EE 374: Blockchain Foundations **Stanford, Winter 2023**

Theory Exercise 1 Solutions

Problem 1

Let $H : \{0,1\}^* \longrightarrow \{0,1\}^\kappa$ be a collision-resistant hash function and define $G(x) = H(H(x))$. Use a reduction to show that G is a collision-resistant hash function.

Proof: For the sake of contradiction, suppose that G weren't a collision-resistant hash function. This means that there exists a PPT adversary $\mathcal{A}$ for which

$$\Pr[\textsf{collision-game}_{G,\mathcal{A}}(\kappa) = 1]$$

is a non-negligible function $f(\kappa)$. We construct another adversary $\mathcal{A}'$ as follows:

Algorithm 1 Adversary $\mathcal{A}'$ that breaks H.

```
1: function $\mathcal{A}'(1^\kappa)$
2:     $x_1, x_2 \leftarrow \mathcal{A}(1^\kappa)$
3:     if $H(x_1) = H(x_2)$ then
4:         return $(x_1, x_2)$
5:     end if
6:     return $(H(x_1), H(x_2))$
7: end function
```

Notice that $\mathcal{A}$ wins the collision-resistance game for G iff $\mathcal{A}'$ wins the collision-resistance game for H. Hence,

$$\Pr[\textsf{collision-game}_{H,\mathcal{A}'}(\kappa) = 1] = \Pr[\textsf{collision-game}_{G,\mathcal{A}}(\kappa) = 1]$$

for all κ. Furthermore, $\mathcal{A}'$ is PPT since $\mathcal{A}$ is PPT, and the if statement in the algorithm above guarantees that the two returned values of $\mathcal{A}'$ are distinct. Since $\mathcal{A}$ breaks G with non-negligible probability, $\mathcal{A}'$ will break H with non-negligible probability, contradicting H's collision resistance. ■

Alternate Proof: For the sake of contradiction, suppose that G weren't a collision-resistant hash function. This means that there exists a PPT adversary $\mathcal{A}$ for which

$$\Pr[\textsf{collision-game}_{G,\mathcal{A}}(\kappa) = 1]$$

is a non-negligible function $f(\kappa)$. We construct another adversary $\mathcal{A}'$ as follows:

Algorithm 2 Adversary $\mathcal{A}'$ that breaks H.

```
1: function $\mathcal{A}'(1^\kappa)$
2:     $x_1, x_2 \leftarrow \mathcal{A}(1^\kappa)$
3:     return $(H(x_1), H(x_2))$
4: end function
```

Notice that the probability $\mathcal{A}'$ breaks H is the same as the probability $\mathcal{A}$ returns x_1 and x_2 that break G and $H(x_1) \neq H(x_2)$ (think about why this latter condition is important). Furthermore, $\mathcal{A}'$ is guaranteed to be PPT since $\mathcal{A}$ is PPT.

Let E_1 correspond to the event that $\mathcal{A}$ outputs x_1 and x_2 that break G (i.e., $G(x_1) = G(x_2)$ and $x_1 \neq x_2$). Let E_2 correspond to the event that $\mathcal{A}$ outputs x_1 and x_2 for which $H(x_1) \neq H(x_2)$. Recall that $P(E_1) = f(\kappa)$ is non-negligible and that $P(E_2) = 1 - \text{negl}(\kappa)$ for some negligible function negl because H is collision-resistant.

If we show that $\Pr[E_1 \cap E_2]$ is non-negligible, then $\mathcal{A}'$ will break the collision resistance of H, completing the contradiction. We have

$$\begin{aligned}
\Pr[E_1 \cap E_2] &= \Pr[E_1] + \Pr[E_2] - \Pr[E_1 \cup E_2] \\
&\geq \Pr[E_1] + \Pr[E_2] - 1 \\
&= \Pr[E_1] + (1 - \text{negl}(\kappa)) - 1 \\
&= f(\kappa) - \text{negl}(\kappa)
\end{aligned}$$

Since the difference between a non-negligible function and negligible function is non-negligible, we are done. ■

Problem 2

Given a collision resistant hash function $H : \{0,1\}^* \longrightarrow \{0,1\}^\kappa$, create a hash function $G : \{0,1\}^* \longrightarrow \{0,1\}^\kappa$ with the following properties:

1. G is collision resistant.
2. The first two bits of G are predictable (a bit is predictable if it can be successfully "guessed" with probability $\frac{1}{2}$ + non-negligible when the input is chosen uniformly at random).

Write out explicitly what this G function is, then define and prove the above two properties formally. For the first property, use a reduction.

Define $G_\kappa(x)$ to be the concatenation of the bits 11 with $H_{\kappa-2}(x)$. The first two bits of G are predictable because they will always be 11 for all κ.

Suppose for the sake of contradiction that G weren't collision resistant. This means that there exists a PPT adversary $\mathcal{A}$ for which

$$\Pr[\textsf{collision-game}_{G,\mathcal{A}}(\kappa) = 1]$$

is a non-negligible function $f(\kappa)$. We construct another PPT adversary $\mathcal{A}'$ as follows:

Algorithm 3 Adversary $\mathcal{A}'$ that breaks H.

```
1: function $\mathcal{A}'(1^\kappa)$
2:     $x_1, x_2 \leftarrow \mathcal{A}\left(1^{\kappa+2}\right)$
3:     return $(x_1, x_2)$
4: end function
```

$\mathcal{A}'$ will succeed in breaking H_κ iff $\mathcal{A}$ succeeds in breaking $G_{\kappa+2}$ (why?). Hence,

$$\Pr[\textsf{collision-game}_{H,\mathcal{A}'}(\kappa) = 1] = f(\kappa + 2)$$

Since a horizontal translation of a non-negligible function is non-negligible, $\mathcal{A}'$ breaks the collision resistance of H, contradiction. ■

Problem 3

Construct a correct but insecure signature scheme. Write the full implementation of your signature scheme in pseudocode. Prove its correctness and insecurity.

The simplest example of a correct but insecure signature scheme is when the Ver function is defined to always return 1. The choice of Gen and Sig won't matter. This scheme is correct trivially by the definition of correctness. To show its insecurity, consider an adversary that returns any message signature combination without use of the oracle. The probability such an adversary wins the forgery game will be 1 for all κ which is non-negligible. Here is a precise implementation of one possible signature scheme:

Algorithm 4 Secret and public key generator Gen.

```
1: function Gen$(1^\kappa)$
2:     $sk \leftarrow 1^\kappa$
3:     $pk \leftarrow 1^\kappa$
4:     return $(sk, pk)$
5: end function
```

Algorithm 5 Signature creator Sig.

```
1: function Sig($sk, m$)
2:     return $sk$
3: end function
```

Algorithm 6 Signature verifier Ver.

```
1: function Ver($pk, m, \sigma$)
2:     return true
3: end function
```

Problem 4

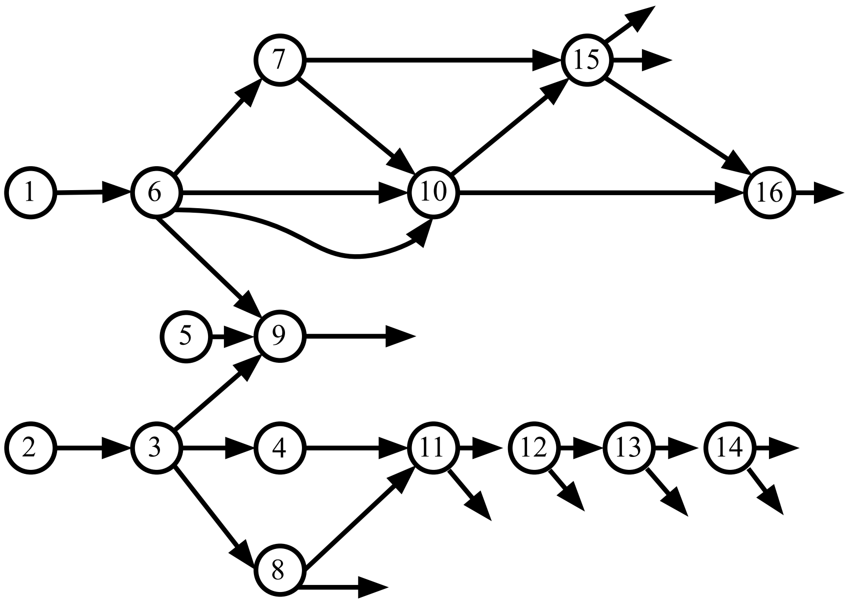

Figure 1: The transaction graph of Problem 4.

Consider the complete transaction graph ***accepted and stored in the database of an honest node*** illustrated in Figure 1. The circles represent transactions and the numbers within them are the txids. The outgoing edges are outputs, and the incoming edges are inputs. For each transaction, edges appear ordered from top to bottom. For this graph:

1. Identify the coinbase transactions.
2. Identify the double spending transactions. For each double spending transaction, identify the transaction it is conflicting with.

3. Identify the UTXO set.

4. Identify the transactions for which the Weak Conservation Law does not hold.

5. Identify the outputs of transactions that are partially, but not fully, spent.

To identify transactions, use their txids. To identify an output, use outpoint notation. Note that transaction 6 has four outputs.

1. Coinbase transactions are transactions with no inputs and one or more outputs. They correspond to transactions $1, 2, 5$ in the graph.

2. Honest nodes won't accept double spends, so there are no double spends in the graph.

3. The UTXO set consists of edges that don't point to other transactions. In the transaction graph, this corresponds to the following outpoint set:

$$\{(9,0),(14,0),(15,0),(16,0),(8,1),(11,1),(12,1),(13,1),(14,1),(15,1)\}$$

4. Honest nodes will only accept non-coinbase transactions that pass the Weak Conservation Law, so only coinbase transactions will violate it. Hence, we will have the same answer as in part 1 (i.e., transactions $1, 2, 5$).

5. It is not possible to only spend part of a transaction's output. They are either not spent (part of UTXO set) or completely spent.

Problem 5

The honest proof-of-work algorithm begins by sampling a uniformly random κ-bit nonce, then iterates for a polynomial number of repetitions while incrementing the nonce. Prove that, if a constant number of honest parties execute this algorithm a polynomial number of times, the probability that they query the hash using the same nonce is negligible.

Proof: Suppose that there are m honest parties that each increment the nonce a polynomial $p(\kappa)$ times. Let's compute the probability P that one or more of the honest parties query overlapping nonces using the Union Bound inequality.

Let X_{ij} be the indicator variable that takes on a value of 1 if parties i and j query overlapping nonces and is 0 otherwise. Notice that

$$\Pr[X_{ij} = 1] \leq \frac{2p(\kappa)}{2^\kappa}$$

because regardless of where party i starts, there are at most $2p(\kappa)$ possible starting nonces for party j that will cause an overlap (draw this out to see why!). Hence, by the Union

Bound inequality,

$$P = \Pr\left[\bigcup_{i<j} X_{ij}\right] \leq \sum_{i<j} \Pr[X_{ij} = 1] = \binom{m}{2}\frac{2p(\kappa)}{2^\kappa} \leq \frac{2m^2 p(\kappa)}{2^\kappa}$$

P is negligible since it is the ratio of a polynomial and exponential in κ, as desired. ■

Alternate Proof: Suppose that there are m honest parties that each increment the nonce a polynomial $p(\kappa)$ times. Let's compute the probability P that none of the honest parties query overlapping nonces.

Split the range from 0 to 2^κ into $\frac{2^\kappa}{p(\kappa)}$ intervals of length $p(\kappa)$. Each time an honest party picks a starting nonce, at most 3 intervals will be removed from the choices of the starting nonce of the next honest party. Hence,

$$\begin{aligned} P &\geq \left(\frac{\frac{2^\kappa}{p(\kappa)}}{\frac{2^\kappa}{p(\kappa)}}\right)\left(\frac{\frac{2^\kappa}{p(\kappa)} - 3}{\frac{2^\kappa}{p(\kappa)}}\right)\left(\frac{\frac{2^\kappa}{p(\kappa)} - 3(2)}{\frac{2^\kappa}{p(\kappa)}}\right)\cdots\left(\frac{\frac{2^\kappa}{p(\kappa)} - 3(m-1)}{\frac{2^\kappa}{p(\kappa)}}\right) \\ &\geq \left(1 - \frac{3m}{\frac{2^\kappa}{p(\kappa)}}\right)^m \end{aligned}$$

For sufficiently large κ, Bernoulli's inequality gives

$$\left(1 - \frac{3m}{\frac{2^\kappa}{p(\kappa)}}\right)^m \geq 1 - \frac{3m^2}{\frac{2^\kappa}{p(\kappa)}}.$$

As a result,

$$1 - P \leq \frac{3m^2}{\frac{2^\kappa}{p(\kappa)}} = \frac{3m^2 p(\kappa)}{2^\kappa}.$$

$1 - P$ can now be seen as negligible since it is the ratio of a polynomial and exponential in κ, as desired. ■

Reference

Some helpful definitions are provided below. For the full definitions, consult the lecture notes.

Definition (Collision Resistance)**.** A hash function $H : \{0,1\}^* \longrightarrow \{0,1\}^\kappa$ is *collision resistant* if for all PPT adversaries $\mathcal{A}$,

$$\Pr[\textsf{collision-game}_{H,\mathcal{A}}(\kappa) = 1] = \text{negl}(\kappa)\,.$$

The game is defined in Algorithm 7.

Algorithm 7 The collision-finding game for a hash function H.

```
function COLLISION-GAME$_{H,\mathcal{A}}(\kappa)$
    $x_1, x_2 \leftarrow \mathcal{A}(1^\kappa)$
    return $H_\kappa(x_1) = H_\kappa(x_2) \wedge x_1 \neq x_2$
end function
```

Definition (Correct Signature)**.** A signature scheme $(\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ is *correct* if, for all $m \in \{0,1\}^*$, whenever $(sk, pk) \leftarrow \mathsf{Gen}(1^\kappa)$, we have that $\mathsf{Ver}(pk, m, \mathsf{Sig}(sk, m)) = 1$.

Definition (Secure Signature)**.** A signature scheme $(\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ is *secure* if for all PPT adversaries $\mathcal{A}$,

$$\Pr[\mathsf{existential\text{-}forgery\text{-}game}_{(\mathsf{Gen},\mathsf{Sig},\mathsf{Ver}),\mathcal{A}}(\kappa) = 1] = \mathrm{negl}(\kappa)\,.$$

The game is defined in Algorithm 8.

Algorithm 8 The existential forgery game for a signature scheme (Gen, Sig, Ver).

```
1: function existential-forgery-game$_{(\mathsf{Gen},\mathsf{Sig},\mathsf{Ver}),\mathcal{A}}(\kappa)$
2:     $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$
3:     $M \leftarrow \emptyset$
4:     function $\mathcal{O}$(m)
5:         $M \leftarrow M \cup \{m\}$
6:         return $\mathsf{Sig}(sk, m)$
7:     end function
8:     $m, \sigma \leftarrow \mathcal{A}^{\mathcal{O}}(pk)$
9:     return $\mathsf{Ver}(pk, \sigma, m) \wedge m \notin M$
10: end function
```

Algorithm 9 The proof-of-work algorithm.

```
1: function PoW$_{H,T}$
2:     $\mathsf{ctr} \xleftarrow{\$} \{0,1\}^\kappa$
3:     while true do
4:         $B \leftarrow \mathsf{ctr}$
5:         if $H(B) \leq T$ then
6:             return B
7:         end if
8:         $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
9:     end while
10: end function
```