Schnorr Signature For Blockchains

Kuei-Fang Hsueh, 1001506346 Aicha Lawand, 1001797530

***Abstract*—Blockchain technology has evolved a lot over the last few years. However, scalability seems to be a major challenge for the prevalent adoption of the already existing blockchain systems. A lot of methods were proposed for scaling the blockchain such as sharding, lightning network and others. This paper presents our proposed solution that focuses on reducing the number of bytes used in each block in the entire blockchain, which will lead to save more space in the block but also increase the throughput. In order reach to this solution, we thought of implementing Schnorr signatures in our own created blockchain and evaluate its performance and compare it with the Elliptic Curve Digital Signature Algorithm (ECDSA) that is more commonly used in the already existing blockchains such as Bitcoin. All over this paper, we will first explain the problem we are trying to solve and our proposed solution, the we explain the Schnorr signature algorithm along with a comparison with ECDSA signature, and finally we analyse and discuss all of our results taken from static and simulation measurements.**

I. Introduction: Research Gap

Blockchain is a decentralized distributed data base which has improved gradually over the last few years, however, it suffers from a scalability problem. This problem is actually the limited size and frequency of blocks in a blockchain causing transactions to be processed at a limited rate in the network.

A. Relevance of research

In recent years, blockchain as a payment system's popularity has increased dramatically. However compared to the almost globally adopted payment systems like Visa and Master Card, each handling with over 1000 transactions a second, the aforementioned problems still deeply impact the usability of blockchain payment systems. Research, like ours, that tackle the scalability of blockchains will help reduce the cost of owning and operating a node in the network. The current proof of work structure of Bitcoin is not only costly for the operator in terms of paying for electricity but it is also negatively impacting the environment by using electricity to solve hash problems with no real world benefits. Another area of the scalability research helps reduce the amount of storage required to store historical ledgers. Despite the original white paper's claim that storage will be trivial and the cheaper cost of storage by the year, increased block sizes and the complexity of data stored in block chains threaten this assumption. A blockchain system that cannot practically allow any one of its node to verify the chain because it is too large to download a copy of the entire ledger would violate the desired decentralized property. As an hypothetical example, if all but a handful of nodes held by one interest group had the compute and storage to verify the validity of the chain, the network will be forced to place its trust in a centralized entity.

B. Research challenge

Scaling the blockchain system is challenging and not straightforward to solve. Recently, blockchain approached its transaction throughput limitation which led to an increase in the transaction's fees. So, some debates were raised on scaling. One group focuses on increasing the block size limit, which is a short-term solution for increased fees and not a sufficient improvement. Another group focuses on off-chain scaling by adding protocols on higher layers. One example for this idea is Lightning Network (LN) which adds a layer to the blockchain and let its users build payment channels between two distinct parties. As a result, we will have immediate transactions, at very low cost. However, some privacy issues may be encountered since transactions on LN happen off the chain and are not followed by the main chain. This debate around blockchain scalability demonstrates that the problem is challenging to solve and current solutions are not enough. Also, it is difficult to make the change since no one can be forced to do anything in a decentralized system. That's why we need a solution that works best for all stakeholders.

II. Related Work

One main concern for blockchains scalability is to increase the transactions per second. The authors of [1] experiments what would happen to a bitcoin network if the block size where allowed to vary up to 16x the default block size of the currently deployed bitcoin protocol. Their conclusion was that due to transmission delays a simple increase in block size creates inconsistent copies of blockchains on the same network. The authors proposed a improved blockchain protocol that involves the mining of keyblocks to help the network reach consensus despite the increased block size and inevitable transmission delay. However the simulation was demonstrated for on a network of 10 mining pools.

As the blockchain increases in length, it's digital size also increased. One method discuss in [2] summarizes the data in the blocks in attempt to compress it. However the summary method proposed by this paper introduces large overheads for the "tip of the blockchain" due to the fact that forking often occurs in this region. Furthermore, the authors does not make it clear how efficient decompression is or whether the compression algorithm can allow the compressed blocks to still be validated.

Other researchers thought of increasing the scalability of blockchain by using a smaller number of bytes in each block. To do so, they used alternative data structures for Merkle trees. The authors of [3] proposed using Merkelized Abstract Syntax Trees (MAST) for regulating transactions inside a block. They implemented two experiments where one of them was for code compression. They quantified code compression by using python scripts. Using a Merkle tree at first, the script had a length of two million characters. However, after applying MAST to the same python script, the number of characters reduced to 23500 characters which indicates a 90% compression rate.

III. Proposed Solution

One potential solution for Blockchain scalability problem is to include more transactions in a block. In fact, each transaction has information that must be kept in each block for verifiability and security. If we use less bytes to represent this information, then we will have a better throughput. Also, nodes will be allowed to easily verify transactions and space saving for blockchain will be increased. To reach to this solution, we employ a more efficient hashing algorithm that produces short signatures such as Schnorr signature. Schnorr signatures used in Bitcoin would be beneficial, where there is a size advantage for transactions, for Schnorr signature covered at 64 bytes, compared to 71-75 bytes for the signing algorithm used for bitcoin ECDSA (Elliptic Curve Digital Signature Algorithm). Every day, 200 000 transactions are sent on average. Which means that reducing signature size has an impact on daily data added to the blockchain. We intend to solve the problem by implementing and evaluating this signature on Bitcoin Blockchain and show some savings. Also, we will demonstrate that our new hash structure still allows for simple payment verification, which is what Merkle trees allow.

A. Schnorr Signature

Schnorr signature was proposed by Claus Schnorr. It is a simple and efficient digital signature that generates short signatures. However, when Bitcoin was introduced, Schnorr signature was not standardized nor employed in open source crypto libraries and that's because it was protected by a patent expired in 2008. In fact, Bitcoin uses Elliptic Curve Digital Signature Algorithm (ECDSA) as a signing algorithm. Despite the fact that Schnorr algorithm was considered as an elegant signature with a simple mathematical proof, it was protected which resulted in the employment and standardization of ECDSA in Bitcoin.

Algorithm: Schnorr signature algorithm starts with choosing the parameters, then there is key generation, after that the message is signed and finally the signature is verified [4]. For choosing parameters, there is a group G of prime order q, with a generator g that all signature users agree on. There is also a cryptographic hash function H, mapping data (message) to a bit string of fixed size (hash), that all users agree on. Next comes the key generation where a private signing key x is chosen from the set of multiplicative group of integers modulo q and the public verification key is derived accordingly as

$$y = g^x$$

. After choosing the parameters and generating the keys, a message m is signed. In order to implement this, we chose a random k for the same set that x was chosen from. A bit string r is initiated such as

$$r = g^k$$

, then e is initiated such as

$$e = H(r||m)$$

which is the hashing function H applied to the concatenation of r and m. Finally, we let

$$s = k–xe$$

, where the pair (s,e) is the signature. The last step in the algorithm is the verification. Here, we let

$$r_v = (g^s).(y^e)$$

and

$$e_v = H(r_v||m)$$

where we check if

$$e_v = e$$

, for that the signature is verified [4]. It is important to note that there is a poof of correctness in Schnorr signature algorithm. This means that when a message is signed correctly, it will verify correctly [4]. As for the security of the algorithm, it is secure if the hash function is modeled as a random oracle. In fact, the response of a random oracle to each unique query is true and random. The response is the same for each time a query is repeated [4].

B. Comparison of Schnorr with ECDSA

Schnorr and ECDSA signatures are compared according to their algorithm, length, security, malleability and linearity.

- Algorithm: Schnorr signature constitute an efficient hashing algorithm producing short signatures. ECDSA signature is used to create Bitcoin signatures, and if compared to Schnorr signature, we realize that both have the same security assumptions. This means that Schnorr signature is compatible with the elliptic curve used in Bitcoin (secp256k1). Actually, the notation secp256k1 refers to the parameters of the elliptic curve employed in Bitcoin's public key cryptography. Therefore, Schnorr signatures are appropriate with the key derivation schemes used currently and their creation can me made with the same private keys as ECDSA. To generate keys, both signatures choose first the private signing key from an allowed set, then compute the public key accordingly. The public key function is different in ECDSA, it is a point on the elliptic curve calculated by multiplying

the private key with the generator point g [5].To sign, they both use the same cryptographic hash function SHA-256 to hash a message and both take as inputs the private key and a message, however the math behind the sign function is a little bit different. To verify, both signatures take as inputs the signature already found from the sign function, the message and the public key already generated [5]. The output in both is a boolean value indicating whether the signature was verified or not, however there is a bit difference in the math behind the verify function. So, when comparing the algorithm between Schnorr and ECDSA signatures, we notice that they have a lot of similarities, but also some differences related to how the keys are generated and to what sets they belong to, which leads to some differences in how the sign and verify functions are computed.

- Size: ECDSA and Schnorr signatures have different sizes per signature. The size of an ECDSA signature varies, but most of the time it is around 71 or 72 bytes. It is higher than the size of Schnorr signature, where 64 bytes is its maximum length [6]. It is concluded from this fact that there is about a 12 percent reduction in signature size [6]. From this conclusion we think that using Schnorr signatures would help reach to keep some space savings for more transactions inside one block. That's how we decided to integrate Schnorr in the blockchain and evaluate its performance.

- Security Proof: Unlike ECDSA signatures, Schnorr signatures are provably secure. In fact, when a message attack occurs, Schnorr signatures are highly unforgeable in the random oracle model while assuming that the elliptic curve discrete logarithm problem (ECDLP) is hard. Contrarily, ECDSA signatures' security depend on stronger assumptions, which makes them non provably secure [7].

- Non–Malleability: ECDSA signatures are inherently malleable, which means that there is a high possibility for a third party to make changes in an existing valid signature for a given public key without accessing the secret key. But also message another valid signature for the same key and message [7]. Contrarily, Schnorr signatures don't have this issue since they are non-malleable.

- Linearity: Various participating parties are enabled by an efficient and simple method employed in Schnorr signatures to produce a valid signature for the sum of their public keys. This is actually the basis for many constructions on a higher-level enhancing efficiency and privacy, such as multisignatures and others [7].

IV. Experimental Setup

In order to evaluate Schnorr signatures against ECDSA ones we developed our own blockchain in python and deployed it over 6 cloud compute nodes to collect experimental results. We initially hoped that our simplified blockchain written in python would give more researchers access to easy to use frameworks upon which different security alrogithms, networks and such parameters could be experimented upon. However, after discovering Parity which readily does what we intended to do, the reason for our choice of sticking with python and developing our own blockchain became just for the sake of our own learning. We used VULTR as our cloud based virtual machine where there is a storage of 25 GB SSD, a virtual CPU with 2.4 GHz, a memory of 1024 MB and a bandwidth of 1000 GB. In total, we set up 6 identical cloud instances as blockchain nodes. Each node had 2 peers. Written as a pythonic set, the directional network is $= \{[2,3],[1,3],[4,5],[5,6],[1,6],[2,3]\}$. For example, node 1 gossips to nodes 2 and 3 and node 6 receives gossips from nodes 4 and 5.

The network was arbitrarily selected and held constant throughout the experiment. While the peer network could prove interesting to study, it wasn't the purpose of our work to explore its effects on the blockchain's performance.

Fig. 1. Map showing the locations of nodes

V. Results and Discussion

Several key indicators for performances were used in our experiments. To introduce them briefly they are:

- Size in bytes of the block - total number of bytes used by serialized block data (json serialized dictionary).
- Verification and signing latency - average time taken per verification or signature
- Transaction throughput - total number of transactions processed after mining 50 blocks (not adjusted for duplicates)
- Longest sustained consensus - number of consecutive blocks in which there were no forks

A. Experiment 1: Latency of Schnorr DSA

In this experiment, 1000 transactions were generated using a randomly selected pair of sender receiver addresses. 50 of

addresses were prepared before generating the transactions and the address pairs were picked from this set. The signing of the 1000 transactions using the Schnorr DSA and ECDSA were timed. Before obtaining the verification latency, 50% of the transactions were altered by simply changing the transaction timestamp. We tamper the transactions before measuring the verification latency to remove the potentially existing bias conditioned upon valid transactions. Again all 1000 transactions were verified and the average time per transaction measured. Our results indicated that Schnorr is slower to both verify and sign. For every Schnorr verification, 23 ECDSA verifications can be completed and for every Schnorr signature, 46 ECDSA signatures can be done. In context of blockchains, signature latency of Schnorr should not greatly affect the chain performance because signatures are done by the clients. However it is worth noting that our simulation framework does not yet have dedicated compute acting as clients. Instead each node randomly generates the transactions at some schedule (this should be improved in the next iteration). Schnorr signatures also have the useful property that aggregate public keys and aggregated signatures can be treated as one single public key and signature pair to be verified. The 0.14s it takes for Schnorr to verify one transaction therefore becomes the constant time for any number of transactions verified. Therefore, if a multi-signature verification scheme were to be implemented for Schnorr and evaluated, we would expect that Schnorr outperforms ECDSA for verification of batches of 23 transactions or more.

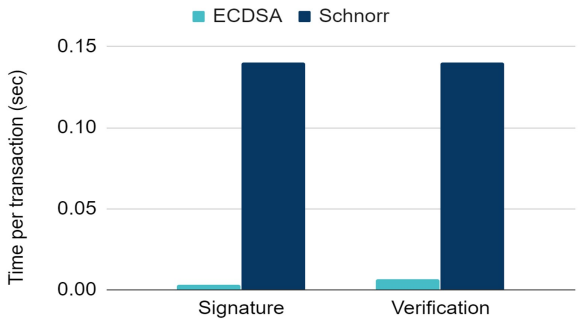

Fig. 2. Latency of Schnorr and ECDSA signatures

B. Experiment 2: Storage Savings using Schnorr

For our second experiment, we wanted to compare Schnorr and ECDSA signatures based on comparing the number of bytes used per block for the two cases. To do so, we set the block limit to 50 at first, then 200 and finally 1000. By block limit we mean the maximum number of transactions inside a block. We noticed that when increasing the block limit, the number of bytes used per block in both of the cases (Schnorr and ECDSA) increases. However, Schnorr uses a smaller number of bytes than ECDSA. If we divide the number of bytes used per block by the block limit, we notice that Schnorr has 463. 75 bytes per transaction on average compared to 705.67 bytes per transaction on average for ECDSA. This tells us that there is 34.28% of reduction in the number of bytes used per transaction on average when Schnorr is used instead of ECDSA. This is actually a significant reduction meaning that Schnorr signatures can actually save a lot of space inside a block.

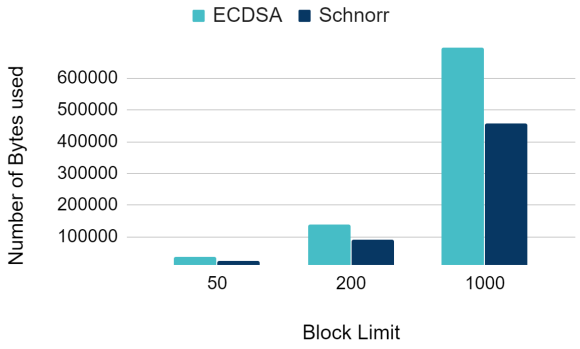

Fig. 3. Storage savings using Schnorr and ECDSA

C. Experiment 3: Increasing difficulty on simulated chain

Central to Bitcoin is the idea of hash problem difficulty. The number of zeros that the block hash needs to start with is a parameter that helps stabilize the network and allow it to reach consensus with certain guarantees. However the slow 10min/block mining rate of Bitcoin also reduces the transaction throughput significantly. In this experiment we varied the hashing problem difficulty in order to test whether there is a difference in trend when using Schnorr DSA vs the ECDSA.

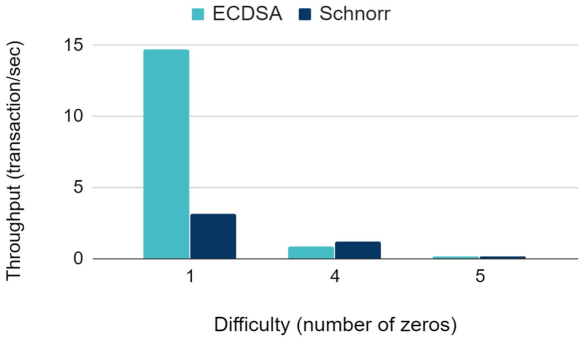

Fig. 4. Throughput of ECDSA and Schnorr signatures according to the change in difficulty

For ECDSA, we increased the difficulty (number of zeros) from 1 to 4 and noticed that the throughput suddenly dropped from 14.7 to 0.89 transactions per second, which is actually a 93.95% decrease in throughput. Then, moving difficulty from 4 to 5, the throughput decreases by 82%. However, for Schnorr we noticed that the throughput decreases by 61% when moving the difficulty from 1 to 4 since it drops from 3.18 to 1.24

transactions per second. This decrease in throughput is actually way smaller than the one for ECDSA. Then, when moving the difficulty from 4 to 5, the throughput decreases by 88.28% since it drops suddenly from 1.24 to 0.15 transactions per second. This is actually similar to the decrease we noticed for ECDSA when when increasing the difficulty from 4 to 5. From this analysis, we concluded that although ECDSA starts with a much higher throughput than Schnorr (14.7 compared to 3.18 transactions per second), it is less consistent. This is because it highly drops at the beginning when we increase the difficulty from 1 to 4. Therefore, Schnorr is consistent since the throughput decreases gradually.

D. Experiment 4: Increasing block size

One naive approach to increasing Bitcoin's transaction throughput was to increase the block size (ie. the number of transactions allowed per block). However as aforementioned this approach often led to forks developing in the block chain. In our experiment we increase the block size from 10 to 90 and test for both ECDSA and Schnorr DSA to see whether or not consensus can be maintained. We ran our blockchain simulation for 50 blocks and measured the maximum number of blocks for which the blockchain was in consensus for (ie. only one block was mined at some height before it was gossiped to and accepted by all nodes in the network). From the graph, we notice that for a limit of 10 transactions, the longest consensus for both Schnorr and ECDSA signatures was the same which is 8. However, when increasing the limit number of transactions to 90, we notice a significant increase from 8 to 31 of the longest consensus for Schnorr and a decrease from 8 to 3 for ECDSA. Therefore, the increase of the block size led to an increase of the consensus for Schnorr signature, which is a good result. The results confirmed that under ECDSA, simply increasing

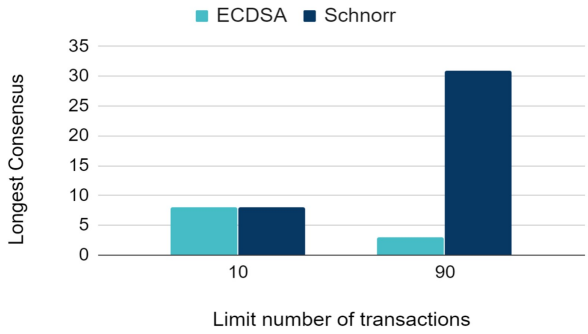

Fig. 5. Longest consensus of Schnorr and ECDSA signatures

the block size would introduce forking and we observed a decrease in the longest consensus count. On the other hand, under Schnorr DSA the longest consensus count increased to 30/50 blocks when we increase the block size. This suggests that by changing to Schnorr DSA, blockchain developers can increase the block size limits in order to increase transaction throughput.

VI. Future Work

A. Hyperparameters

Several hyperparaemeters should be tuned in order to optimize the Schnorr and ECDSA blockchains. Our study was conducted on the assumption that the coordinate of the optimal blockchain in hyperparameter space would coincide for Schnorr and ECDSA. However, since the verification speeds of Schnorr and ECDSA differ dramatically, we think that perhaps a better evaluation would have been to compare the best possible performance attainable with Schnorr versus ECDSA. During our study, we found that the following hyperparameters should be taken into consideration when tuning:

- Hashing problem difficulty
- Block size
- Number of peers per node
- Nounce distance

To elaborate on the parameters here that were undiscussed previously in our study, the number of peers per node is a integer count for how many peers each node gossips to. In the context of the blockchain network we used for our experiments, our peers per node count was 2. Several prerequisites for the network such as connected and minimum cycle size should also be considered. As an unproven conjecture, we observed that more peers equals more gossips on the network communications layer. If the policy towards gossiped transactions involves verification, this may prove to slow down and overwhelm nodes from being able to receive new block broadcasts, thus affecting consensus. Finally the nounce distance is the integer distance between each node's start nounce when mining for blocks. We think that for certain difficulty settings and network sizes, larger or smaller distances may be better also. Of course here we model nodes that are not allowed to chose their own start nounces and assume that the range of valid nounces (nounces subject to computational bounds that produce unique block hashes) is from 0 to infinity. These hyperparameters should be varied and checked for affect. The best set of hyperparameters given the choice of digital signature algorithm should be selected and the consequent blockhcain performances compared. This way the maximum performances of Schnorr versus ECDSA could be demonstrated.

B. MuSig technology

From the experiments we did, we realized that Schnorr signatures employed in single signature transactions save up a little more space for more transactions, but this is not enough. In fact, there is more that attracts the Bitcoin development community. In order to reach to a higher level of scaling, we may think of the possibility of utilizing multi-signature transactions. This is done by a mathematical property present in the Schnorr signature algorithm that allows for aggregation [6]. Schnorr signature algorithm enable the aggregation of multiple public keys into a single public key leading the group to produce a unique digital signature representing multiple independent ones [6]. This eliminates the need for multiple

keys and signatures, so that the blockchain network uses and records a unique public key and signature representing the whole multi-signature scheme [6]. Therefore, the knowledge of the original public keys for participants is not required for verifiers and can be provided with the aggregated key. MuSig technology is introduced and has already been proposed for testing. It is actually a key aggregation scheme for Schnorr signatures [8]. This technology offers provable security, which means that signers are offered the flexibility to take part of multi-signature using ordinary keys without required knowledge of the production and control of these keys [9]. Also, MuSig generates small and constant size signatures that show similar to the original single-signer Schnorr signatures in the eyes of verifiers [9]. It is actually not necessary to give verifiers the details of signer composition with the use of MuSig, which prevents thefts and enhances privacy [9].

VII. Conclusion

We proposed Schnorr signature as a solution to reduce the number of bytes used inside each block of the blockchain, and by that we were solving the scalability problem of blockchain. Schnorr signatures have a smaller length per signature compared to ECDSA, are non-malleable, provably secure and linear. We actually integrated Schnorr signatures in our own blockchain that already uses ECDSA signature in order to evaluate them and compare their performances. From experiments done on both signatures, a lot of conclusions were made about Schnorr. We took some static and simulation measurements, where the results obtained from them led us to make some conclusions on the efficiency of Schnorr signature. From static results we learned that Schnorr provides some storage savings, where we found that there is almost 34% of reduction in the number of bytes used per transaction on average when Schnorr is used instead of ECDSA. However, Schnorr is way slower than ECDSA for the sigining and verifying functions. From the simulation measurments, we concluded that although Schnorr signature had a more consistent behavior in throughput than ECDSA, it actually did not provide significant enhancement to the throughput. We concluded that Schnorr is actually a good approach to make some space savings inside a block, however, the big scaling comes from implementing Schnorr multi-signatures MuSig, which provides key aggregation but also more privacy to the blockchain.

References

[1] J. Göbel and A. E. Krzesinski, "Increased block size and bitcoin blockchain dynamics," in *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, Nov 2017, pp. 1–6.

[2] U. Nadiya, K. Mutijarsa, and C. Y. Rizqi, "Block summarization and compression in bitcoin blockchain," *2018 International Symposium on Electronics and Smart Devices (ISESD)*, pp. 1–4, 2018.

[3] N. S. Jeremy Rubin, Manali Naik, "Merkelized abstract syntax trees."

[4] Wikipedia, "Schnorr signature," https://en.wikipedia.org/wiki/Schnorr_signature.

[5] S. Nakov, "Ecdsa: Elliptic curve signatures," "https://cryptobook.nakov.com/digital-signatures/ecdsa-sign-verify-messages".

[6] B. Musser, "Bitcoin development: Schnorr signatures," "https://edge.app/blog/bitcoin-development-schnorr-signatures/", 2019.

[7] P. Wuille, JonasNick, and T. Ruffing, "Schnorr signatures for secp256k1," "https://github.com/sipa/bips/blob/bip-schnorr/bip-schnorr.mediawikiMotivation".

[8] "The musig schnorr signature scheme," "https://tlu.tarilabs.com/cryptography/musig-schnorr-sig-scheme/The_MuSig_Schnorr_Signature_Scheme.html#musig".

[9] A. Poelstra, "Musig: A new multisignature standard," "https://blockstream.com/2019/02/18/en-musig-a-new-multisignature-standard/".

[10] R. Garg, "schnorr-python," "https://github.com/vihu/schnorr-python/blob/master/naive.py".

VIII. Attribution

	Aicha Lawand	Albert Hsueh
Blockchain code and experiment framework		x
Integration test		x
Cloud deployment and experimentation		x
Results Analysis		x
Presentation and Report writing	x	x
Problem and Solution Proposal	x	
Study of ECDSA and Schnorr signature algorithms	x	
Integration of Schnorr and ECDSA signatures in the blockchain code	x [10]	

TABLE I
Attribution of each team member in the project. Code: https://github.com/ahsueh1996/Schnorr_Blockchain