Question 1

Hash functions and proofs of work

Let $H : P \times S \to \{0, 1, ...2^n - 1\}$ be a collision resistant hash function.

Let H' be a new hash function, $H' : P \times S \to \{0, 1, ...2^{n'} - 1\}$ (where $n' = n + \log(d)$), defined as:

$$H'(p, s) = (num(s) \bmod d) \mid H(p, s)$$

Here $num(s)$, denotes the number representation of s. In other words the new hash function is defined by append $H(p, s)$, to modulo of s wrt d.

Claim 1**:** H' is collision resistant hash function.

Proof. Suppose H' is not a collision resistant hash function, ie. its easy to find x, y, such that $H'(x) = H'(y)$ and $x \neq y$. Which means that last n bits of $H'(x)$ are same as that of $H'(y)$, or $H(x) = H(y)$. This contradicts or assumption that H was collision resistant. ■

claim 2**:** H' is not proof-of-work secure.

Proof. Note that for every puzzle $p \in P$, $H'(p, 0) = (0 \bmod d) \mid H(p, 0) < \dfrac{2^{n'}}{d}$.

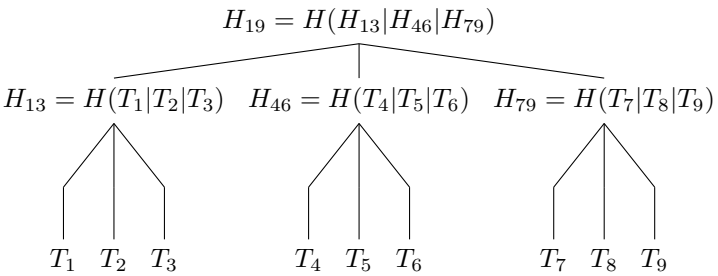
For a fixed difficulty(d), it's trivial to find a solution, or H' is not proof-of-work secure. ■

Therefore, there is a collision resistant hash function that is not proof-of-work secure.

Question 2

Beyond binary Merkle trees

(a)



Commitment tree for $S = \{T_1, T_2, ...T_9\}$
Each node of the tree stores the hash of it's 3 children concatenated. Finally Alice keeps the hash value of root of merkle tree.
To prove membership of T_4, Alice need to provide values of T_5, T_6, H_{13}, H_{79}. Bob verifies them by computing H_{46} using T_4, T_5, T_6, and then computes H_{19} using computes H_{46} and provided H_{13}, H_{79}, and compare with the root hash stored earlier.

(b)

Note that height of above merkle tree as a function of n, k is $\lceil \log_k n \rceil$ For each level we need to store $k - 1$ for proof.
Length of proof to prove existance of T_i is $(k - 1)\lceil \log_k n \rceil$, in above example it is $2\lceil \log_3 n \rceil$

(c)

In a binary tree $k = 2$, ie proof size = $\lceil \log n \rceil$, while for a general k-ary tree it is $(k-1)\lceil \log_k n \rceil$

Their ratio is:

$$\frac{(k-1)\lceil \log_k n \rceil}{\lceil \log n \rceil}$$

ignoring the ceil function we get:

$$\frac{(k-1)\log_k n}{\log n} = \frac{(k-1)\log 2}{\log k} >> 1 \text{ as } k \text{ increases}$$

One advantage of this over the regular binary merkle tree can be that we would need to compute very few hash , ie this would be beneficial for constly hash functions. Precisely we need to call H $\lceil \log_k n \rceil$ times, which is approximately $\log k$ times better that binary tree.

Question 3

Hiding vs. binding commitments

(a)

Assuming 10 digit phone numbers there can be atmost 10^{10} valid phone numbers. Bob can precompute the hash of all these valid phone numbers, and create a hashmap of hash value to mobile number. Whenever a new user registers, he just checks his/her phone number from the hashmap, he can do the same thing for the contacts of this user. Hence Bob can act maliciously to determine the phone numbers and contacts of all BobCrypt users.

(b)

Using a 128 bit nounce will not work since then it would not be possible to add friends on the app. Let's say B creates a new account, and B has $A's$ contact number in his phone, but since the does not store the nounce explicity, B will not be able to add A just using his phone number. Hence BobCrypt 2.0 will not be able to provide intended functionality.

Question 4

Bitcoin script

(a)

ScriptSig is just the password.

(b)

This is not a secure way since this SciptSig would be visible in the blockchain when she does a transaction. In other words her password is publically visible in blockchain. So if there is any money left in her account, someone can make a transaction using this password.

(c)

In order to spend the bitcoin send via P2SH, the recipient must provide a script matching the script hash, and the data that will make the script evaluate to true. Now to make a transaction alice need to

Question 5

Lightweight clients

(a)

Since Bob only has the header of last block of blockchain, Alice needs to send the merkle tree proof of the block in which this transaction is present. Along with merkle tree proof, she also need to send the all the block headers from the second most recent block to the one having this transaction.

To check Bob will verify merkle tree existance using the proof provided, compute block hashes consequtively and then finally verify the block hash with current block header's previous block field.

(b)

$$\begin{aligned}\text{Proof size} &= \text{Block headers + Merkle Proof} \\ &= k * size\_of(\text{block\_header}) + \lceil \log n \rceil * size\_of(\text{one merkle proof unit})\end{aligned}$$

Assuming that merkle root is implement using SHA2, we need 32 bytes for one unit of merkle proof. Also note that one block header is approximately 80 bytes.

$$\begin{aligned}\text{Proof Size} &= 8 * 80 + \lceil \log 256 \rceil * 32 \\ &= 640 + 8 * 32 \\ &= 896\end{aligned}$$

(c)

Question 6

BitcoinLotto

(a)

To prevent the winner from spending money on first we can implement a 2 of 2 multisig. In other words, the winner has a private key, but to use the money, transaction must be signed by trusted printing factory (who have the other private key) and the winner. Private key of winner is implemented using 2-of-2 multisig.

(b)

To prevent loss of lost ticket's money, we can make a transaction with locktime of 1 week to the next week's address. If the ticket is lost, the money will automatically go to the next week's lottery.