# Cryptography

## Problem set 2

1. Alice receives the ElGamal ciphertext $c = (32, 7)$, her public key is $(p = 47, g = 3)$. Determine the corresponding plaintext.
2. How can two ElGamal ciphertexts be used to generate a third ElGamal ciphertext of an unknown plaintext? How can this attack be prevented?
3. Prove that the hardness of Computational Diffie-Hellman (CDH) problem relative to $\mathcal{G}$ implies the hardness of Discrete Log (DL) problem relative to $\mathcal{G}$.
4. Prove that the hardness of the Decisional Diffie-Helman (DDH) problem relative to $\mathcal{G}$ implies the hardness of the Computational Diffie-Hellman problem (CDH) relative to $\mathcal{G}$.
5. Prove that the DDH problem is not hard in $\mathbb{Z}_p^*$. Hint: use the fact that quadratic residuosity can be decided efficiently modulo a prime.
6. Let $p$ be a large prime, such that $q := (p-1)/2$ is also prime. Let $\mathcal{G}$ be a subgroup of order $q$ in $\mathbb{Z}_p^*$. Let $g$ and $h$ be randomly chosen generators in $\mathcal{G}$. We assume that it is infeasible to compute discrete logarithms in $\mathcal{G}$. Show that

   $$f : \{0, \ldots, q-1\}^2 \to \mathcal{G}, f(x, y) := g^x h^y$$

   can be used to obtain a collision-resistant compression function.
7. Determine whether or not the following problem is hard. Let $p$ be prime, and fix $x \in \mathcal{Z}_{p-1}^*$. Given $p, x$, and $y := [g^x \bmod p]$ (where $g$ is a random value between 1 and $p-1$), find $g$; *i.e.,* compute $y^{1/x} \bmod p$. If you claim the problem is hard, show a reduction (to *i.e.,* discrete logarithm problem). If you claim the problem is easy, present an algorithm, justify its correctness, and analyze its complexity.
8. Show that a "textbook" RSA encryption scheme is not CCA-secure. Hint: having a ciphertext $c$ show how to select $c' \neq c$ such that knowledge of the corresponding plaintext $x' = \mathsf{Dec}_K(c')$ lets one to find $x$ $(= \mathsf{Dec}_K(c))$.
9. RSA (and Rabin) scheme is insecure if one is able to factor the modulus $n = pq$. It can happen [HDWH12] that because of *e.g.,* implementation mistakes two users share the same prime factor *i.e.,* Alice has $n = pq$ while Bob $n' = p'q$ in such a case Eve can find their private keys by just computing $gcd(n, n')$.

   RSA can be implemented in such a way that instead of generating $n = pq$, more primes are used *i.e.,* $n = p_1 \ldots p_k$. Lets call such a system $RSA - k$ (where original RSA is $RSA - 2$).

   Given $N$ number of $RSA-k$ keys, each generated from $l$-bit long primes, find the probability that there exists at least one pair of keys that share the same modulus (answer depends on: $N, k, l$). Compute exact values for $N = 6 \times 2^{20}; k = 2, 3, 4, 5; l = 128, 256, 512, 1024, 2028$.

# References


[HDWH12] Nadia Heninger, Zakir Durumeric, Eric Wustrow, and J Alex Halderman. Mining your ps and qs: Detection of widespread weak keys in network devices. In *USENIX Security Symposium*, pages 205–220, 2012.