Superquantum's Challenge Description

One of the center pieces of the proof-of-work concept used in cryptocurrencies is an existence of a certain hash function, which is resilient to preimage attacks. This provides a way for the blockchain participants to know for certain that a meaningful amount of computational work has been performed to find an input that results in a specific output of that function. Our challenge is centered around a goal of developing a high-quality hash function based on the simulation of quantum circuits to be used in quantum-powered blockchain systems.

As for the more specific challenge requirements, the students will be required to develop a hash function in Python using their quantum simulator of choice. The input to this function will be an array of 2^N bytes, where N can be any natural number greater than or equal to 5, and the function must produce the byte output of the same size. The function can only utilize the resulting expectation values for each qubit after conducting quantum simulations.

The grading will be based on the following criteria:

1. Output determinism - the function must always produce the same output for a given input.
2. Preservation of entropy - provided that the function receives random samples from the uniform distribution over its domain, its outputs should, on average, be distributed as uniformly as possible over its range.
3. Computational difficulty - the function should not be computable in sub-exponential time complexity in the number of input bits.
4. Preimage resistance - the inverse of the function should not be trivially computable.
5. Collision resistance - there should be no trivial way to find two distinct input values for which the hash function produces the same output.
6. Computational feasibility - the function should not utilize the simulation of circuits with the number of qubits exceeding 20 for any number of input bits not exceeding 256.
7. Computation time - the average time taken to calculate a single hash value should be as low as possible.
8. Purely quantum hashing - the function must not use any classical hash function on the inputs to and the outputs from the quantum circuit.

Some of these criteria will be judged quantitatively using a set of predetermined metrics for the hash function’s performance (strict avalanche criterion, bit independence, etc.), while others, such as the computational difficulty, will be evaluated by manually inspecting the structure of the hash function.

For each of the listed criteria, the contestants are encouraged to prove their function’s compliance or quantify its performance, to gain additional points. Another way to improve the team’s standing is to do the same analysis for the quantum portion of the Qubitcoin’s qhash algorithm for the criteria 1-5.

Lastly, as a bonus problem, contestants can try to develop a hash function that can accept variable-length input and produce fixed-length 256 bit output. It is advised to take on this challenge only after designing and analysing a solution to the main problem.

The contestants are required to include the following materials in their final submission:

- *main.py*, along with other necessary Python files, implementing the proposed hash function. The function should take *bytes* as input, return *bytes*, and be clearly marked in the code with appropriate comments.
- *requirements.txt*, listing the necessary Python packages for executing main.py
- *writeup.pdf*, detailing the work performed by the team and covering all obtained results
- any additional Python scripts/notebooks containing the analysis code used in the write-up
- *presentation.pptx*, containing the team's solution presentation

Description du Défi de Superquantum

L'un des éléments centraux du concept de preuve de travail utilisé dans les cryptomonnaies est l'existence d'une certaine fonction de hachage, résistante aux attaques de préimage. Cela permet aux participants de la blockchain de savoir avec certitude qu'une quantité significative de travail computationnel a été effectuée pour trouver une entrée qui produit une sortie spécifique de cette fonction. Notre défi est centré sur l'objectif de développer une fonction de hachage de haute qualité basée sur la simulation de circuits quantiques à utiliser dans des systèmes blockchain alimentés par la technologie quantique.

En ce qui concerne les exigences spécifiques du défi, les étudiants devront développer une fonction de hachage en Python en utilisant le simulateur quantique de leur choix. L'entrée de cette fonction sera un tableau de 2^N octets, où N peut être tout nombre naturel supérieur ou égal à 5, et la fonction doit produire une sortie en octets de la même taille. La fonction ne peut utiliser que les valeurs d'espérance résultantes pour chaque qubit après avoir effectué des simulations quantiques.

Les critères d'évaluation seront les suivants :

1. Déterminisme de la sortie - la fonction doit toujours produire la même sortie pour une entrée donnée.
2. Préservation de l'entropie - à condition que la fonction reçoive des échantillons aléatoires de la distribution uniforme sur son domaine, ses sorties doivent, en moyenne, être distribuées aussi uniformément que possible sur son image.
3. Difficulté computationnelle - la fonction ne doit pas être calculable en complexité temporelle sous-exponentielle en fonction du nombre de bits d'entrée.
4. Résistance aux préimages - l'inverse de la fonction ne doit pas être trivialement calculable.
5. Résistance aux collisions - il ne doit pas exister de moyen trivial de trouver deux valeurs d'entrée distinctes pour lesquelles la fonction de hachage produit la même sortie.
6. Faisabilité computationnelle - la fonction ne doit pas utiliser la simulation de circuits avec un nombre de qubits dépassant 20 pour tout nombre de bits d'entrée ne dépassant pas 256.
7. Temps de calcul - le temps moyen nécessaire pour calculer une seule valeur de hachage doit être aussi faible que possible.
8. Hachage purement quantique - la fonction ne doit pas utiliser de fonction de hachage classique sur les entrées et les sorties du circuit quantique.

Certains de ces critères seront jugés quantitativement à l'aide d'un ensemble de métriques prédéterminées pour la performance de la fonction de hachage (critère strict d'avalanche, indépendance des bits, etc.), tandis que d'autres, comme la difficulté computationnelle, seront évalués en inspectant manuellement la structure de la fonction de hachage.

Pour chacun des critères énumérés, les participants sont encouragés à prouver la conformité de leur fonction ou à quantifier ses performances, afin de gagner des points supplémentaires. Une autre façon d'améliorer le classement de l'équipe est de réaliser la même analyse pour la partie quantique de l'algorithme qhash de Qubitcoin pour les critères 1 à 5.

Enfin, en tant que problème bonus, les participants peuvent essayer de développer une fonction de hachage capable d'accepter une entrée de longueur variable et de produire une sortie fixe de 256 bits. Il est conseillé de relever ce défi uniquement après avoir conçu et analysé une solution au problème principal.

Les participants doivent inclure les éléments suivants dans leur soumission finale :

- *main.py*, ainsi que d'autres fichiers Python nécessaires, implémentant la fonction de hachage proposée. La fonction doit prendre des *bytes* en entrée, retourner des *bytes*, et être clairement marquée dans le code avec des commentaires appropriés.
- *requirements.txt*, listant les packages Python nécessaires pour exécuter main.py
- *writeup.pdf*, détaillant le travail effectué par l'équipe et couvrant tous les résultats obtenus
- tout script/notebook Python supplémentaire contenant le code d'analyse utilisé dans le rapport
- *presentation.pptx*, contenant la présentation de la solution de l'équipe