CSCI-GA.3033-107: Cryptography of Blockchains **Spring 2024**

Homework #2

Due: 11:59pm on Sunday, **3/10, 2024**
Submit via Brightspace

Problem 1. Distributed Randomness Beacons using VDFs A distributed randomness beacon (DRB) allows mutually untrusted parties to engage in a protocol and derive a random value ω that is not biased by any party. Consider the verifiable delay function $\mathsf{VDF}(x) : \mathbb{G} \to \mathbb{G} = x^{2^t}$. Let g be a generator of the group $\mathbb{G}$ and let $h = \mathsf{VDF}(g)$. Consider the following one-round protocol DRB_1:

Round 1: Each party reveals a random value r_i.

Suppose n parties participate in Round 1. The DRB output is obtained as $\omega = (\prod_{i=1}^{n} r_i)^{2^t}$.

a. Show that this protocol is not secure and can be biased. That is, describe an attack an adversary acting as of the parties can perform that lets them set the DRB output to any value they want. (Note the adversary can get preparation time before the protocol starts.)

b. Suggest a solution to this attack based on the solution used to prevent the rogue-key attack when aggregating BLS signatures. (You need not prove your answer. A brief justification is enough.)

c. An issue with the VDF-based DRBs is that they always require a costly VDF computation to be done. Consider the following protocol DRB_2:

Round 1: Commit Round: Each party i samples a random nonce α_i and shares $c_i = g^{\alpha_i}$.

Round 2: Reveal Round: After all parties share their commitments, each party reveals α_i. (If $c_i \neq g^{\alpha_i}$ then this α_i is ignored).

Suppose n parties participate in Round 1. There are two scenarios here, based on the behavior of the n parties in Round 2. The optimistic scenario allows all the parties to compute the output quickly while the pessimistic one resorts to the costly VDF computation.

- Optimistic case: If all n parties reveal a valid nonce in Round 2, then the DRB output ω is quickly obtained as $\omega = \mathsf{VDF}(\prod_{i=1}^{n} h^{\alpha_i})$.
- Pessimistic case: If any party skips Round 2, then the DRB output is obtained as $\omega = \mathsf{VDF}(\prod_{i=1}^{n} g^{\alpha_i})^{2^t}$.

d. This scheme also suffers from a biasing attack similar to the one in part (a). Show how.

e. As in part (b), write out a solution that prevents the above attack. (Again, it's based on the technique used in BLS signature aggregation. And you need not formally prove security.)

f. Even after this fix, there is an inherent weakness associated with any DRB that has an "optimistic" fast case and a "pessimistic" slow case. Show how the a party participating in the protocol can learn ω much faster than all the other parties. (Note that they don't bias the value. They only learn it much faster than the others.)

Problem 2. Certificates of Correctness

In class, we so far have seen ways to obtain quorum certificates where each signature has equal weight. What if we want to extend this to a scenario where different nodes have different weights? This may be the scenario in a Proof-of-Stake blockchain where the more stake you have, the more your vote counts. Instead of needing 2/3rd of all miners for a quorum, you need 2/3rd of all the weight (that is, stake).

a. Let node i have weight w_i for $i = 1, \ldots, n$. Modify the Merkle tree construction for QC to show that some fraction x of all stake is in the quorum.

Problem 3. More VDFs

a. Is Proof-of-Work a VDF? Why or why not? If it is, describe the VDF input, output, and prove and verify functions. If not, describe what property of a VDF is not satisfied.

b. Can a VDF be used as a Proof-of-Work function? For a given challenge ch, the goal of the miner to now produce a valid output and proof for a given VDF function on input ch.

c. A commonly used randomness beacon is the Bitcoin blockchain. Suppose random value r is obtained by hashing the latest block header b as $r = H(b)$. What is one issue with this method? (Hint: think about which parties can bias the randomness.)

d. What if the randomness is obtained by hashing the VDF output on the header, instead? That is, $r = H(\mathsf{VDF}(b))$. Briefly explain (doesn't have to be formal) how this is secure or insecure.

Problem 4. Batching Polynomial Commitments

In this question, we'll build and efficient batch evaluation proofs from a linearly homomorphic polynomial commitment scheme. Here, a prover P and verifier V have input polynomials $f_0, \ldots, f_{\ell-1} \in \mathbb{F}^{<d}[X]$. The prover claims that $f_i(\omega_i) = 0$ for all $i = 0, \ldots, \ell - 1$ and elements $\omega_0, \ldots, \omega_{\ell-1}$. The interactive protocol with a non-succinct verifier to prove the following is:

1. V sends P a challenge $\rho \in \mathbb{F}$.
2. P sends V the polynomial $q(X) = \sum_{i=0}^{\ell-1} \rho^i f_i(X)/(X - \omega_i)$.
3. V sends P a challenge $r \in \mathbb{F}$.
4. Let $z(X) = \prod_{i=0}^{\ell-1}$ and $z_i(X) = z(X)/(X - \omega_i)$. V computes the polynomials $g(X) = \left(\sum_{i=0}^{\ell-1} \rho^i z_i(r) \cdot f_i(X)\right) - z(r) \cdot q(X)$
5. V accepts if $g(r) = 0$.

The verifier can be made succinct using polynomial commitments. In this setting, the input additionally contains commitments $C_0, \ldots, C_{\ell-1}$ to the polynomials $f_0, \ldots, f_{\ell-1}$. In the questions below, you will work out how the steps change when using polynomial commitment schemes.

a. In step 2 above, the prover will send a commitment C_q to polynomial $q(X)$. Using C_q and all other inputs and challenges, show how the verifier can compute a commitment C_g to polynomial $g(X)$ in Step 4 efficiently.

b. How does step 5 change when using polynomial commitments? (Note the succinct verifier only has commitment to $q(X)$ and $g(X)$.)

We'll generalize the protocol to prove the claim $f_i(\omega) = v_i$ for $i = 0, \ldots, \ell - 1$.

c. What is the new q in this scenario? And how does V compute $C_g(X)$? Short one-line answers are sufficient. (Hint: reduce this to the original case by modifying the polynomials sent as input.)

Problem 5. Security of Wesolowski VDF

This question will analyze the role of the prime challenge ℓ in Wesolowski VDFs. To establish notation, the input is $(\mathbb{G}, g, h, t)$, w here $g, h \in \mathbb{G}$ and the prover claim is that $h = g^{2^t}$. The verifier sends a challenge ℓ, which is a prime. The prover responds π where $\pi = g^q$ and $2^t = q\ell + r$ and $r < \ell$. The

a. What goes wrong when ℓ is not a prime? Show how an adversary can either produce the VDF output without performing t sequential steps, or produce a convincing proof of an invalid output.

b.

c. Show that the scheme isn't secure when ℓ is the product of small primes each less than k (for some constant k), by describing an attack that runs in time $O(k \cdot \ell)$ group operations.