Homework 8
COMP543 Fall 2020 - Modern Cryptography

Erhan Tezcan 0070881

25.11.2020

---

1. Quesitons

Q1: Let p, N be integers with $p|N$. Prove that for any integer x, $[[x \bmod N] \bmod p] = [x \bmod p]$. Show that, in contrast, $[[x \bmod p] \bmod N]$ need not equal $[x \bmod N]$.

A1: If $p|N$ then $p \leq N$ and $\mathbb{Z}_p \subseteq \mathbb{Z}_N$. We can suppose that x is positive without loss of generality. For a positive x with respect to N and p there are 3 cases:

- $0 \leq x < p$, then $x \in \mathbb{Z}_p$ and so $[x \bmod N] = [x \bmod p]$, then $[[x \bmod N] \bmod p] = [x \bmod p]$ is true.
- $p \leq x < N$ then $\exists x' \in \mathbb{Z}_N$ s.t. $x = pk + x'$ for some $k \in \mathbb{N}$. If $x' \in \mathbb{Z}_N$ and $x' > p$ then $\exists x'' \in \mathbb{Z}_p$ s.t. $x' = pk' + x''$ for some $k' \in \mathbb{N}$. If $x' = p$ then $x'' = 0$ and $k' := k' + 1$ instead. If $x' < p$ then $x'' = x'$ where $x'' \in \mathbb{Z}_p$ and $k' := 0$. As a result, $x = pk + pk' + x'' = p(k + k') + x''$.
- $N \leq x$ then $\exists x' \in \mathbb{Z}_N$ s.t. $x = Nk + x'$ for some $k \in \mathbb{N}$. Since $p|N$ we can say $\exists k_p \in \mathbb{N}$ where $N = pk_p$. By following a similar logic to what we did above, $x = Nk + pk' + x''$.. So, $x = pk_pk + pk' + x'' = p(k_pk + k') + x''$.

As the 3 cases demonstrate, the claim $[[x \bmod N] \bmod p] = [x \bmod p]$ is true, both sides eventually reduce to x''.

However, it is not always true that $[[x \bmod p] \bmod N] = [x \bmod N]$. We can prove this just by giving a counter-example. Take any p, $N = pk$ for some $k \in \mathbb{N}$ and $x = pk+r$ where $p < r < N$. Then $[x \bmod N] = r > p$, but on the left hand-side since we do $[x \bmod p]$ first, nothing after that will never be equal to r as $r \notin \mathbb{Z}_p$.

Q2: [1] Let ρ be a polynomial-time algorithm that, on input 1^n, outputs a (description of a) cyclic group G, its order q (with $||q|| = n$), and a generator g. If the discrete-logarithm problem is hard relative to ρ, then prove that the following hash function family (Gen, H) is a fixed-length collision-resistant hash function family.

- Gen: on input 1^n, run $\rho(1^n)$ to obtain (G, q, g), and then select $h \leftarrow G$. Output $s := \langle G, q, g, h \rangle$ as the key.
- H: given a key $s = \langle G, q, g, h \rangle$ and input $(x_1, x_2) \in Z_q \times Z_q$, output $H^s(x_1, x_2) := g^{x_1} \times h^{x_2} \in G$.

A2: I am following a proof similar to what is shown in section 8.4.2. of KL Book 2$^{\text{nd}}$ edition. To prove that if discrete-logarithm problem is hard relative to ρ then the hash function family (Gen, H) is secure, we take the contrapositive and assume that there exists an algorithm $\mathcal{A}$ than can break the hash function family, and construct an algorithm B that easily solves the discrete-logarithm problem. We see our constructed game in algorithm 1: for a security parameter n, `Chal` and B plays discrete-logarithm game, B and $\mathcal{A}$ plays hash-collision game. $\mathcal{A}$ breaks the hash-collision with probability $\epsilon(n)$.

Algorithm 1 The mixed game.

```
Chal runs ρ(n) and obtains (G, q, g).
h ← G.
(G, q, g, h) is given to B.
At B, s = ⟨G, q, g, h⟩ is constructed.
B gives s to 𝒜.
𝒜 returns x, x' to B.
if x ≠ x' and H^s(x) = H^s(x') then
  if h = 1 then
    Return 0.
  else
    Parse x as (x_1, x_2) and x' as (x'_1, x'_2) where x_1, x_2, x'_1, x'_2 ∈ ℤ_q.
    Return [(x_1 − x'_1)(x_2 − x'_2)^{−1} mod q].
  end if
else
  Return 0.
end if
```

What does it mean to have $H^s(x_1, x_2) = H^s(x'_1, x'_2)$? It means:

$$H^s(x_1, x_2) = g^{x_1} h^{x_2} = g^{x'_1} h^{x'_2} = H^s(x'_1, x'_2)$$

[1] "Intractable Problems" segment of Dan Boneh's Coursera course discusses this. Also see KL Book ed.2 section 8.4.2

Now if $g^{x_1}h^{x_2} = g^{x'_1}h^{x'_2}$ we can't have $x_1 = x'_1$ in $\mathbb{Z}_q$ because that would imply $x_2 = x'_2$ in $\mathbb{Z}_q$ and then inadvertnetly $x = x'$, which is not a collision. So indeed $x_2 \neq x'_2$ in $\mathbb{Z}_q$ and $x_1 \neq x'_1$ in $\mathbb{Z}_q$. Leaving g's and h's alone we get:

$$g^{x_1}g^{-x'_1} = h^{x'_2}h^{-x_2}$$

Since q is a prime order, the inverse $[(x'_2 - x_2)^{-1} \bmod q]$ exists. Show this inverse as i_2 (2 to indicate x_2 and x'_2). If we raise the expression above to this power:

$$g^{(x_1-x'_1)\times i_2} = h^{(x'_2-x_2)\times i_2} = h^1 = h$$

We see that $g^{(x_1-x'_1)^{i_2}} = h$, which solves the discrete logarithm of $log_g h$ to be $[(x_1 - x'_1) \times i_2 \bmod q] = [(x_1 - x'_1)(x_2 - x'_2)^{-1} \bmod q]$, which is what our algorithm returned. This tells us that if $\mathcal{A}$ find a collision with $\epsilon(n)$ probability then B solves discrete-logarithm with $\epsilon(n)$ probability. Since we assumed it is hard to break the discrete-logarithm probability, this $\epsilon(n)$ must be negligible, therefore the hash function is secure!