EE374 Blockchain Infrastructure **Stanford, Spring 2025**

Homework #1 Solutions

Due: Fri, 11-April-2025, 11:59pm – Gradescope entry code: `R57ZN7`

Please upload your answers timely to Gradescope. Start a new page for every problem. We strongly suggest LaTeX to type your answers. For the programming/simulation questions you can use any reasonable programming language (please no assembly, brainfuck, etc. ☺). Comment your source code and include the code and a brief overall explanation with your answers. A tentative point distribution (in % of the total) is provided in brackets. For most problems there is more than one valid way of solving them!

1. (20%) Nakamoto mentioned the word "trust" multiple times in his P2P Foundation post.

   a) What's wrong with "trust"?

   Answer:

   Trust can be breached by malicious or faulty parties.

   ---

   b) Explain in what sense is a secure digital signature scheme "trustless"?

   Answer:

   No third party needs to be trusted for the unforgeability of digital signatures; this is guaranteed by cryptographic hardness assumptions.

   ---

   c) Is Nakamoto's proof-of-work longest chain protocol trustless? If not state the trust assumptions for the protocol. How do they differ from the trust assumptions of a traditional currency?

   Answer:

   No, it is not trustless; it requires over half of the hash power to be honest. It differs from the trust assumptions of a traditional currency; as trust is decentralized to

many parties, only half of which (by hash power) must be assumed honest for security (rather than a single trusted third party).

---

2. (30%) In this question we discuss the stochastic modelling of the mining times of Bitcoin.

   a) A reasonable model for the distribution of the time between two consecutive mining events is exponential. Write down this probability density function with specific choice of all parameters to model ideal Bitcoin.

Answer:

Mining consists of trying many nonces until a block $b \triangleq (\mathsf{prev}, \mathsf{txs}, \mathsf{nonce})$ satisfies a hash inequality $H(b) \leq \mathsf{threshold}$. Trials in the PoW mining process can be modeled as independent Bernoulli random variables with success probability $\mathsf{threshold}$. The number of trials until the first success follows a geometric distribution, and as the success probability is very low and the number of trials very large, the block inter-arrival time is well approximated by an exponential distribution.

The probability density function of an exponential random variable T, with rate λ is $f_T(t) = \lambda e^{-\lambda t}$. The mean of this exponential distribution is $1/\lambda$. (See `https://en.wikipedia.org/wiki/Exponential_distribution`)

In Bitcoin, the average time between consecutive mining events is 10 minutes. This means $1/\lambda = 600\,\mathrm{s}$. With t in seconds, the probability density function is

$$f_T(t) = \frac{1}{600} e^{-t/600}.$$

---

   b) What is the standard deviation of the inter-mining time under this model? What is the ratio of the standard deviation over the mean?

Answer:

For an exponential random variable with rate λ, both the mean and standard deviation are equal to $1/\lambda$. So, the standard deviation is 10 minutes, and the ratio of the standard deviation over the mean is 1.

---

c) What is the mean of the time it takes to mine 10 blocks? What is the standard deviation, and the ratio of the standard deviation over the mean? Be explicit about any assumptions you made to get this conclusion, and justify your modeling assumptions.

Answer:

Suppose the time taken to mine each of the 10 blocks, after their previous block was mined, is $T_1, T_2, \ldots, T_{10}$. We know that each of $T_1, T_2, \ldots, T_{10}$ has an exponential distribution with mean 10 mins. We can further make a reasonable assumption that $T_1, T_2, \ldots, T_{10}$ are independent random variables. This is because the hash values for different blocks are independent. Due to this independence,

$$\mathbb{E}[T_1 + \cdots + T_{10}] = \mathbb{E}[T_1] + \cdots + \mathbb{E}[T_{10}] = 100 \text{ mins},$$
$$\mathrm{Var}(T_1 + \cdots + T_{10}) = \mathrm{Var}(T_1) + \cdots + \mathrm{Var}(T_{10}) = 10 \cdot (10 \text{ mins})^2.$$

The standard deviation of $T_1 + \cdots + T_{10}$ is $10\sqrt{10}$ mins ≈ 31.62 mins. Thus the ratio of standard deviation over mean is $1/\sqrt{10} \approx 0.316$. Comparing with the ratio for mining a single block (above), this indicates that the time taken to mine a large number of blocks becomes relatively less spread around its mean.

---

d) Using data from `https://btc.com/block`, estimate the standard deviation of the inter-block mining time. Is it close to what your model in part (a) predicts? Explain if there is any significant discrepancy. State carefully how you perform the estimation and justify why you estimate this way.

Answer:

From `https://btc.com/block?date=2021-01-18`, we can extract the mining time of 145 blocks. Taking the difference of successive mining times, we get $n = 144$ samples for the time between mining events: $T_1, \ldots, T_n$. First calculate the sample mean $\overline{T} = \frac{1}{n}\sum_{i=1}^{n} T_i$. The sample standard deviation can be calculated as

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(T_i - \overline{T})^2}.$$

For the blocks from 2021-01-18, the sample mean was 596.31 s, and the sample standard deviation was 629.99 s. Both the values are very close to 600 s, indicating that the mining times fit the exponential distribution of part a) very well.

---

3. (20%) The total hashrate of the Bitcoin network on January 1, 2018 was 14.4 EH/s.

   a) Estimate the threshold in the hash inequality on that day from this fact. Compare this with the true threshold. Why might there be a discrepancy?

Answer:

14.4×10^{18} hashes per second corresponds to 8.64×10^{21} hashes per 10 minutes. Since only one block on an average solves the hash inequality in 10 minutes, the threshold should be $2^{-\log_2(8.64\times 10^{21})} \approx 2^{-73}$. This corresponds to ≈ 73 leading 0 bits or ≈ 18 leading 0 hex digits.

Looking at the hash of the first block mined in 2018,
`https://btc.com/00000000000000000001a73ede5478e76141520514aec65f0257e29499c53771b`
(cf. `https://btc.com/block?date=2018-01-01`), we find this confirmed.

A discrepancy could result from a mismatch in difficulty, or if by chance the block's hash is significantly smaller than required by the difficulty. In this case, the heuristic of estimating the 'true threshold' by inspecting block hashes fails. (For the proper way of obtaining the true threshold from the difficulty, see `https://en.bitcoin.it/wiki/Difficulty`. However, all that was expected for this problem was the heuristic used in class.)

---

   b) Assume all mining was conducted using back then state-of-the-art Antminer S9 hardware, which delivers 14 TH/s at a power consumption of 1372 W. What was the power consumption of the Bitcoin network? Find a comparable country in `https://en.wikipedia.org/wiki/List_of_countries_by_electricity_consumption`.

Answer:

Under the specified circumstances, the Bitcoin network would have used a power of $\frac{14.4\times 10^{18}}{14\times 10^{12}} \cdot 1372 = 1.4 \times 10^9$ Watt. (This is comparable to a nuclear power station, `https://en.wikipedia.org/wiki/List_of_nuclear_power_stations`.)

From the referenced Wikipedia list, 1 kW·h/yr = 0.11408 Watt, so that 1.4×10^9 Watt $\approx 1.23 \times 10^{10}$ kW·h/yr. This corresponds roughly to the electricity consumption of Lithuania.

Remark: As of April 2025, the energy consumption of Bitcoin is estimated (`https://digiconomist.net/bitcoin-energy-consumption`) to be $\approx$ 175.87 TW·h/yr, which is comparable to Poland.

---

4. (30%) Let $H : \{0,1\}^k \to \{0,1\}^k$ denote a cryptographic hash function, and define the following signature scheme with security parameter k:

   - $\mathsf{G}()$: Choose 2ℓ elements (each consisting of k bits) randomly from the set $\{0,1\}^k$: $(x_{1,0}, x_{1,1}, \ldots, x_{\ell,0}, x_{\ell,1}) \leftarrow (\{0,1\}^k)^{2\ell}$. Find $y_{i,j} = H(x_{i,j})$ for all $i \in \{1,\ldots,\ell\}$ and $j \in \{0,1\}$. Then, output

   $$(\mathsf{pk} \triangleq (y_{1,0}, y_{1,1}, \ldots, y_{\ell,0}, y_{\ell,1}), \mathsf{sk} \triangleq (x_{1,0}, x_{1,1}, \ldots, x_{\ell,0}, x_{\ell,1}))$$

   - $\mathsf{S}(\mathsf{sk}, m \in \{0,1\}^\ell)$: Parse sk as $(x_{1,0}, x_{1,1}, \ldots, x_{\ell,0}, x_{\ell,1})$. To sign a message $m \in \{0,1\}^\ell$, first parse m as a bit string $m = (b_1, \ldots, b_\ell)$, where each b_i is a bit. Then, output the signature $\sigma \triangleq (x_{1,b_1}, \ldots x_{\ell,b_\ell})$.
   - $\mathsf{V}(\mathsf{pk}, m, \sigma)$: Parse pk as $(y_{1,0}, y_{1,1}, \ldots, y_{\ell,0}, y_{\ell,1})$, σ as $(\hat{x}_1, \ldots, \hat{x}_\ell)$ and the message m as a bit string $m = (b_1, \ldots, b_\ell)$. If $H(\hat{x}_i) = y_{i,b_i}$ for all $i \in \{1,\ldots,\ell\}$, then output 1. Else, output 0.

   To make the signature scheme concrete, let's walk through a small example. Suppose $k = 256$ and $\ell = 4$. In this case, $\mathsf{G}()$ randomly samples some values $x_{1,0}, x_{1,1}, x_{2,0}, x_{2,1}, x_{3,0}, x_{3,1}$, $x_{4,0}$, and $x_{4,1}$ from $\{0,1\}^{256}$, and outputs

   $$\mathsf{sk} \triangleq (x_{1,0}, x_{1,1}, x_{2,0}, x_{2,1}, x_{3,0}, x_{3,1}, x_{4,0}, x_{4,1})$$

   and

   $$\mathsf{pk} \triangleq (y_{1,0}, y_{1,1}, y_{2,0}, y_{2,1}, y_{3,0}, y_{3,1}, y_{4,0}, y_{4,1}),$$

   where

   $$y_{1,0} \triangleq H(x_{1,0}),\ y_{1,1} \triangleq H(x_{1,1}), y_{2,0} \triangleq H(x_{2,0}),\ y_{2,1} \triangleq H(x_{2,1})$$
   $$y_{3,0} \triangleq H(x_{3,0}),\ y_{3,1} \triangleq H(x_{3,1}),\ y_{4,0} \triangleq H(x_{4,0}),\ y_{4,1} \triangleq H(x_{4,1})$$

   Now, as the signature σ_m for the message $m = (0,1,0,0)$, the signing algorithm S outputs $\sigma_{m=(0,1,0,0)} = (x_{1,0}, x_{2,1}, x_{3,0}, x_{4,0})$. Upon receiving the public key pk, the message m and the signature σ_m, the verification algorithm V checks if $y_{1,0} = H(x_{1,0})$, $y_{2,1} = H(x_{2,1})$, $y_{3,0} = H(x_{3,0})$ and $y_{4,0} = H(x_{4,0})$.

   Please answer the following questions:

   a) In the small example above, what is the signature for the message $m = (1,1,1,0)$? Does this signature verify given the message m (i.e., does V output 1 upon receiving pk, m and this signature)?

   b) For a given k and ℓ (some polynomial of k), is the signature scheme correct? Explain.

   c) For a given k and ℓ (some polynomial of k), is this signature scheme secure? If so, argue briefly. If not, describe an adversary $\mathcal{A}$ that wins the chosen message attack game against the challenger (recall the security definition for signature schemes).

d) If each pair of keys $(\mathsf{pk}, \mathsf{sk})$ were to be used to sign a *single* message, would this signature scheme be secure? If so, argue briefly. If not, describe an adversary $\mathcal{A}$ that wins the chosen message attack game against the challenger (recall the security definition for signature schemes).

This is called a Lamport signature, named after Leslie Lamport (also an author of the paper 'The Byzantine Generals Problem').

Answer:

a) The signature is $\sigma_{m=(1,1,1,0)} = (x_{1,1}, x_{2,1}, x_{3,1}, x_{4,0})$. Yes, the signature verifies against the message m; since $y_{1,1} = H(x_{1,1})$, $y_{2,1} = H(x_{2,1})$, $y_{3,1} = H(x_{3,1})$ and $y_{4,0} = H(x_{4,0})$.

b) Yes, the signature scheme is correct. Consider a message $m = (b_1, \ldots, b_\ell) \in \{0,1\}^\ell$, and some keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{G}()$. Then, given the signature $\sigma_m = (x_{1,b_1}, \ldots, x_{\ell,b_\ell}) \leftarrow \mathsf{S}(\mathsf{sk}, m)$, it holds that for all $i \in \{1, \ldots, \ell\}$, $y_{i,b_i} = H(x_{i,b_i})$. In other words,

$$\Pr[\mathsf{V}(\mathsf{pk}, m, \mathsf{S}(\mathsf{sk}, m)) = 1] = 1$$

Since this holds for all messages $m \in \{0,1\}^\ell$ and all keys $(\mathsf{pk}, \mathsf{sk})$ output by $\mathsf{G}()$, the signature scheme satisfies correctness.

c) No, the signature scheme is not secure. Consider an adversary $\mathcal{A}$ that queries the challenger with the messages $m_0 = (1, 1, \ldots, 1)$ and $m_1 = (0, 0, \ldots, 0)$, and obtains the signatures σ_0 and σ_1 respectively, in the signature security game. Let $\sigma_0 = (\hat{x}_1^0, \ldots, \hat{x}_\ell^0)$ and $\sigma_1 = (\hat{x}_1^1, \ldots, \hat{x}_\ell^1)$. After obtaining σ_0 and σ_1, $\mathcal{A}$ can forge a signature for any message! In other words, it can output the candidate forgery pair $(m^* = (b_1, \ldots, b_\ell), \sigma^* = (\hat{x}_1^{b_1}, \ldots, \hat{x}_\ell^{b_\ell}))$ for any $m^* \neq m_0, m_1$, and $\mathsf{V}(\mathsf{pk}, m^*, \sigma^*) = 1$. This is because, by definition $\hat{x}_i^0 = x_{i,0}$ and $\hat{x}_i^1 = x_{i,1}$ for each $i \in \{1, \ldots, \ell\}$, implying that $\sigma^* = (x_{1,b_1}, \ldots, x_{\ell,b_\ell})$ (which verifies by the argument in part (b)).

d) Yes, then the signature scheme would be secure. In fact, Lamport signatures are secure *one-time* signatures. When each pair of keys are used to sign a single message, the adversary is allowed to query the challenger only once before outputting its forgery pair in the signature security game. Towards contradiction, suppose there exists an efficient adversary $\mathcal{A}$, which given $\mathsf{pk} = (y_{1,0}, y_{1,1}, \ldots, y_{\ell,0}, y_{\ell,1})$, and the response σ_m to its query m, outputs a correct forgery pair consisting of $m^* = (b_1^*, \ldots, b_\ell^*) \neq m$ and $\sigma^* = (\hat{x}_1^*, \ldots \hat{x}_\ell^*)$. By definition, it holds that $H(\hat{x}_i^*) = y_{i,b_i^*}$ for all $i \in \{1, \ldots, \ell\}$. Denoting $m = (b_1, \ldots, b_\ell)$ and $\sigma_m = (\hat{x}_1, \ldots \hat{x}_\ell)$, since $m^* \neq m$, there exists an index i' such that $b_{i'} \neq b_{i'}^*$. This means that $\hat{x}_{i'}^* \notin \{\hat{x}_1, \ldots \hat{x}_\ell\}$, i.e., the adversary *inverts* the cryptographic hash function $H(.)$ by finding $\hat{x}_{i'}^*$ such that $H(\hat{x}_{i'}^*) = y_{i',b_{i'}^*}$. This is a contradiction with our definition of the hash function (to be formal, the hash function is assumed to be a one-way function).
