(6) Use DLA to build a fixed length collision resistant hash function.

Hash function are simply function that take inputs of some length and compress them into short fixed length outputs. And a collision occurs when 2 elements end up being stored in same cell, increasing the lookup time. We can make our hash function collision resistant if it is infeasible for any probabilistic polynomial time algorithm to find a collision.

Proof:- Let $\Pi = (Gen, H)$ as

Gen: On input 1^n, run $G(1^n)$ to obtain (G, q, g) is then select $h \leftarrow G$ output $\langle G, q, g, h \rangle$

H: given by $s = \langle G, q, g, h \rangle$, input $= \langle x_1, x_2 \rangle \in Z_q \times Z_q$, output $H^s(x_1, x_2) = g^{x_1} h^{x_2}$

Let A be PPT algorithm with

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

we show how A can be used by an algorithm A' to solve DLP with succes ε.

Algorithm A'

→ with input G, q, g, h

→ Let $s: \langle G, q, g, h \rangle$ run $A(s)$ & obtain output x & x'.

→ If $x \neq x'$ & $H^s(x) = H^s(x')$, then

- if $h = 1$ return 0.
- otherwise parse x as (x_1, x_2) & parse x' as (x_1', x_2') return $(x_1 - x_1')(x_2 - x_2')^{-1} \bmod q$

- Clearly A' runs in polynomial time.
- Furthermore the input I is given to A when run as subroutine A' is distributed exactly as in expr Hash-coll$_{A,\pi}$.
- So with probability $\geq \varepsilon(n)$ their is a collision. We claim that whenever there is a collision, A' return the correct $\log_g h$.

If $h=1$, then $\log_g h = 0$ which is previously what A' returns.
otherwise

$$H^S(x_1, x_2) = H^S(x_1', x_2') \Rightarrow g^{x_1} h^{x_2} = g^{x_1'} h^{x_2'}$$
$$\Rightarrow g^{x_1 - x_1'} = h^{x_2' - x_2}$$

if $(x_2 - x_2') \bmod q = 0$, then $g^{x_1 - x_1'} = h^{x_2 - x_2'} = h^0 = 1$.
& $(x_1 - x_1') \bmod q = 0$, but then $x = (x_1, x_2)$
$= (x_1', x_2') = x'$ in contradiction

Then $(x_2' - x_2) \bmod q \neq 0$ has an inverse.

- $g^{(x_1 - x_1')[(x_2' - x_2)^{-1} \bmod q]} = (h^{(x_2' - x_2)})^{[(x_2' - x_2)^{-1} \bmod q]} = h^1 = h$

and also

$$\log_g h = [(x_1 - x_1')(x_2 - x_2')^{-1} \bmod q]$$

Conclusion:- If there exists a probabilistic polynomial time algorithm G relative to which DLP is hard, then there exist a collision resistant hash function.