdot 4 = -15 \equiv 8 \pmod{23}. \end{aligned}$$

   Hence $P + Q = (3, 8)$.

   (d) Since $\#\mathcal{E}(\mathbb{F}_{23}) = 17$, we know by Lagrange's theorem that any point P on this curve has order 1 or 17. Since the only point of order 1 is the neutral element $\mathcal{O}$, any other point has order 17. Thus one may take any point, P, Q, $P + Q$, $[2]Q$ as generator, as long as one does not pick $\mathcal{O}$.

(e) The compressed point representation for P is given by $(7, 1)$, as y_P is odd.

[[Grading Instruction:

Grading (total 15):	
aspect:	points
(a)	**2**
each correct verification	1
(b)	**4**
λ, x_3, y_3 each	1
conclusion	1
(c)	**4**
λ, x_3, y_3	each 1
conclusion	1
(d)	**3**
Lagrange's Theorem's conclusion	1
neutral element is not OK, or any other point does not have order 1	1
any other point given	1
(e)	**2**
answer	1 or 2
explanation	1

Take into account "doorrekenfouten", but an error will cost a point.

Since there is no real explanation at (e) anyway, without explanation can also be good enough.

]]

8. (**10 points**) **Discrete logarithm.** Compute $\mathrm{dlog}_2(6)$ in $(\mathbb{Z}/23\mathbb{Z})^*$ (i.e., compute the smallest positive integer x such that $2^x \equiv 6 \pmod{23}$). You should solve this by using Pollard's ρ algorithm, or the baby-step giant-step (BSGS) algorithm. For the BSGS algorithm, you should take $m = \lfloor \sqrt{q} \rfloor$. [Hint: The order of 2 in $(\mathbb{Z}/23\mathbb{Z})^*$ is 11.]

Solution:
Pollard-ρ: For this method, we construct the following table where we have $a_i = g^{b_i} \cdot h^{c_i}$ following the function

$$(a_{i+1}, b_{i+1}, c_{i+1}) = \begin{cases} (a_i \cdot g, b_i + 1, c_i) & \text{if } a_i \equiv 1 \pmod 3; \\ (a_i \cdot h, b_i, c_i + 1) & \text{if } a_i \equiv 2 \pmod 3; \\ (a_i^2, 2b_i, 2c_i) & \text{if } a_i \equiv 0 \pmod 3. \end{cases}$$

In this case, we have $g = 2$ and $h = 6$. Assume moreover that we start with $b_0 = 1$ and $c_0 = 0$.

i	0	1	2	3	4	5	6
a_i	2	12	6	13	3	9	12
b_i	1	1	2	4	5	10	20
c_i	0	1	2	4	4	8	16

We see that $a_1 = a_{12}$ which implies $g^1 \cdot h^1 \equiv g^{20} \cdot h^{16} \pmod{23}$. This translates to $2^1 \cdot 6^1 \equiv 2^{20} \cdot 6^{16} \pmod{23}$, which in particular implies that $2^{-19} \equiv 6^{15} \pmod{23}$. Alternatively, rewriting

$g^1 \cdot h^1 \equiv g^{20} \cdot h^{16} \pmod{23}$ yields $h^1 \cdot h^{-16} \equiv g^{20} \cdot g^{-1} \pmod{23}$, which yields $h \equiv g^{\frac{20-1}{1-16}} \pmod{23}$. Since 2 (mod 23) has order 11, we have that

$$\begin{aligned} x &\equiv -19 \cdot 15^{-1} (\equiv 19 \cdot (-15)^{-1} \text{ if using the alternative notation}) \\ &\equiv 3 \cdot 4^{-1} \equiv 3 \cdot 3 \equiv 9 \pmod{11}. \end{aligned}$$

Hence $x = \text{dlog}_2(6) = 9$.

Baby-step giant-step: We have $q = 11$, so $m = \lfloor \sqrt{q} \rfloor = 3$. We then precompute the value $g_0 := g^{-m} = 2^{-3} \equiv 8^{-1} \equiv 3 \pmod{23}$. We get the table with the Baby-Steps:

i	0	1	2	3
2^i	1	2	4	8

and then the Giant-Steps:

j	0	1	2
$h \cdot g_0^j$	6	18	8

So we find that the solution is $x = \text{dlog}_2(6) = 3 + 3 \cdot 2 = 9$.

[[Grading Instruction:

Grading (total 10):	
Pollard's ρ:	points
Correctly stating/referring to the function for $(a_{i+1}, b_{i+1}, c_{i+1})$	**2**
Giving the correct table using the previous function until collision of a_i	**5**
Correct method of calculating $x = \text{dlog}_2(6)$ from the table	**2**
Correct answer of $\text{dlog}_2(6)$	**1**
	OR
Baby-Step Giant-Step:	points
Concluding that $m = \lfloor \sqrt{11} \rfloor = 3$	**1**
Calculating $g_0 \equiv 2^{-3} \equiv 3 \pmod{23}$ correctly	**2**
Giving correct Baby-Step table	**2**
Giving correct Giant-Step table	**3**
Correct calculation of $x = \text{dlog}_2(6)$	**2**

Marloes

zegt: 'Take into account "doorrekenfouten", but an error will cost a point.'?

]]

9. **(14 points) A composite protocol.** Let p be a large prime number of 3072 bits, and let $g \in (\mathbb{Z}/p\mathbb{Z})^*$ be an element of order q, where q is a large prime number of 256 bits. We assume that $\langle g \rangle$ provides 128 bits of security with respect to the decisional Diffie-Hellman problem. Consider the protocol in Figure 1, which is a composite of two protocols or schemes that we have treated in the course.

   (a) Which key establishment protocol or public-key encryption scheme is used? 2 pt

   (b) Give an alternative protocol or scheme for the one in (a) that can better be used instead. Note that your alternative needs to be IND-CPA-secure. 2 pt

Alice		Bob
$p, g, q, A = g^a, a$, (Bob: B)		$p, g, q, B = g^b, b$, (Alice: A)
$v \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$, $V \leftarrow g^v$		
$a' \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$, $A' \leftarrow g^{a'}$		
$C \leftarrow M \times B^{a'}$	$\xrightarrow{\text{Alice}, V, (C, A')}$	$c \xleftarrow{\$} \{0,1\}$
	$\xleftarrow{c}$	
$r \leftarrow v - ca \pmod q$	$\xrightarrow{r}$	$V \stackrel{?}{=} g^r \times A^c$
		$M' \leftarrow C \times (A')^{q-b}$

Figure 1: The composite protocol

(c) Which authentication protocol or signature scheme is used in the composite protocol? 2 pt

(d) Explain why the protocol or scheme in (c) might not provide sufficient security against attackers, who might want to impersonate Alice. 3 pt

(e) Explain how you can fix the protocol or scheme to provide sufficient security. 2 pt

(f) Explain what simple adjustments you can make to the protocol to ensure that the size of $V, (C, A')$ is reduced from $9216 = 3 \cdot 3072$ bits to $771 = 3 \cdot (256 + 1)$ bits without compromising the security strength of the scheme. [Hint: First, reduce from $9216 = 3 \cdot 3072$ bits to $1536 = 3 \cdot (256 + 256)$ bits. Then, further reduce to $771 = 3 \cdot (256 + 1)$ bits.] 3 pt

Solution:

(a) The ElGamal encryption scheme

(b) One could better use a key encapsulation mechanism, such as the KEM from ElGamal or the KEM from RSA.

(c) The Chaum-Evertse-van de Graaf protocol.

(d) Because you take challenges from $\{0, 1\}$, any cheating prover can simply guess which challenge $c' \in \{0, 1\}$ is going to be sent by Bob. She can then generate $r \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$ and set $V \leftarrow g^r A^{c'}$, and send this V to Bob. She has a probability of $\frac{1}{2}$ that Bob sends her $c = c'$, and thus of passing the check (with r).

(e) You can pick the challenges from $\mathbb{Z}/q\mathbb{Z}$ instead. This gives you the Schnorr authentication protocol, which is both sound and honest-verifier zero-knowledge. Specifically, cheating provers have a probability of only $1/q$ of guessing the challenge correctly, which is negligible. Alternatively, you can iterate the CEG protocol multiple times (i.e., at least 128 times to provide sufficient security). In this case, cheating provers have a probability of only $1/2^{128}$ of guessing each challenge correctly, which is negligible (with respect to the security strength).

(f) They can use elliptic-curve groups, which only require a field size of 256 bits for 128 bits of security. To ensure that $V, (C, A')$ can indeed be expressed in 771 bits, they need to use point compression.

[[Grading Instruction:

Grading (total 14):

aspect:	points
(a)	**2**
(b)	**2**
RSA-OAEP	only 1
(c)	**2**
(d)	**3**
cheating prover can guess c	1
correct with probability $\frac{1}{2}$	1
explain how V and r are generated	1
(e)	**2**
take challenge from $\mathbb{Z}/q\mathbb{Z}$/change to Schnorr	1
explanation why this is secure	1
Schnorr authentication protocol is sound	
probability of guessing c is only $1/q$	
OR	
iterate the protocol	1
at least 128 times	
explanation why this is secure	1
probability of guessing all challenges is only $1/2^{128}$	
(f)	**3**
use elliptic curve groups	1
field size of 256 bits for 128 bits of security	1
point compression	1

]]

Formula sheet of Introduction to Cryptography

1 Mathematical concepts

1.1 Euler's totient function

Let $n > 1$ be an integer such that $n = \prod_i p_i^{k_i}$, where p_i are distinct prime numbers and $k_i > 0$. Then $\varphi(n)$ is computed as

$$\varphi(n) = \varphi\left(\prod_i p_i^{k_i}\right) = \prod_i \varphi(p_i^{k_i}) = \prod_i (p^{k_i} - p^{k_i - 1}).$$

1.2 Square-and-multiply algorithm

Data: Integers a, d, n
Result: x with $x \equiv a^d \pmod{n}$
1. Write $d = (d_{k-1}d_{k-2}\cdots d_1 d_0)_2$;
2. $x \leftarrow 1$;
3. **for** $i = k-1$ *to* 0 **do**
   $x \leftarrow x^2 \bmod n$;
   if $d_i = 1$ **then**
      $x \leftarrow ax \bmod n$;
   end
end
4. **return** x.

1.3 CRT, specifically for RSA

Suppose that we want to solve a system of modular equations like

$$\begin{cases} x \equiv a_0 & \pmod{p}; \\ x \equiv a_1 & \pmod{q}. \end{cases}$$

A solution is $x = u_0 a_0 + u_1 a_1 \pmod{n}$, where $u_0 = (q^{-1} \bmod p) \cdot q$ and $u_1 = (p^{-1} \bmod q) \cdot p$.

Garner's method:

A solution is $x = a_1 + q \cdot ((a_0 - a_1 \bmod p) \cdot (q^{-1} \bmod p) \bmod p)$.

2 Security strength

(Computational) Security strength:

A cryptographic scheme offers security strength s if there are no attacks with $(M+N)/p < 2^s$ with N and M the adversary's (offline and online) resources and p the success probability.

Advantage:

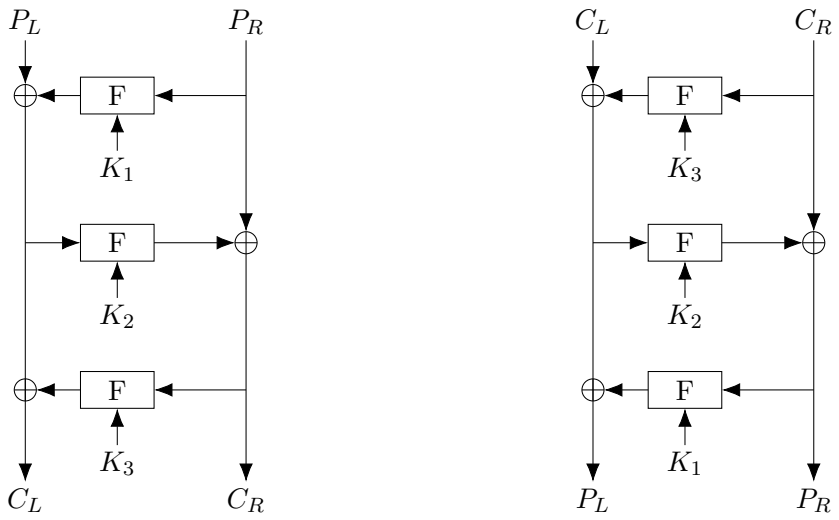
The advantage of distinguishing a stream cipher SC from a random oracle $\mathcal{RO}$ is given by: $\mathrm{Adv}_{\mathcal{A}} = |\Pr(\mathcal{A} = 1 \mid \mathrm{SC}_K) - \Pr(\mathcal{A} = 1 \mid \mathcal{RO})|$

(Decisional) Security strength:

A cryptographic scheme offers security strength s if there are no attacks with $(M+N)/\mathrm{Adv} < 2^s$ with N and M the adversary's (offline and online) resources and Adv the advantage of the adversary.

3 Symmetric cryptography

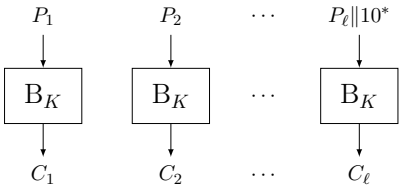
3.1 Feistel structure



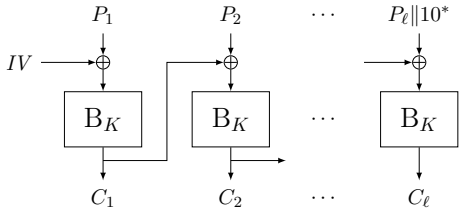
(a) Encryption (b) Decryption

Figure 1: Three-round Feistel structure.

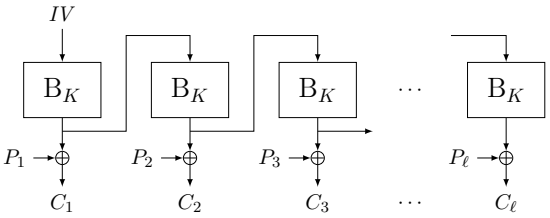
3.2 Block cipher modes



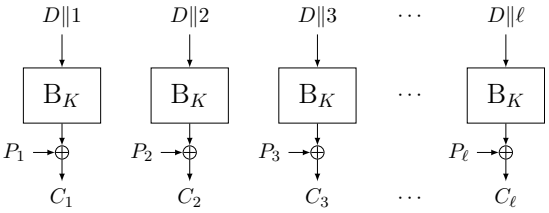
(a) Electronic codebook mode



(b) Cipher block chaining mode



(a) Output Feedback mode



(b) Counter mode

3.3 Hash function constructions

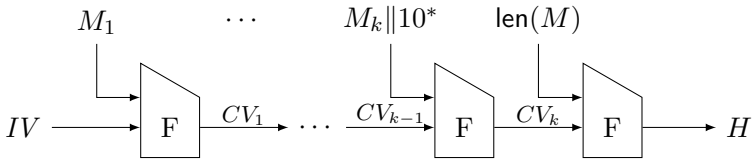


Figure 4: Merkle-Damgård construction for hash functions.

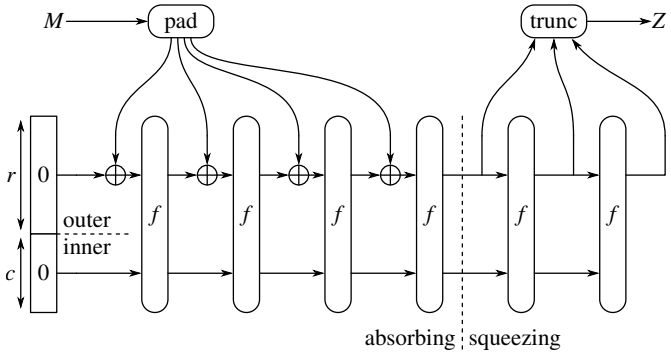


Figure 5: Sponge function.

4 Public-key cryptography

4.1 Key agreement schemes

4.1.1 Textbook (Merkle-)Diffie-Hellman key agreement

Alice		Bob
p, g, q		p, g, q
$a \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$		$b \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$
$A \leftarrow g^a$		$B \leftarrow g^b$
	$\xrightarrow{\text{Alice},A}$	
	$\xleftarrow{\text{Bob},B}$	
$K_{A,B} \leftarrow B^a$		$K_{B,A} \leftarrow A^b$

4.2 Encryption schemes

4.2.1 ElGamal encryption scheme

Alice		Bob
$p, g, (q), B$		$p, g, (q), b, B(= g^b)$
$a \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$		
$A \leftarrow g^a$		
$C \leftarrow M \times B^a$	$\xrightarrow{(C,A)}$	$M \leftarrow C \times A^{q-b}$

4.2.2 Textbook RSA encryption scheme

Bob		Alice
Alice's public key (n, e)		Alice's private key (n, d)
$c \leftarrow m^e \bmod n$	$\xrightarrow{c}$	$m \leftarrow c^d \bmod n$

4.3 Key encapsulation mechanisms (KEM)

4.3.1 KEM from ElGamal

Alice		Bob
$p, g, (q), B$		$p, g, (q), b, B(= g^b)$
$a \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$		
$A \leftarrow g^a$		
$K \leftarrow \mathrm{h}(\text{“KDF”}; B^a)$		
$CT \leftarrow \mathrm{Enc}_K(m)$	$\xrightarrow{(A,CT)}$	$K \leftarrow \mathrm{h}(\text{“KDF”}; A^b)$
		$m \leftarrow \mathrm{Dec}_K(CT)$

4.3.2 KEM from RSA

Bob has Alice's public key (n, e)		Alice with private key (n, d)
$r \xleftarrow{\$} \mathbb{Z}/n\mathbb{Z}$		
$c \leftarrow r^e \bmod n$		
$K \leftarrow \mathrm{h}(\text{“KDF”}; r)$		
$CT \leftarrow \mathrm{Enc}_K(m)$	$\xrightarrow{(c,CT)}$	$r \leftarrow c^d \bmod n$
		$K \leftarrow \mathrm{h}(\text{“KDF”}; r)$
		$m \leftarrow \mathrm{Dec}_K(CT)$

4.4 Authentication protocols

4.4.1 Chaum-Evertse-van de Graaf (CEG) protocol

Alice		Bob
p, g, q, A, a		p, g, q (Alice: A)
$v \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$		
$V \leftarrow g^v$	$\xrightarrow{\text{Alice},V}$	$c \xleftarrow{\$} \{0, 1\}$
	$\xleftarrow{c}$	
$r \leftarrow v - ca$	$\xrightarrow{r}$	$V \stackrel{?}{=} g^r A^c$

4.4.2 Schnorr's authentication protocol

Alice		Bob
p, g, q, A, a		p, g, q (Alice: A)
$v \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$		
$V \leftarrow g^v$	$\xrightarrow{\text{Alice},V}$	$c \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$
	$\xleftarrow{c}$	
$r \leftarrow v - ca$	$\xrightarrow{r}$	$V \stackrel{?}{=} g^r A^c$

4.5 Signature schemes

4.5.1 Schnorr's signature scheme

Alice		Bob
p, g, q, A, a		p, g, q (Alice: A)
$v \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$		
$V \leftarrow g^v$		
$c \leftarrow \mathrm{h}(p; g; A; V; m)$		
$r \leftarrow v - ca$	$\xrightarrow{\text{Alice},m,(r,V)}$	$c \leftarrow \mathrm{h}(p; g; A; V; m)$
		$V \stackrel{?}{=} g^r A^c$

4.5.2 Full-domain hash RSA signatures

Alice with private key (n, d)		Bob with Alice's public key (n, e)
$H \leftarrow \mathrm{h}(m)$		
$s \leftarrow H^d \bmod n$	$\xrightarrow{\text{Alice}, m, s}$	$H \leftarrow \mathrm{h}(m)$
		$H \stackrel{?}{=} s^e \bmod n$

4.5.3 Security notions

Discrete log (DL) problem:
Let $a \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$ and $A \leftarrow g^a$. Given $\langle g \rangle$ and A, determine a.

Computational Diffie-Hellman (CDH) problem:
Let $a, b \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$, $A \leftarrow g^a$ and $B \leftarrow g^b$. Given $\langle g \rangle$ and A, B, determine g^{ab}.

Decisional Diffie-Hellman (DDH) problem:
Let $a, b, c \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$, and $A \leftarrow g^a$, and $B \leftarrow g^b$. With probability $\frac{1}{2}$, set $C \leftarrow g^c$, and otherwise $C \leftarrow g^{ab}$. Given $\langle g \rangle$ and A, B, C, determine whether $C = g^{ab}$ holds.

Advantage:

The advantage of an adversary on the decisional Diffie-Hellman problem is given by:

$$\mathrm{Adv}_{\mathcal{A}} = |\Pr(\mathcal{A} = 1 \mid C = g^{ab}) - \Pr(\mathcal{A} = 1 \mid C = g^c)|$$

IND-CPA security:

Challenger		Adversary
$p, g, (q)$		$p, g, (q)$
$b \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$		
$PK \leftarrow g^b$	$\xrightarrow{PK}$	Repeat: $\mathrm{Enc}_{PK}(M)$
	$\xleftarrow{M_0, M_1}$	M_0, M_1 messages
$i \xleftarrow{\$} \{0, 1\}$		
$CT \leftarrow \mathrm{Enc}_{PK}(M_i)$	$\xrightarrow{CT}$	Repeat: $\mathrm{Enc}_{PK}(M)$

4.6 Elliptic curves

4.6.1 Addition formulas for Weierstrass curves over prime fields

An elliptic curve (in short Weierstrass form) is the set of points in $\mathbb{F}_p^2$ that satisfy

$$\mathcal{E}\colon y^2 = x^3 + ax + b, \qquad (a, b \in \mathbb{F}_p)$$

together with the point at infinity $\mathcal{O}$.

If points $P = (x_1, y_1)$, $Q = (x_2, y_2)$ are on curve $\mathcal{E}$, then we can compute their sum, $R = (x_3, y_3)$, algebraically as follows:

	$P = -Q$	$P \neq \pm Q$	$P = Q$
$R =$	$\mathcal{O}$	$\lambda = \frac{y_1 - y_2}{x_1 - x_2}$ $x_3 = \lambda^2 - x_1 - x_2$ $y_3 = -y_1 + \lambda(x_1 - x_3)$	$\lambda = \frac{3x_1^2 + a}{2y_1}$ $x_3 = \lambda^2 - 2x_1$ $y_3 = -y_1 + \lambda(x_1 - x_3)$

4.6.2 Projective coordinates

We can convert any point $(X : Y : Z)$ with $Z \neq 0$ to affine coordinates, as (XZ^{-1}, YZ^{-1}). The homogeneous elliptic curve has the form

$$Y^2 Z = X^3 + aXZ^2 + bZ^3.$$

The curve's point at infinity is $\mathcal{O} = (0 : 1 : 0)$.

4.7 Attacks on the discrete logarithm problem

We use multiplicative notation in the following. In additive notation, multiplications are replaced by additions and exponentiations by scalar multiplications.

4.7.1 Baby-step giant-step algorithm

Data: Group elements g, h and table size m
Result: Integer a such that $h = g^a$
1. $q \leftarrow \#\langle g \rangle$;
2. **for** $i = 0$ *to* m **do**
   $b_i \leftarrow g^i$;
end
3. $j \leftarrow 0$;
4. **repeat** $c_j \leftarrow h \cdot g^{-m \cdot j}$;
until $c_j = b_i$;
return $i + m \cdot j$.

4.7.2 Pollard's ρ algorithm

Let p be a prime number such that $g \in (\mathbb{Z}/p\mathbb{Z})^*$ has order q. We want to compute $x = \mathrm{dlog}_g(h)$ for some $h \in \langle g \rangle$. We take as starting point $(g, 1, 0)$ and as our function:

$$(a_{i+1}, b_{i+1}, c_{i+1}) = \begin{cases} (a_i \cdot g, b_i + 1, c_i) & \text{if } a_i \equiv 1 \pmod 3; \\ (a_i \cdot h, b_i, c_i + 1) & \text{if } a_i \equiv 2 \pmod 3; \\ (a_i^2, 2b_i, 2c_i) & \text{if } a_i \equiv 0 \pmod 3. \end{cases}$$

Note that we take computations on a_i modulo p, and computations on b_i and c_i modulo q.

When we find $i \neq j$ with $a_i = a_j$, then we have

$$g^{b_i} h^{c_i} \equiv g^{b_j} h^{c_j} \pmod p,$$

so we get

$$g^{b_i - b_j} \equiv h^{c_j - c_i} \equiv g^{x(c_j - c_i)} \pmod{p}.$$

We then find x by solving $b_i - b_j \equiv x(c_j - c_i)$ modulo q.