Sudipta Halder (2021202011)

Use DLP to build a fixed length collision resistant Hash function

Below is the construction of the DLP based fixed length collision resistant hash function.

A Fixed Length Hash Function

Let P be a polynomial time algorithm that on input 1^n outputs a cyclic group G of order q (length of q is n) and generator g

Gen: Run $P(1^n)$ obtain (G,q,g); select uniformly at random an element h from G; Output **s = (G,q,g,h)**

H: On input x_1 and x_2 (both in the range 0 to q-1), output

$$H^s(x_1,x_2) = g^{x_1} h^{x_2}$$

Theorem: If discrete logarithm problem is hard then the above is a fixed length collision resistant hash function

Now, we need to show that this construction is collision resistant.

THEOREM 7.73 *If the discrete logarithm problem is hard relative to $\mathcal{G}$, then Construction 7.72 is collision resistant.*

PROOF Let $\Pi = (\mathsf{Gen}, H)$, and let $\mathcal{A}$ be an arbitrary probabilistic, polynomial-time algorithm with

$$\varepsilon(n) \stackrel{\text{def}}{=} \Pr[\mathsf{Hash\text{-}coll}_{\mathcal{A},\Pi}(n) = 1]$$

(cf. Definition 4.10). Consider the following algorithm $\mathcal{A}'$ solving the discrete logarithm problem relative to $\mathcal{G}$:

Algorithm $\mathcal{A}'$:
The algorithm is given $\mathbb{G}, q, g, h$ as input.

1. Let $s := \langle \mathbb{G}, q, g, h \rangle$. Run $\mathcal{A}(s)$ and obtain output x and x'.

2. If $x \neq x'$ and $H_s(x) = H_s(x')$ then:
   (a) If $h = 1$ return 0
   (b) Otherwise ($h \neq 1$), parse x as $x_1 \| x_2$ and parse x' as $x'_1 \| x'_2$. Return

$$\left[(x_1 - x'_1) \cdot (x'_2 - x_2)^{-1} \bmod q\right].$$

Clearly, $\mathcal{A}'$ runs in polynomial time. Furthermore, the input s given to $\mathcal{A}$ when run as a subroutine by $\mathcal{A}'$ is distributed exactly as in experiment $\mathsf{Hash\text{-}coll}_{\mathcal{A},\Pi}$ for the same value of the security parameter n. (The input to $\mathcal{A}'$ is generated by running $\mathcal{G}(1^n)$ to obtain $\mathbb{G}, q, g$ and then choosing $h \in \mathbb{G}$ uniformly at random. This is exactly how s is generated by $\mathsf{Gen}(1^n)$.) So, with probability exactly $\varepsilon(n)$ there is a *collision*: i.e., $x \neq x'$ and $H_s(x) = H_s(x')$.

We claim that whenever there is a collision, $\mathcal{A}'$ returns the correct answer $\log_g h$. If $h = 1$ then this is clearly true (since $\log_g h = 0$ in this case). Otherwise, the existence of a collision means that

$$\begin{aligned} H_s(x_1 \| x_2) = H_s(x'_1 \| x'_2) &\Rightarrow g^{x_1} h^{x_2} = g^{x'_1} h^{x'_2} \\ &\Rightarrow g^{x_1 - x'_1} = h^{x'_2 - x_2}. \end{aligned} \tag{7.3}$$

Let $\Delta \stackrel{\text{def}}{=} x'_2 - x_2$. Note that $[\Delta \bmod q] \neq 0$ since this would imply that $[(x_1 - x'_1) \bmod q] = 0$, but then $x = x_1 \| x_2 = x'_1 \| x'_2 = x'$ in contradiction to the assumption that there was a collision. Since q is prime and $\Delta \neq 0 \bmod q$, the inverse Δ^{-1} exists. Raising each side of Equation (7.3) to the power Δ^{-1} gives:

$$g^{(x_1 - x'_1) \cdot \Delta^{-1}} = \left(h^{x'_2 - x_2}\right)^{\Delta^{-1}} = h^{[\Delta \cdot \Delta^{-1} \bmod q]} = h^1 = h,$$

and so

$$\log_g h = [(x_1 - x'_1)\Delta^{-1} \bmod q] = \left[(x_1 - x'_1) \cdot (x'_2 - x_2)^{-1} \bmod q\right],$$

the output returned by $\mathcal{A}'$.

We see that $\mathcal{A}'$ correctly solves the discrete logarithm problem with probability exactly $\varepsilon(n)$. Since, by assumption, the discrete logarithm problem is hard relative to $\mathcal{G}$, we conclude that $\varepsilon(n)$ is negligible. ■

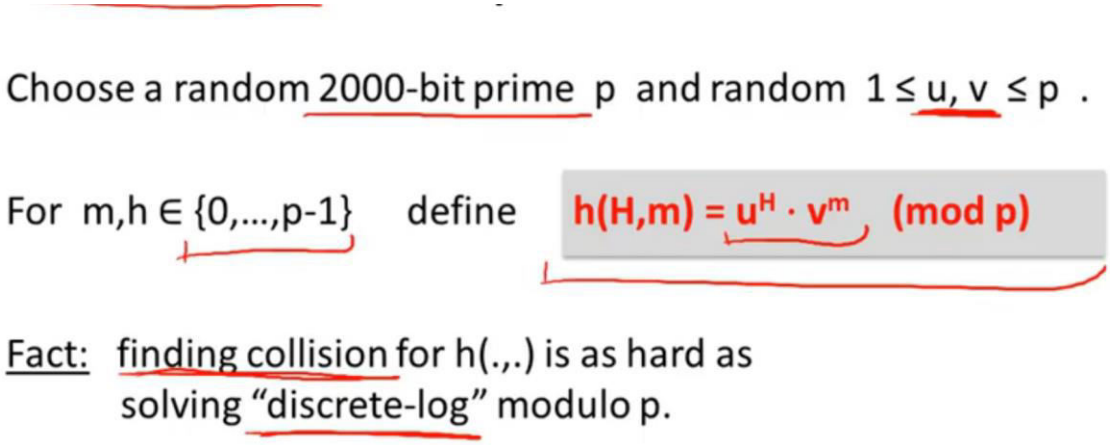
The above only shows collision resistance, but does not necessarily mean that Construction 7.72 is compressing. Indeed, as discussed earlier, whether or not the construction is compressing depends on the number of bits required to represent elements of $\mathbb{G}$. For many natural choices of groups, however, compression is attained; you are asked to prove this for one concrete example in Exercise 7.16. Interestingly, a generalization of Construction 7.72 can be used to obtain compression from any $\mathcal{G}$ for which the discrete logarithm problem is hard, regardless of the number of bits required to represent group elements.

[1]

Now, let's discuss the whole construction at once.

1. I have first taken input prime(p), generator(g), seed for PRG to generate h.
2. It will generate h which is in range [1, prime].
3. Then taken input two binary data x1 and x2. Which should be in range [0, prime-1].
4. It will first check whether x1 and x2 are in range of (0, prime-1) or not. If not, then it will apply mod operation on then to bring them in range (0, prime-1).
5. Then it calculate (g^{x1} mod p * h^{x2} mod p)mod p where p is prime. This result will then be output as binary.

This function ensures 50% data compression when data x1 and x2 are of same length and length = prime length in binary format i.e., if x1 and x2 is of length n, then hash result will output a n-bit hash.



References

[1] J. K. a. Y. Lindell, Introduction to Modern Cryptography.