CS 491/591 Blockchains, HW2

Prof. Jared Saia, University of New Mexico

Note some of the problems here are based on problems from our textbook

1. Birthday Attack. Let H be an ideal hash function with range $[0, 2^n)$. H is ideal in that the first time $H(x)$ is computed for a value x, the result is distributed independently and uniformly at random among all values in the range. Trivially, we can compute $2^n + 1$ different values of H to guarantee that we find a collision. This takes only $O(1)$ space.

   (a) What if you have $O(2^{n/2})$ space? Can you take $o(2^n)$ expected time to get a collision? Hint: Birthday paradox. You may also find useful the fact that $1 - x \leq e^{-x}$ (this follows from the Taylor expansion of e^x).

   (b) What if you have 2^x space for some value x? What is the best expected time algorithm you can get with that much space?

   (c) If you can find values x and y such that $x \neq y$ and $H(x) = H(y)$, what type of attack can you run on Bitcoin? Maximize the monetary reward of your attack :) Note: Finding two input values that collide is *not* the same as finding a value x that collides with a **specific** target value t (i.e. $H(x) = H(t)$).

   (d) *Hard:* Is there an attack where the product of time and space complexity is $o(2^n)$?

2. In the following, you can use `<data>` as a shorthand to represent data values pushed onto the stack. You can also use non-standard transactions and op codes that are currently disabled. For a reference see `https://en.bitcoin.it/wiki/Script`

   (a) Write a Bitcoin ScriptPubKey script for a transaction that can be redeemed by anyone who supplies a square root of 1681. (Don't include the square root in your script - you're paying someone to compute it for you. Your script should just check the supplied answer)

(b) Write a ScriptSig script to redeem your transaction

(c) Sketch how (in theory) you might create a ScriptPubKey script that can be redeemed by anyone factoring a RSA factoring challenge, i.e. finding the factors of a very large number. Are there additional op commands you might need to write your script?

3. Imagine that Alice and Bob are entities with known names (i.e. public keys) in Bitcoin. For example, Alice is a bank (like Mt Gox) and Bob represents a privately-held company. Alice wants to purchase Bob's company, 1% of the company for 1 bitcoin, over a 100 day period. Alice and Bob don't want anyone to learn about the purchase plan until it has been fully completed (for example, so that competitors don't learn about the purchase beforehand). Also, they do not want to create or use any additional IDs - all bitcoins can only be transferred from Alice's account to Bobs, this protects against double-spend attacks because of the potential damage to their public reputations.

   Explain exactly how they perform this process without leaking any information to the blockchain about the purchase. Ensure that if Alice backs out after buying say just x% of the company over x days, that Bob can get x bitcoins transferred to him. And ensure that if Bob tries to walk away without transferring any percentage of the company that Alice does not lose any of her bitcoins. Hint: Review Lecture 3 "Mechanics".

4. Your boss suggests that the message sender should add a round number to each message of the Dolev-Strong flooding algorithm. Then processes in round r should wait to receive at least $n - t$ "round r" messages before completing that round, where n is the total number of processes and t is the maximum number of faults. Your boss claims that this new algorithm will solve consensus with $t < n$ crash faults.

   Will this work? If so, prove it. In not, use the FLP theorem to show precisely what the adversarial scheduler/process crasher should do to keep this new algorithm from terminating correctly.

5. A signature scheme is said be *malleable* if for all messages m, a valid signature σ on m can be easily transformed into a different valid signature σ' on m. In this problem, you will show that malleable schemes can lead to security problems in a cryptocurrency.

   Consider a simple, newspaper-ad-based cryptocurrency blockchain, where transactions look like this:

$$\begin{aligned}
\text{coin}_1 &\leftarrow \{\text{Create 1 Bitcoin (serial\#53401) for } PK_{Fred}\}_{SK_{Fred}} \\
\text{coin}_2 &\leftarrow \{\text{Pay H}(coin_1) \text{ to } PK_{David}\}_{SK_{Fred}} \\
\text{coin}_3 &\leftarrow \{\text{Pay H}(coin_2) \text{ to } PK_{Dan}\}_{SK_{David}}
\end{aligned}$$

Here the notation m_{SK} denotes a pair (m, σ) where σ is a signature on m generated using the key SK. Every day the paper publishes a classified ad that looks like: new-coins, H(yesterday's ad).
Everyone using the system builds a database containing a hash of every coin ever published, along with one bit saying if the coin has already been spent. Nodes ignore blocks that contain a coin whose hash is already in the database (a duplicate hash) or spending a coin containing a spent or non-existent hash. For instance, an attempt to republish $coin_1$ would fail, as would an attempt to re-spend $coin_2$ such as:

$$\text{coin}_4 \leftarrow \{\text{Pay H}(coin_2) \text{ to } PK_{David}\}_{SK_{David}}$$

Nevertheless, malleable signatures introduce a vulnerability. Show how someone might attack this system when we use malleable digital signatures. Hint: Assume a node may see a transaction before it is published in the newspaper.