CS 491/591 Blockchains, HW1

Prof. Jared Saia, University of New Mexico

Note: The first problem in this hw is from the CS 251 "Cryptocurrencies and Blockchain Technologies" class at Stanford.

1. In class we defined two security properties for a hash function, one called collision resistance and the other called puzzle-friendly. Show that a collision-resistant hash function may not be puzzle friendly. Hint: Let $H : X \rightarrow \{0, \ldots 2^n - 1\}$ be a collision-resistant hash function. Construct a new hash function $H' : X \rightarrow \{0, \ldots 2^m - 1\}$ (for m possibly larger than n), where H' is collision-resistant but not puzzle-friendly. To show H' is collision-resistant, you can show that whenever there is a collision in H, there is also a collision in H'. To show that H' is not puzzle-friendly, you can show that for some difficulty D (say 2^{32}), it is computationally easy to find a value x such that $H'(x) \leq 2^m/D$.

2. k-ary Merkle trees. Alice can use a binary Merkle tree to commit to a set of elements S= $\{T_1, ..., T_n\}$ so that later she can prove to Bob that some T_i is in S using an inclusion proof containing at most $\lceil \log n \rceil$ hash values. The binding commitment to S is a single hash value.

   In this question your goal is to explain how to do the same using a k-ary tree, that is, where every non-leaf node has up to k children. The hash value for every non-leaf node is computed as the hash of the concatenation of the values of all its children.

   (a) Suppose $S = \{T_1, \ldots, T_9\}$. Explain how Alice computes a commitment to S using a 3-ary Merkle tree. How does Alice later prove to Bob that T_4 is in S?

   (b) Suppose S contains n elements. What is the length of the proof that proves that some T_i is in S, as a function of n and k?

   (c) For large n, if we want to minimize the proof size, is it better to use a binary or a 3-ary tree? Why?

3. A one-way function is a function that can be computed in polynomial time on every input, but can not be inverted in polynomial time, i.e. $f(x)$ can be computed in polynomial time. But the output of any randomized algorithm F that tries to invert f is correct only with small probability. In particular, for all randomized algorithms F, any constant $c > 0$, and for $length(x) = n$ sufficiently large,

$$Pr(f(F(f(x)) = f(x)) < n^{-c}$$

The cryptographic hash functions discussed in class are special cases of one-way functions.

Recall that the set P is the set of languages that can be recognized in polynomial time, e.g. a language like "The set of graphs with Minimum spanning trees with cost less than 100." And NP is the set of languages that have a polynomial time checkable certificate that a string is in the language. For example, one language in NP consists of the pairs (x, y), where x is a graph that is 3 colorable and y is a certificate giving the 3-coloring of G.

   - Show that if one way hash functions exist, then $P \neq NP$. You can assume that the functions always map inputs of length n bits to outputs of length $\Theta(n)$ bits. Hint: You may find it easier to prove the contrapositive: If $P = NP$, then one-way functions do not exist. Then, think about developing a language that is in NP (and thus in P by the assumption), where that language can help you, when given y, build up bit by bit an inverse value x such that $f(x) = y$.

4. In lecture, we stated that in a finite size group, every element has finite order. Prove that statement. Hint: Remember that in a group, every element has an inverse.

5. Consider the following magic trick. Let $p = 13$. Pick **any** positive integer x. Now compute $(px + 1)^p \bmod p$. You will always get back the number 1.

   (a) Compute the outcome of this trick by hand with your favorite integer x. Use multiplicative and additive facts about modular arithmetic to speed up your computation by hand. Show your work. What final answer did you get? Eerie, right?

   (b) Now explain this magic trick based on things we proved in lecture.

(c) Now, make up your own magic trick based on your favorite lemma or theorem from lecture.

6. Recall the Fiat-Shamir public-key digital signature scheme we discussed in class. In this problem, you'll do a toy example of that scheme over the multiplicative group $Z_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. Note that a generator for this group is $g = 2$. Assume that Alice has private-key $x = 2$ and public key $y = 2^x$. Let the random target (or commitment) chosen by Alice be $t = g^7 = 7 \pmod{11}$, and let the random challenge chosen by Bob be $c = 2$.

   (a) What is the correct response, r that Alice will choose such that $g^r y^c = t$, and how does she compute it?

   (b) When the group size is large, what makes it hard for someone else to find the correct value for r?

   (c) If Alice wants to prove she knows x without relying on Bob, how should she choose c?