DD2448 Foundations of Cryptography (krypto24)
Homework

Douglas Wikström, dog@kth.se

May 18, 2024

Abstract

Make sure that you read and understand `Files`→`Homework/solution_rules.pdf` at Canvas before you start. This document details the rules for solving and handing in your solutions.

This homework has 50 T points in total. Problems appear in no particular order.

Please consult the most recent version of this document at Canvas before you contact us regarding something you think is wrong or should be clarified. We post versions with corrections or clarifications if necessary.

Problem 1 (Ciphers). In class we considered a substitution-permutation network taken from `https://www.engr.mun.ca/~howard/PAPERS/ldc_tutorial.pdf`. The paper describes linear cryptanalysis. Suppose that we change the permutation to the following and that the sender encrypts 8-bit ASCII-encoded English sentences.

input	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
output	2	3	4	1	7	8	5	6	12	11	10	9	14	13	16	15

Task 1.1 (1T). Describe in 2-3 sentences why this is a bad permutation.

Task 1.2 (5T). Describe a practical attack that exploits that the permutation is bad.

Problem 2 (Ciphers, 3T). Let G be a non-trivial finite abelian group with group operation $\otimes$. Consider the following symmetric cryptosystem. The key generation algorithm samples a random value $k \in G$. The encryption and decryption algorithms, respectively, are defined by $\mathsf{Enc}_k(m) = m \otimes k$, and $\mathsf{Enc}_k^{-1}(c) = c \otimes k^{-1}$, where k^{-1} is the inverse of k in G. Prove that this is a symmetric cryptosystem and that it has perfect secrecy for the encryption of a single message $m \in G$.

Problem 3 (Negligible Functions). Let $l(n)$ be a polynomial and let $\epsilon(n)$ be a negligible function.

Task 3.1 (1T). Prove that $l(n)\epsilon(n)$ is negligible in the parameter n.

Task 3.2 (1T). Consider a sequence of binary random variables $X_{n,1}, \ldots, X_{n,l(n)}$ such that $\Pr[X_{n,i} = 1] \leq \epsilon(n)$. Prove that $\Pr[\sum_{i=1}^{l(n)} X_{n,i} > 0]$ is negligible in the parameter n.

"If each bad event occurs with negligible probability and we have polynomially many events, then the probability that any bad event occurs is also negligible."

Problem 4 (Non-negligible Functions). Let $l(n)$ be a polynomial and let $\Delta(n)$ be a non-negligible function.

Task 4.1 (1T). Prove that $\Delta(n)/l(n)$ is non-negligible in the parameter n.

Task 4.2 (1T). Consider a sequence of binary random variables $X_{n,1}, \ldots, X_{n,l(n)}$ such that $\Pr[\sum_{i=1}^{l(n)} X_{n,i} > 0] \geq \Delta(n)$. Prove that there exists an index $i(n)$ such that $\Pr[X_{n,i(n)} = 1]$ is non-negligible in the parameter n.

"If something bad occurs with non-neglible probability and we have polynomially many events, then there is at least one specific bad event that occurs with non-negligible probability."

Problem 5 (Pseudo-random Generators). Consider two distinct functions f_1 and f_2 such that on input $x \in \{0,1\}^n$ give outputs in $\{0,1\}^{4n}$, i.e., they are expanding their inputs by a factor of 4. You know that at least one of the two functions is a pseudo-random generator, but not which one. Your goal is to construct a single pseudo-random generator under this assumption.

Task 5.1 (2T). Prove that $f(x) = f_1(x) \oplus f_2(x)$ is not necessarily a pseudo random generator, i.e., define distinct f_1 and f_2 (of which at least one is a pseudo-random generator) such that it is not and explain why. (You may assume that g is a pseudo-random generator that expands its input by a factor of 4 and use it to define suitable f_1 and f_2 as a counter example.)

Task 5.2 (5T). Prove that $f(x) = f_1(x_1) \oplus f_2(x_2)$, where x_1 and x_2 denote the first and second half of x is a pseudo-random generator with expansion 2, i.e., prove that if there is an adversary that violates the definition of a PRG for your function f, then there exists an adversary that violates it for f_1 and (a possibly different adversary) that violates it for f_2.

Problem 6 (Hash Functions). Suppose that $H = \{H_{n,\alpha}\}_{n \in \mathbb{N}, \alpha \in \{0,1\}^n}$ is a collision-resistant family of hash functions, where n is the security parameter and α is the key/index chosen randomly, and $H_{n,\alpha} : \{0,1\}^* \to \{0,1\}^n$.

Customers and data are identified by unique bit strings of finite lengths. Unfortunately, we do not know anything else about the formats used. You must generate an n-bit string identifier from any pair of customer identifier and data and make sure that collisions happen with negligible probability.

Task 6.1 (2T). Describe a polynomial time computable function $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ of a customer identifier and a data such that the unique bit string identifier is defined by $H(f(x,s))$, or more precisely as $H_{n,\alpha}(f(x,s))$ for the chosen security parameter n and a randomly chosen α.

Task 6.2 (3T). Prove that your construction is collision resistant, i.e., that no algorithm given α, where α is randomly chosen, can find $(x,s) \neq (x',s')$ such that $H_{n,\alpha}(f(x,s)) = H_{n,\alpha}(f(x',s'))$ in polynomial time in n except with negligible probability over the random choice of α.

Assume that α is chosen uniformly at random in $\{0,1\}^n$ for concreteness and to make sure that n is defined by the input to the adversary.

Problem 7 (Signature Schemes). Read about Lamport's one-time signatures that first computes a digest of a message and then signs the digest using Lamport's idea. Assume that the hash function is a random oracle with suitable range.

Task 7.1 (1T). Define the key generation, signature, and verification algorithms.

Task 7.2 (3T). How many signatures computed using distinct messages, but the same secret key, suffices to recover the complete secret key? Introduce suitable notation, restate the question mathematically, describe your attack, and prove that it works with probability at least $1/2$.

Problem 8 (Definitions). Recall the definition of CPA security.

Definition 1. Let $\mathcal{CS} = (\mathsf{Gen}, \mathsf{E}, \mathsf{D})$ be a cryptosystem and let $\mathcal{A}$ be a polynomial time algorithm. The chosen plaintext experiment $\mathrm{Exp}^b_{\mathcal{CS},\mathcal{A}}(n)$ is defined as follows, parametrized by a bit $b \in \{0,1\}$ and a security parameter $n \in \mathbb{N}$:

$$
\begin{aligned}
(1) \qquad & pk = \mathsf{Gen}(1^n) \\
(2) \qquad & (m_0, m_1, s) = \mathcal{A}(pk) \\
(3) \qquad & c = \mathsf{E}_{pk}(m_b) \\
(4) \qquad & d = \mathcal{A}(c, s) \\
(5) \qquad & \mathbf{output} \quad d
\end{aligned}
$$

Definition 2. A cryptosystem $\mathcal{CS} = (\mathsf{Gen}, \mathsf{E}, \mathsf{D})$ is CPA secure if for every polynomial time algorithm $\mathcal{A}$: $\left|\Pr\left[\mathrm{Exp}^0_{\mathcal{CS},\mathcal{A}}(n) = 1\right] - \Pr\left[\mathrm{Exp}^1_{\mathcal{CS},\mathcal{A}}(n) = 1\right]\right|$ is negligible in n, where the probability is taken over the randomness of the experiment.

Task 8.1 (2T). Prove that a CPA secure cryptosystem cannot have a deterministic encryption function.

Task 8.2 (1T). Suppose that the message space is $\{0,1\}^n$ and that the cryptosystem is only used to encrypt randomly chosen messages. Explain informally why the result you proved does not imply that this is necessarily insecure.

Problem 9 (Discrete Logarithms). Let q be an odd prime integer.

Task 9.1 (1T). Consider the additive group $\mathbb{Z}_q$, let g be a generator, and let y be a randomly chosen element in the group. Describe an efficient algorithm for computing the discrete logarithm in $\mathbb{Z}_q$ of y in the basis g.

Task 9.2 (2T). Consider a group G_q of order q and suppose that there is an isomorphism $f : G_q \to \mathbb{Z}_q$, which can be computed and inverted in polynomial time. Prove that the discrete logarithm problem in G_q can be solved in polynomial time.

Task 9.3 (3T). Suppose that $p = 2q+1$, where p is prime, let G_q be the subgroup of squares modulo p (i.e., the elements with Legendre symbol 1), and consider the function $\tau : G_q \to \mathbb{Z}_q$ defined by $\tau(x) = \min\{x, p-x\}$. Prove that this is: (i) a bijection by providing the inverse[1], and (ii) not an isomorphism.

Task 9.4 (1T). Google the MOV attack for elliptic curves, summarize it in a paragraph, and relate it to the subproblems above.

Task 9.5 (Optional, 0T). Recall that the map $\mathbb{Z}_q \to G_q$, $x \mapsto g^x$ is an isomorphism for any group G_q with prime order q and generator g. Why is this not a security problem? (Do not submit any solution. Think about it!)

---

[1] Hint: Use the Legendre symbol modulo p of $y = \tau(x)$ to recover x from y.

Problem 10 (Hybrid Argument, 7T). Let $f : \{0,1\}^* \to \{0,1\}^*$ be a pseudo-random generator (PRG) such that $|f(x)| = 2|x|$ for any input $x \in \{0,1\}^*$, i.e., it has expansion factor two. Let $0 < d(n) = O(\log n)$ and consider the function $t_d : \{0,1\}^* \to \{0,1\}^*$ defined as follows which increases expansion and allows parallelization.

1. On input $x \in \{0,1\}^n$ set $d = d(n)$.
2. Let (V, E) be the complete ordered binary tree of depth d with root r and leaves $l_0, l_1, \ldots, l_{2^d-1}$.
3. Define $x_r = x$, i.e., assign the value x to the root.
4. For every inner node $u \in V$ which has been assigned a value x_u with children $v_0, v_1 \in V$ which have not been assigned values:
   (a) compute $y = f(x_u)$ and
   (b) assign x_{v_0} and x_{v_1} such that $x_{v_0}||x_{v_1} = y$,

   i.e., evaluate the PRG on the value x_u assigned to u and assign the left and right halfs of the output to v_0 and v_1, respectively.
5. If any node has not been assigned a value, then go to Step 4.
6. Output $x_{l_0}||x_{l_1}||\cdots||x_{l_{2^d-1}}$, i.e., output the concatenation of the values assigned to the leaves.

Use a hybrid argument to prove that t_d is a pseudo-random generator, i.e., prove that if there is an adversary that violates the definition of a PRG for t_d, then there exists an adversary that violates it for f.

Problem 11 (Broken Cryptography). MD5 was constructed by Ron Rivest in (1991) and widely adopted in industry. Increasingly more practical attacks were discovered. Read the summary at Wikipedia [2].

RAINBOW[3] and SIKE[4] were submitted to NIST's competition for cryptographic algorithms. They were intended to be secure against quantum computers, but were broken by classical (non-quantum) algorithms. Read about this at, e.g., [5] [6] [7].

Task 11.1 (1T). Find three other practical attacks on cryptographic algorithms or protocols from the last 20 years and provide references for them.

Task 11.2 (3T). Imagine that you are a writer at Ars Technica. Summarize in at most one page one of the attacks for a general audience of computer scientists similarly to the one about SIKE.

Task 11.3 (Optional, 0T). Try to come up with your own efficient, practical, and hard computational problem that lends itself to use in cryptography. (Do not submit this as part of your solution.)

---

[2] https://en.wikipedia.org/wiki/MD5
[3] https://www.pqcrainbow.org
[4] https://sike.org
[5] https://medium.com/cambridge-quantum-computing/what-does-the-breaking-of-rainbow-mean-for-cybersecur
[6] https://www.cryptomathic.com/news-events/blog/nist-pqc-finalists-update-its-over-for-the-rainbow
[7] https://arstechnica.com/information-technology/2022/08/sike-once-a-post-quantum-encryption-contender