DD2448 Foundations of Cryptography (krypto20)
Exercises to prepare for the homework

Douglas Wikström, dog@kth.se

March 10, 2020

Instructions

These problems are meant to be used for self-study within your study group to prepare for the homework, but no solutions are handed in. The problems are given in no particular order, but the number of points assigned to a problem is an estimate of its difficulty or the amount of work involved to solve it.

Some problems may appear as part of the homework, possibly in slightly modified form and with slightly different number of points. The rules for the homework applies. You may discuss the problems informally within your group, e.g., with the help of a blackboard, but you may not share written up solutions (not even hand written notes).

If you think that you have found an error, or if you believe that information is missing to solve a problem, then download the most recent version of this document and see if it has already been corrected. If this does not help, then please email dog@kth.se with subject `DD2448` and explain the issue to help us improve this document.

1 Ciphers

Problem 1 (5T). Your friend is a high school teacher and you are asked to visit his or her class and talk about cryptography. In preparation for your visit, the students are asked to construct their own ciphers. You may assume that the students have no knowledge of cryptography and that none of them is a genius.

Your job is to write a program that can break their ciphers. Please describe what assumptions you make and how your program works in plain English (not using pseudo-code). Your solution is graded based on completeness, clarity, and conciseness.

Problem 2 (6T). Read the paper of Khazaei and Ahmadi on the Hill cipher `https://eprint.iacr.org/2015/802.pdf`. Describe in detail what kinds of attacks you can mount on the Hill cipher. Is it a known-plaintext, chosen-plaintext, etc attack? How many ciphertexts do you need? What is the approximate time complexity of the attacks?

Problem 3. Recall the AES block cipher with n rounds. It encrypts a block specified as a 4×4 matrix P of bytes and a sequence of matrices $W = (W_0, \ldots, W_n)$ as subkeys, derived from the secret key using a key schedule. The AES encryption function, $\mathsf{AES}(P, W)$, is then given by the following pseudo-code.

```
1. $P = \mathsf{AddRoundKey}(P, W_0)$
```

2. For $r = 1, \dots, n-1$ do:
   (a) $P = \mathsf{SubBytes}(P)$
   (b) $P = \mathsf{ShiftRows}(P)$
   (c) $P = \mathsf{MixColumns}(P)$
   (d) $P = \mathsf{AddRoundKey}(P, W_r)$
3. $P = \mathsf{SubBytes}(P)$
4. $P = \mathsf{ShiftRows}(P)$
5. Return $\mathsf{AddRoundKey}(P, W_n)$

where $\mathsf{AddRoundKey}(P, W_i) = P \oplus W_i$ and SubBytes, ShiftRows, and MixColumns are invertible operations on 4×4 matrices. Denote the inverse of the three transformations by InvSubBytes, InvShiftRows, and InvMixColumns.

The linear transformation $\mathsf{MixColumns}(P) = MP$, where M is the following matrix and the operation is defined over $GF(2^8)$.

$$M = \begin{pmatrix} \texttt{0x02} & \texttt{0x03} & \texttt{0x01} & \texttt{0x01} \\ \texttt{0x01} & \texttt{0x02} & \texttt{0x03} & \texttt{0x01} \\ \texttt{0x01} & \texttt{0x01} & \texttt{0x02} & \texttt{0x03} \\ \texttt{0x03} & \texttt{0x01} & \texttt{0x01} & \texttt{0x02} \end{pmatrix}$$

Each byte in M represents a polynomial in $\mathbb{Z}_2[z]$ of degree at most 7 with coefficients in $\mathbb{Z}_2$. More precisely, a byte $(a_7, \dots, a_0) \in \{0,1\}^8$ represents the polynomial $\sum_{i=0}^{7} a_i z^i$. The field $GF(2^8)$ is represented in polynomial basis with the irreducible polynomial $f(z) = z^8 + z^4 + z^3 + z + 1$, i.e., $GF(2^8)$ is represented by $F = \mathbb{Z}_2[z]/f(z)$.

Task 3.1 (2T). Provide pseudo-code for AES decryption, $\mathsf{AES}^{-1}(C, W)$, in terms of the above subroutines.

Task 3.2 (2T). Compute $\texttt{0x53} + \texttt{0xb8}$, $\texttt{0x21} \times \texttt{0x25}$ and inverse of $\texttt{0x02}$ in F.

Task 3.3 (2T). Compute M^{-1} using Gaussian elimination, i.e., the matrix that is used to implement $\mathsf{InvMixColumns}(P)$.

Problem 4. Let $E^t : \{0,1\}^n \times \{0,1\}^{tn} \to \{0,1\}^n$ be an n-bit block cipher with tn-bit keys, consisting of a t-round Feistel network. Let "$\|$" denote concatenation and let f_i be the ith Feistel function. Then denote the key by $k = k_1 \| k_2 \| .. \| k_t$, the plaintext by $L_0 \| R_0 \in \{0,1\}^n$, and the output in round $s \geq 1$ by $L_s \| R_s$, i.e., the output ciphertext is $L_t \| R_t$. Assume that $f_i(k_i, \cdot)$ is pseudo-random function for a random k_i.

Task 4.1 (2T). Show that if $t = 1$, then the Feistel network is not a pseudorandom permutation.

Task 4.2 (4T). Show that if $t = 2$, then the Feistel network is not a pseudorandom permutation.

Task 4.3 (10T). Show that if $t = 3$, then the Feistel network is not a pseudorandom permutation. (Hint: Look at several related inputs and outputs. Evaluate the permutation as well as its inverse on these.)

Problem 5. Much of cryptographic theory is based on computational assumptions and the security of a cipher is effectively a cryptographic assumption. Although we could all use our own ciphers, interoperability would be a nightmare, and some protocols require jointly trusted assumptions.

Read the paper at `https://arxiv.org/pdf/1702.06475.pdf` which is an attempt to construct a cipher that appears to be secure even to experts in symmetric cryptography, but contains a backdoor that can be used by the designer to decrypt without the secret key. Go through the slides for their talk at `http://ubm.io/2E0k5Cj` and pay special attention to the history section.

ISO reviewed a proposal to adopt the light-weight ciphers Speck and Simon proposed by NIST into a standard. Read about the controversy online.

Task 5.1 (6T). Summarize the claims made in the paper in your own words in a few sentences.

Task 5.2 (3T). Summarize the public discussion regarding Speck and Simon in your own words in a few sentences. How are the objections phrased? Are they technical in nature or political? How have the designers responded?

Task 5.3 (3T). Try to formulate decision making criteria that suffices for us to be convinced that a cipher is secure in demanding applications. Who should decide? Whom should we trust? Why? Does it differ depending on who you represent (state, company, NGO, individual)?

---

2 Hash Function

Problem 6. Consider SHA-256 to be a random oracle in a practical application.

Task 6.1 (1T). Construct an (almost) random oracle $\{0,1\}^* \rightarrow \{0,1\}^{3000}$ based on SHA-256.

Task 6.2 (2T). What is the collision resistance of your function?

Task 6.3 (1T). Explain how to solve this problem using SHA-3/Keccak.

---

3 Information Theory

Problem 7. Denote by X a random variable with probability function $\mathsf{P}_X : \mathbb{Z}_{12} \rightarrow [0,1]$ defined by the following table.

x	0	1	2	3	4	5	6	7	8	9	10	11
$\mathsf{P}_X(x)$	0.12	0.07	0.02	0.05	0.10	0.13	0.14	0.11	0.08	0.04	0.09	0.05

Task 7.1 (2T). Compute the binary Hoffman code for P_X and derive how far it is from theoretical lower bound on any binary code for P_X.

Task 7.2 (2T). Compute the ternary Hoffman code for P_X and derive how far it is from theoretical lower bound on any **ternary** code for P_X.

Task 7.3 (1T). Is there a difference? If so, then explain informally why this is so.

Problem 8. In each case below, say as much as you can about the entropy of Y and motivate your answers. Make sure that you **do not assume anything about the distribution of Y that is not stated explicitly**. More precisely, for each description of the random variable Y given below, explain if, why, and how, the information given about Y:

1. is sufficient/insufficient to compute the entropy of Y,
2. allows you to give a closed expression[1] of the entropy of Y, or
3. only allows you to bound the entropy of Y from above and/or below.

(Possibly in terms of the entropies of X and S.)

Task 8.1 (1T). Let $Y = (X_1, \ldots, X_n)$ be a random variable over $\{0,1\}^n$ such that $\Pr[X_i = 1] = 1/3$ for $i = 1, \ldots, n$.

Task 8.2 (1T). Let S be a uniformly distributed random variable over $\{0,1\}^{256}$ and define $Y = \mathsf{SHA256}(S)$.

Task 8.3 (1T). Let S be a random variable over $\{0,1\}^{256}$ and define $Y = \mathcal{RO}(S)$, where $\mathcal{RO} : \{0,1\}^{256} \rightarrow \{0,1\}^{256}$ is a random oracle.

Task 8.4 (1T). Let q be an odd prime and let $H = \{H_k\}_{k \in \mathcal{K}}$ be a universal$_2$ hash function such that $H_k : \mathbb{Z}_q \rightarrow \mathbb{Z}_q$ for every k, let X be uniformly distributed over the set of keys $\mathcal{K}$, and define $Y = \big(H_X(1), H_X(2)\big)$.

Task 8.5 (2T). Let $Y = (X_0, \ldots, X_n)$ be a random variable over $\{0,1\}^n$ such that $X_0 = 1$ and for every $(x_1, \ldots, x_{i-1}) \in \{0,1\}^{i-1}$ we have $\Pr[X_i = X_{i-1} | (X_1, \ldots, X_{i-1}) = (x_1, \ldots, x_{i-1})] = 2^{-i}$ for $i = 1, \ldots, n$.

Task 8.6 (2T). Let f be a function, let X be a random variable over a set $\mathcal{X}$, and define $Y = f(X)$. Only the probability function $\mathsf{P}_X(x)$ of X is given, not the one for Y.

Task 8.7 (2T). Let f be a function, let Y be a random variable over a set $\mathcal{Y}$, and define $X = f(Y)$. Only the probability function $\mathsf{P}_X(x)$ of X is given, not the one for Y.

---

4 Provable Security

Problem 9 (2T). The standard definition of an efficient algorithm in complexity theory is an algorithm with polynomial running time (if it is uniform or not does not matter in this problem). Why is this notion unsatisfactory when constructing algorithms for optimization problems, but it is rarely unsatisfactory when modeling adversaries in cryptography?

Problem 10. Search for information about uniform and non-uniform adversaries.

Task 10.1 (1T). Describe the difference in your own words.

Task 10.2 (2T). Does it matter which view we take on efficient adversaries? (Both in theory and practice.) Are they equivalent?

Problem 11 (3T). Suppose that $f(\cdot)$ is not negligible. If there exist integers $c, n_0 > 0$ such that $f(n) > n^{-c}$ for all $n > n_0$, then prove this and otherwise give a counter example.

Problem 12. Motivate the definition of negligible functions. Let $l(n)$ be a polynomial and let $\epsilon(n)$ be a negligible function.

[1] `https://en.wikipedia.org/wiki/Closed-form_expression`

Task 12.1 (2T). Prove that $l(n)\epsilon(n)$ is negligible.

Task 12.2 (1T). Consider a sequence of binary random variables $X_1, \ldots, X_{l(n)}$ such that $\Pr[X_i = 1] \leq \epsilon(n)$. Prove the smallest upper bound you can on the probability that $\sum_{i=1}^{l(n)} X_i > 0$.

Task 12.3 (2T). Suppose that you use a cryptographic protocol based on the n-bit RSA cryptosystem and that you have proved that some fatal event occurs with probability at most $\epsilon(n)$.

A common security parameter for RSA is $n = 2048$, since factoring 2048-bit RSA moduli is considered hard. Should you worry about the risk of the fatal event if you set $n = 2048$ when run your protocol? (Motivate your answer.)

Problem 13 (2T). List the indices of the functions below that are negligible functions. For example, if you think $f_1(n)$ and $f_2(n)$ are negligible and no other, then your answer should simply be "$1, 2$".

$$f_1(n) = n^{-\log n} \quad f_2(n) = n^{-256} \quad f_3(n) = 2^{-n} \quad f_4(n) = n^{-128n} \quad f_5(n) = f_2(n) + f_4(n)$$

To get any points your answer must be completely correct, i.e., this is an all-or-nothing problem. You do *not* need to motivate your answer for this problem.

Problem 14 (2T). Describe the structure of a proof by reduction in cryptography as explained in class, i.e., describe the roles of definitions, assumptions, reductions, parties, adversaries, and conclusions.

A fellow student that does not follow the course should be able to understand your description. You may show your description to somebody that does not follow the course to check this! Feel free to use handmade illustrations, but please use A4 paper that the copying machine can handle.

Problem 15. The goal of this problem is that you write out the details of a proof of security in cryptography on your own. We have already covered this result in class. Let $\mathsf{CS} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public key cryptosystem. More precisely:

- Gen is a probabilistic key generation algorithm that on input 1^n (security parameter n in unary representation) outputs a key pair. We denote this by $(pk, sk) = \mathsf{Gen}(1^n)$.
- Enc is an encryption algorithm that takes a public key pk, a message $m \in \{0,1\}^n$, and randomness $r \in \{0,1\}^n$ as input and produces a ciphertext. We denote this by $\mathsf{Enc}_{pk}(m, r)$.
- Dec is a decryption algorithm that takes a secret key sk and a ciphertext c as input and outputs the plaintext. We denote this by $m = \mathsf{Dec}_{sk}(c)$.

Denote by $\mathsf{CS}^k = (\mathsf{Gen}^k, \mathsf{Enc}^k, \mathsf{Dec}^k)$ the cryptosystem defined as follows:

- Gen^k is identical to Gen
- Enc^k takes a public key pk, a message $m \in \{0,1\}^{n \times k}$, and randomness $r \in \{0,1\}^{n \times k}$ as input and outputs $(\mathsf{Enc}_{pk}(m_1, r_1), \ldots, \mathsf{Enc}_{pk}(m_k, r_k))$.
- Dec^k takes a secret key sk and a ciphertext $c = (c_1, \ldots, c_k)$ as input and outputs a plaintext $(\mathsf{Dec}_{sk}(c_1), \ldots, \mathsf{Dec}_{sk}(c_k))$.

Task 15.1 (7T). Prove that if CS is CPA secure, then CS^2 is CPA secure.

Task 15.2 (3T). Prove that for every polynomial $k(n)$, if CS is CPA secure, then $\mathsf{CS}^{k(n)}$ is CPA secure.

You need to be **more rigorous** than what we did in class that! Imagine that your life depended on convincing your worst enemy.

---

5 Elliptic Curves

Problem 16. The generic elliptic curves covered in class uses separate code for doubling, adding, and treatment of the point at infinity. For some curves this is not necessary, i.e., there is no need for special code.

Task 16.1 (3T). Dan Bernstein has published several papers about this. Read enough to be able to explain the key ideas.

Task 16.2 (2T). What are the advantages of such curves in practice?

Problem 17. Read about the *Dual Elliptic Curve Deterministic Random Bit Generator* proposed by NIST in the original document `http://csrc.nist.gov/publications/nistpubs/800-90A/SP800-90A.pdf`. Read other sources you find online as well.

Task 17.1 (1T). Briefly summarize the controversy regarding this PRG.

Task 17.2 (1T). Why do you think it was obvious to most researchers that something was not right with this construction even before the backdoor was made public?

Task 17.3 (2T). More generally it is worthwhile to consider how a good elliptic curve is chosen. What is the purpose of the *million dollar curve* and what is special about how it is generated?

Problem 18. Read about Curve25519 online and then listen to the talk of Yarom and Genkin at YouTube from RWC 2016: `https://www.youtube.com/watch?v=mcEHVvcqUzU`.

Task 18.1 (2T). Explain how Curve25519 is different from popular standardized curves such as P-256 with respect to side-channel attacks?

Task 18.2 (1T). For what kinds of applications does this difference matter?

Task 18.3 (3T). Yarom's and Genkin's attack illustrates that mastering real-world security today requires a deep understanding of both theory and hardware. Summarize their attack at a high level.

Problem 19. You are given a non-singular elliptic curve over a prime order field $\mathbb{Z}_q$ on Weierstrass normal form, i.e., $E : y^2 = f(x)$, where $f(x) = x^3 + ax + b$.

Task 19.1 (2T). Construct an efficiently computable invertible injection $\{0,1\}^k \to E$, i.e., describe: (1) an algorithm Encode that takes a bitstring as input and outputs an element in the curve, and (2) an algorithm Decode that takes a group element and outputs a bitstring.

Task 19.2 (2T). Prove that your construction satisfies $\mathsf{Decode}(\mathsf{Encode}(m)) = m$ for all $m \in \{0,1\}^k$ and explain how big you can make k relative to q.

Task 19.3 (1T). It turns out that you may need to allow your encoding algorithm to be probabilistic, so it suffices to prove (under reasonable assumptions) that the *expected* running time (over the randomness of your algorithms) for any fixed input is bounded by a polynomial in $\log q$. What is the polynomial?

---

6 Related to RSA Cryptosystem

Definition 1. The **RSA** assumption states that if $N = pq$, where p and q are randomly chosen primes with the same number of bits, $e \in \mathbb{Z}^*_{\phi(N)}$, and g is randomly chosen in $\mathbb{Z}^*_N$, then for every polynomial time algorithm A, $\Pr[A(N, e, g) = \beta \wedge \beta^e = g \bmod N]$ is negligible.

Definition 2. The **Strong RSA** assumption states that if $N = pq$, where p and q are randomly chosen primes with the same number of bits and g is randomly chosen in $\mathbb{Z}^*_N$, then for every polynomial time algorithm A, $\Pr[A(N, g) = (e, \beta) \wedge \beta^e = g \bmod N \wedge e > 1]$ is negligible.

Problem 20. Consider the hash function defined as follows. Let $N = pq$ where p and q are randomly chosen safe primes of the same bit-size, and let g be randomly chosen in $\mathbb{Z}^*_N$ with order $(p-1)(q-1)/4$. Then define $h_{N,g} : \mathbb{N} \to \mathbb{Z}^*_N$, $h_{N,g}(x) = g^x \bmod N$.

Task 20.1 (4T). Prove that a multiple of $(p-1)(q-1)/4$ can be computed from a collision when the primes are distinct.

Task 20.2 (2T). Use this fact to prove that the hash function is collision-resistant under the strong RSA assumption.

Problem 21. Suppose you need to generate an n-bit prime p_0 of the form $p_0 = 2p_1 + 1$, where $p_1 = 2p_2 + 1$ and p_2 are primes.

Task 21.1 (5T). Describe an algorithm and perform a heuristic analysis of its expected running time. Implement your algorithm and compare your practical results with your theoretical analysis.

Task 21.2 (2T). What happens if we relax the problem and replace 2 in each equality by integers $1 < k_1, k_2 < 2^{\sqrt{n}}$ of your own choice (i.e., you must find any p_0, p_1, p_2 and k_1, k_2)?

Problem 22. In class we considered the RSA signature scheme, i.e., RSA with full domain hash. In this problem we develop a different scheme based on the strong RSA assumption. Our construction is similar to some efficient *provably secure* signature schemes, but we only consider a simplified scheme and analyze its security in the random oracle model.

The private key of our scheme consists of two random $n/2$-bit safe[2] primes p and q. The public key consists of the modulus $N = pq$ and a random element g from the subgroup QR_N of quadratic residues in $\mathbb{Z}_N$. Suppose that $H : \{0,1\}^* \to \mathcal{P} \cap \{0,1\}^{n/3}$ is a random oracle, where $\mathcal{P}$ denotes the set of odd primes. A signature s of a message m is computed as $s = g^{1/H(m)} \bmod N$, where $1/H(m)$ should be understood as $H(m)^{-1} \bmod \frac{1}{2}(p-1)(q-1)$. To verify a signature s, one simply checks that $s^{H(m)} \bmod N = g$.

Task 22.1 (1T). For a standard RSA modulus we do not require that p and q are safe. Prove that this does not change the hardness of factoring N in any essential way. Hint: Use the prime number theorem to estimate heuristically the probability that a randomly chosen prime is safe by chance.

Task 22.2 (1T). Prove that if the strong RSA assumption holds, then the standard RSA assumption holds. (The opposite direction is unknown.)

Task 22.3 (1T). Prove that the signature scheme is correct, i.e., that $(g^{1/H(m)} \bmod N)^{H(m)} \bmod N = g$ for every message m.

[2]A prime p is safe if $(p-1)/2$ is prime as well.

Task 22.4 (1T). Let $p_1, \ldots, p_k \in \mathcal{P} \cap \{0,1\}^{n/3}$ be primes and let $g' \in \mathsf{QR}_N$ be randomly chosen. Prove that if we define $g = (g')^{\prod_{i=1}^k p_i} \bmod N$, then g is randomly distributed in QR_N. Thus, given a random element g' we can construct another random element g of which we can take any p_ith root modulo N.

Task 22.5 (1T). Suppose that there exists a polynomial time algorithm A such that for random keys $(pk, sk) = ((N, g), (p, q))$ and random H,

$$\Pr[A^{\mathsf{Sign}_{sk}(\cdot),H(\cdot)}(pk) = (m, s) \wedge \mathsf{Verify}_{pk}(m, s) = 1 \wedge \forall i : m_i \neq m] \geq \delta \enspace ,$$

where m_i is the ith query to the signature oracle $\mathsf{Sign}_{sk}(\cdot)$ and δ is non-negligible, i.e., A breaks the signature scheme. (In the literature the random oracle is often implicit. Here we make it explicit.)

Prove that without loss of generality we may assume that A never asks the same query twice and that it always evaluates the random oracle H on the message m of its output.

Task 22.6 (1T). Use the above to prove that given a random RSA modulus N and a random element $g' \in \mathsf{QR}_N$ you can generate a public key $pk = (N, g)$ such that you can simulate (without the secret key sk) a signature oracle $\mathsf{Sign}'(\cdot)$ and a random oracle $H(\cdot)$ such that

$$\Pr[A^{\mathsf{Sign}'(\cdot),H(\cdot)}(pk) = (m, s) \wedge \mathsf{Verify}_{pk}(m, s) = 1 \wedge \forall i : m_i \neq m] \geq \delta - \epsilon \enspace ,$$

where m_i is the ith query to the "signature oracle" $\mathsf{Sign}'(\cdot)$ and ϵ is exponentially small.

Task 22.7 (1T). Prove that if A has polynomial running time $T(n)$ and j is randomly chosen in $\{1, 2, \ldots, T(n)\}$, then

$$\begin{aligned}&\Pr[A^{\mathsf{Sign}'(\cdot),H(\cdot)}(pk) = (m, s) \wedge \mathsf{Verify}_{pk}(m, s) = 1 \wedge \forall i : m_i \neq m \wedge m = m'_j]\\ &\geq \delta/T(n) - \epsilon'\end{aligned}$$

where m_i is the ith query to the signature oracle and m'_j is the ith query to the random oracle, and ϵ' is exponentially small.

Task 22.8 (2T). Let N be an RSA modulus, let $g' \in \mathsf{QR}_N$, and define $g = (g')^{\prod_{i \neq j} p_i} \bmod N$. Prove that if (m, s) satisfies $\mathsf{Verify}_{pk}(m, s) = 1$ and $H(m) = p_j$, then we can find integers a and b such that $ap_j + b\prod_{i \neq j} p_i = 1$ and construct (β, ρ) such that $\beta^\rho \bmod N = g'$ and $\rho > 2$.

Task 22.9 (2T). Use the above observations to describe an algorithm A' that runs A as a subroutine and breaks the strong RSA assumption, i.e., A' takes an RSA modulus N and a random element $g' \in \mathsf{QR}_N$ as input and must use A to output (β, ρ) such that $\beta^\rho \bmod N = g'$ and $\rho > 2$.

Task 22.10 (2T). Suppose that we only wish to sign a polynomial $h(n)$ number of distinct messages known in advance (we can think of the messages as the integers $1, \ldots, h(n)$). Can you modify the signature scheme for this setting and prove its security without the random oracle?

Problem 23 (2T). Let p and q be distinct odd primes greater than five such that $(p-1)/2$ and $(q-1)/2$ are prime and define $N = pq$. What is the order of the largest cyclic non-trivial proper subgroup of $\mathbb{Z}_N^*$?

Problem 24. Recall that the fundamental theorem of finite abelian groups says that any finite commutative group is isomorphic to a direct sum of groups of prime power orders and let $N = 120$. We may make the sum canonical by ordering terms by prime and then power.

Task 24.1 (2T). Give the direct sum of subgroups isomorphic to $\mathbb{Z}_N$.

Task 24.2 (2T). Give the direct sum of subgroups isomorphic to $\mathbb{Z}_N^*$.

Problem 25 (2T). We denote the Legendre/Jacobi symbol of a modulo b by $\left(\frac{a}{b}\right)$. For each of the following symbols: (1) determine if the symbol is defined on the given inputs, (2) state if it is a Legendre or Jacobi symbol, and (3) *compute the symbol by hand* (we want to see some of your intermediate results) if possible: $\left(\frac{15}{28}\right)$, $\left(\frac{19}{357}\right)$, $\left(\frac{39}{33}\right)$, $\left(\frac{598120457575754}{75456831}\right)$, and $\left(\frac{57384}{53475546935698}\right)$.

Problem 26 (5T). Read the paper by Lenstra et al. `http://eprint.iacr.org/2012/064.pdf`. Summarize on roughly a page the findings in this paper in your own words. Make sure that you do not submit a solution unless you actually read the paper.

---

7 Discrete Logarithms

Problem 27. The goal of this problem is to *prove* the following implications covered in class. In other words, the difficulty in solving this problem is not understanding that the implications hold, but to write down a *rigorous* proof. Thus, in this particular problem, any handwaving give zero points. A proof consists of a description of an efficient reduction and a mathematical analysis thereof.

Task 27.1 (2T). Prove that the DH assumption implies the DL assumption.

Task 27.2 (2T). Prove that the DDH assumption implies the DH assumption.

Problem 28. Let $p = kq + 1$ and q be primes such that $\log q = n$, $\log k = n$ and such that the bit size of every prime factor of k is bounded by $\log n$. Let g be a generator of the unique subgroup of $\mathbb{Z}_p^*$ of order q. I pick $x \in \mathbb{Z}_q$ randomly and hand you $y = g^x$. Then you may ask me any number of questions of the form $u \in \mathbb{Z}_p^*$, which I answer by $u^x \bmod p$.

Task 28.1 (2T). Explain how you can compute x efficiently (describe your algorithm and analyze its running time).

Task 28.2 (1T). What is the important lesson to learn from this example? (This was mentioned in class, but you will not find it on any slides.)

Task 28.3 (1T). How would you address this problem in an implementation of the protocol?

---

8 Signature Schemes

Problem 29. Read about Lamport's one-time signatures.

Task 29.1 (2T). Define the key generation, signature, and verification algorithms.

Task 29.2 (2T). How many signatures of randomly chosen messages computed using the same secret key do you need to recover it? Describe your attack and prove that it works with large probability.

---

9 Protocols and Applications

Problem 30 (4T). In some applications side channel attacks are a concern. Describe what a side channel attack is. Find as many side channels as possible that as been exploited in the research literature. Cite each relevant paper properly in your answer with a brief description in a few sentences.

Problem 31. You are given an odd n-bit composite modulus N and two elements $g, y \in \mathbb{Z}_N^*$, where $y = g^x \bmod N$ and $x \in [0, 2^n - 1]$, and then we execute the following protocol:

1. I choose $r \in [0, 2^{3n} - 1]$ randomly and compute $\alpha = g^r \bmod N$.
2. You choose a challenge $c \in [0, 2^n - 1]$ and hand it to me.
3. I reply by $d = xc + r$ (computed over the integers without any modular reduction).
4. You accept if $y^c \alpha = g^d \bmod N$.

In class we considered a similar protocol executed over a group of prime order and argued that this is a so called proof of knowledge.

Task 31.1 (3T). Describe what goes wrong if we use the same analysis for the above protocol?

Task 31.2 (1T). Can you still say something useful about this protocol? Points for interesting ideas!

Problem 32 (8T). Do a literature study of the (in)security of the WEP protocol. Summarize on roughly a page[3] the findings in this paper in your own words.

Summarizing WEP and issues with it in one page is challenging. Both your choice of content (5T) and the presentation (3T) is judged. Thus, choose carefully what you present and make sure you have time to proof read and revise your text before submitting.

Problem 33. Do a literature study of the (in)security of the SSL/TLS protocol (through the years).

Task 33.1 (5T). Summarize in roughly a page your findings in your own words. Summarizing SSL/TLS and issues with it in one page is challenging. Choose carefully what you present and how you present it.

Task 33.2 (2T). In class we considered Diffie-Hellman key exchange and noted that we can derive a key for AES, and this is essentially the secure channel functionality provided by SSL/TLS, so why is it so complex? (mention two major reasons)

Problem 34. The goal of this problem is to study the OpenSSL source code to get feeling for what real world code for cryptography can look like.

Task 34.1 (1T). Identify and report the path to the file containing the optimized implementation of P-256.

Task 34.2 (1T). Determine if any security critical bugs have been fixed in this code since it was committed to the code base.

Task 34.3 (3T). Describe the techniques used in the implementation of P-256 to counter side channel attacks.

[3]Use your own judgement to figure out what a page is.

Problem 35. ECRYPT II Yearly Report on Algorithms and Keysizes `https://www.ecrypt.eu.org/csa/documents/D5.4-FinalAlgKeySizeProt.pdf` discusses the security levels of various cryptographic primitives.

Task 35.1 (4T). Give a one-page high level summary of the approach used to derive the recommended key sizes. Your target audience are fellow students that did not solve this problem.

Task 35.2 (2T). Argue in the same way as the ECRYPT II project and fill in additional rows in Table 7.2 and Table 7.3. You must motivate your solution.

Task 35.3 (1T). What do you think about the future of cryptography based on factoring versus cryptography based on the discrete logarithm assumption in elliptic curve groups?

10 Pseudo-random Functions and Generators

Problem 36 (4T). You are given a pseudo-random function $F_n = \{f_{n,\gamma}\}_{\gamma \in \Gamma_n}$, where $n \in \mathbb{N}$ is the security parameter and Γ_n is a set of possible keys for the security parameter n. Suppose that $f_{n,\gamma} : \{0,1\}^n \rightarrow \{0,1\}^{\log n}$ for every $\gamma \in \Gamma_n$. Can you construct a pseudorandom function $F'_n = \{f'_{n,\gamma}\}_{\gamma \in \Gamma'_n}$ such that $f'_{n,\gamma} : \{0,1\}^n \rightarrow \{0,1\}^n$? Prove that it works in that case, or explain informally why you think it is not possible if you think it is not possible.

Problem 37 (4T). You are given a pseudo-random generator such that $\mathsf{PRG} : \{0,1\}^n \rightarrow \{0,1\}^{n+1}$ for every security parameter $n \in \mathbb{N}$. Construct a pseudo-random function PRG' such that $\mathsf{PRG}' : \{0,1\}^n \rightarrow \{0,1\}^{2n}$ for every $n \in \mathbb{N}$, and prove that it is a pseudo-random generator.

Problem 38 (4T). You are given a pseudo-random generator such that $\mathsf{PRG} : \{0,1\}^n \rightarrow \{0,1\}^{2n}$ for every security parameter $n \in \mathbb{N}$. Construct a pseudo-random function $F_n = \{f_{n,\gamma}\}_{\gamma \in \Gamma_n}$ such that $f_{n,\gamma} : \{0,1\}^{\log n} \rightarrow \{0,1\}^n$, where $n \in \mathbb{N}$ is the security parameter and Γ_n is a set of possible keys for the security parameter n, and prove that it is a pseudo-random function.

Problem 39. We consider how randomness for cryptographic use is generated in the real world.

Task 39.1 (1T). Investigate how randomness for cryptographic use is generated for software written in JavaScript in at least two open source browsers, and: (1) include a link to the code that does this, (2) explain briefly the cryptographic construction, and (3) write a minimal example of how to use it. (It does not have to be executable code, a snippet suffices.)

Task 39.2 (1T). Investigate how randomness for cryptographic use is generated for software written in OracleJDK, and: (1) include a link to the code that does this, (2) explain briefly the cryptographic construction, and (3) write a minimal example of how to use it. (It does not have to be executable code, a snippet suffices.)