Secure Digital Service Payments using Zero Knowledge Proof in Distributed Network

1st Harikrishnan M
TIFAC-CORE in Cyber Security,
Amrita School of Engineering,
Coimbatore,
Amrita Vishwa Vidyapeetham, India
harikrishnanm1711@gmail.com

2nd Lakshmy KV
TIFAC-CORE in Cyber Security,
Amrita School of Engineering,
Coimbatore,
Amrita Vishwa Vidyapeetham, India
kv_lakshmy@cb.amrita.edu

***Abstract*—Performing a fair exchange without a Trusted Third Party (TTP) was considered to be impossible. With multi party computation and practices like Proof-of-Work (PoW), blockchain accomplishes a fair exchange in a trustless network. Data confidentiality is a key challenge that has to be resolved before adopting blockchain for enterprise applications where tokenized assets will be transferred. Protocols like Zcash are already providing the same for financial transactions but lacks flexibility required to apply in most of the potential use cases of blockchain. Most of the real world application work in a way where a transaction is carried out when a particular action is performed. Also, the zero knowledge proof method used in Zcash, ZKSNARK has certain weaknesses restricting its adoption. One of the major drawbacks of ZKSNARK is that it requires an initial trust setup phase which is difficult to achieve in blockchain ecosystem. ZKSTARK, an interactive zero knowledge proof does not require this phase and also provides security against post quantum attacks. We propose a system that uses two indistinguishable hash functions along with ZKSTARK to improve the flexibility of blockchain platforms. The two indistinguishable hash functions are chosen from SHA3-finalists based on their security, performance and inner designs.**

***Index Terms*—Zero Knowledge Proof, Blockchain, Data Confidentiality, Fair Exchange, Witness Indistinguishability**

I. Introduction

An exchange is considered to be fair if and only if at the end of the exchange, both participants receive the item they expect and neither obtain any additional information about the other participant's item. A common example could be a purchase of an item in exchange of money or signing a contract where the signed terms are agreed. Assume a scenario where Alice want to purchase a product *M* from Bob for *x* coins. Alice will make the payment for product *x* if it satisfies predicate $P(M) = 1$. The major challenge that is to make sure that no honest will be cheated in the process. Such an exchange is coined as fair exchange. Informally, this mean the following things: that Alice will not get product *M* if she did not pay *x* coins to Bob and Bob will not get *x* coins if the product *M* is not genuine or/and Alice did not get the product. For decades it was considered that such a fair exchange is not possible without a Trusted Third Party (TTP) [1]. But distributed trustless networks like Bitcoin proved that a fair exchange can in fact be performed without a TTP. Cryptocurrencies like Bitcoin replace the TTP by a distributed network, where the validators or more commonly called miners maintain a shared data structure, that is the blockchain where transactions are stored. Such a setup works because of protocols like Proof-of-Work (PoW) and Proof-of-Stake (PoS) that makes it economically infeasible for the verifiers to cheat.

In addition to simple payment transactions, many blockchain platforms like Ethereum provide more flexible transactions, normally mentioned as smart contracts, that let users write contracts that preform payments depending on the criterion mentioned in the program. By using Hashed Time Lock Contracts (HTLC), a special type of smart contract, users can design an up-front resolution for the challenge of fairly exchanging digital goods. Initially, the buyer Alice can write a smart contract, where Alice will load x coins in to the blockchain platform which will assure the transfer of coins if Bob sends a proof that the predicates are satisfied. Once the smart contract is deployed over the distributed platform, either Bob can activate the payment ow of x coins by publishing P to the blockchain, or Alice can transfer coins back to her account after a certain time frame.

Along with improving scalability of blockchain platforms, data confidentiality is a major area of study among the blockchain researchers and developers. In the recent years many solutions have been bought to solve the issue of scalability right from adopting light weight Proof-of-Work protocols to concept of transchain. We are focusing upon the issue of securely transferring data across the blockchain network which in our opinion is the crucial challenge ahead before blockchain's adoption into enterprise and military applications. Protocols like Zcash that assure data confidentiality through a cryptographic technique called Zero Knowledge Proof(ZKP). Zcash using Zero-Knowledge Succinct Non-interactive Argument of Knowledge(ZKSNARK) is the widely adopted Zero knowledge proof because of its high scalability and good security parameters.

Through this paper, we are proposing a zero knowledge schema that will provide the flexibility of the ZKP based blockchain applications needed for many of the markets where blockchain is expected to be adopted in coming years.

978-1-5386-9533-3/19/$31.00 ©2019 IEEE

We are adopting an interactive ZKP called ZKSTARK that is considered to be secure enough to withstand quantum computer attacks. Also, we propose an idea of using two indistinguishable hash functions that can be integrated along with zero knowledge proofs protocols to make the platform flexible to be implemented in use cases where transactions has to be performed based on service or yes/no query.

This paper is structured as follows: First a study on the weakness and challenges of existing models are analyzed in Section II. Section III is core of the paper that deals with different techniques that can be used to solve the challenges of existing models. Section IV explains the proposed system model and Section V gives analysis report of methodologies used in Section III and Section IV. Finally, Section VI concludes the paper.

II. Existing System

In this section, we try to formalize the existing systems for carrying out digital transactions through blockchain. Bitcoin, the first and most famous cryptocurrency is a pure Peer-to-Peer electronic cash system that provide a platform to send and receive money directly from one participant to another without a trusted institution, say like Visa, MasterCard or PayPal.

By using digital signatures, network timestamping [2] and protocols like PoW Bitcoin achieve a system with fair exchange without a TTP. Bitcoin however is not fully anonymous as the transaction content is shared publicly across the network. The sender and receiver addresses are pseudonymous and hence can be tracked down. Thus the basic Bitcoin model cannot be used for enterprise and military applications where data content has to be in secret. There has been a widespread research to come up with a blockchain network which provides data confidentiality. Platforms like Zcash [3] leverages such a blockchain by using Zero-Knowledge Succinct Non-interactive Argument of Knowledge (ZKSNARK) , a Non Interactive Zero Knowledge (NIZK) scheme.

A. ZKSNARK

ZKSNARK [4] proofs can validate the correctness of computations without performing the exact calculations and the verifier will not gain any information what was executed but just that whatever that was executed is done correctly. For example, let us consider the following transaction:

$f(x_1, x_2, a, b, p_a, p_b, v) = 1$ if and only if x_1 and x_2 are the root hashes, a and b are sender and receiver addresses respectively and p_a, p_b are proofs that affirm the balance of a is at least v in x_1 and they hash to x_2 instead of x_1 if v is moved from the balance of a to the balance of b. It is relatively feasible to validate the calculation of function f if all inputs are provided. Hence, we convert function f into a SNARK proof where only x_1 and x_2 are public and (a, b, v, p_a, p_b) are called as the witnesses [5]. The zero knowledge property provides the verifier an ability to determine that the prover knows some witness that turns the root hash from x_1 to x_2 in a way that does not violate any requirement on correct transactions, but gets no information about the participants involved in transaction as well as about the content that is been transferred.

B. Commitment Schemes in ZKP

Commitments are utilized in Zero Knowledge Proofs for two fundamental purposes: first, to permit the prover to take an interest in "cut and pick" proofs where the verifier will be given a decision of what to realize, and the prover will uncover just what relates to the verifier's decision. Duty plans permit the prover to indicate all the data ahead of time, and just uncover what ought to be uncovered later in the confirmation. Commitments are additionally utilized in zero knowledge proofs by the verifier, who will regularly indicate their decisions early in a dedication. This permits zero knowledge proofs to be formed in parallel without uncovering extra data to the prover. Zcash uses Pedersen commitment scheme for its Zero knowledge commitments.

C. Pay to Script Hash(P2SH)

Pay to Script Hash transactions allow to transfer coins to a script hash instead of a public key hash. In Bitcoin P2SH, the receiver must provide a script matching the script hash and inputs to the script which executes to true in order to receive the payment. Using P2SH, users can transfer coins to an address that is secured in different ways without knowing anything about the details of how the security is set up. In fair exchange problem , the buyer can write a script to pay whoever provide the preimage of hash output of the key. If the user to transfer x coins to address and the required key K was hashed using a secure hash function $H()$ to obtain hash output Y ,then the transaction could look like:

Transfer x coins to address that provide $H()$ preimage of Y

D. Hashed Time Lock Contracts Using ZKSNARK

Consider a scenario where Alice wants to buy a digital good G through a Distributed Crypto Platform P. Bob, a merchant is ready to sell the same good G through P. A fair exchange can be performed by writing a smart contract S with following flow.

Bob will first encrypt solution S_p to S_e using key K and broadcast it. Along with it, Bob will also provide hash Y of the K using a secure hash function $H()$. Finally, Bob will also announce a ZKP proving the veracity of the digital good. This way, Alice understand that Bob indeed know a Key K that is used to generate S_p. Now Alice can write a smart contract that will transfer m coins to whoever who provide the preimage of Y using same hash, $H^{-1}(Y)$. (Refer Figure 1)

E. Challenges in Existing System

In this subsection, we are attempting to primarily analyse the Zcash implementation. An in-depth analysis on issues with ZKSNARK system is formulated. Further, with an example, the lack of flexibility of current Zero knowledge proofs platforms is highlighted.

A) **Issues with ZKSNARK**

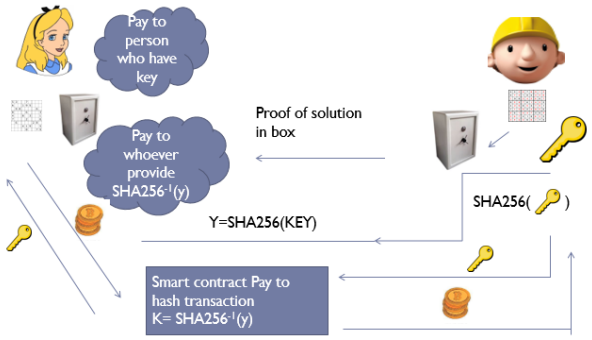

Fig. 1. HTLC with ZKSNARK Proof Setup.

a) **Initial setup phase**
ZKSNARK is a NIZK proof. As of now, the only efficient way to perform NIZK proof is through a trusted setup phase. In the trusted setup phase, few attributes called Common Reference Strings (CRS) are chosen. Traditionally, the CRS parameters are provided by a trusted party. Since there is no TTP in blockchain platforms, the conventional approach cannot be applied. In the actual model of Zcash, the CRS parameters were chosen by the verifier. But then the verifier could manipulate give the CRS parameters in such a way that the proof will reveal some information of the content. In the latest test models of Zcash, approaches are made to choose the CRS through a secure MPC scheme. Even though this is more secure that letting any one of the two involved parties choose CRS parameters alone, there is no formal proof or verification mechanism to check whether the setup has been manipulated or not.

b) **Vulnerable to Post Quantum attacks**
ZKSNARK uses public key cryptosystem on its core. Public key cryptosystems like RSA and ECDSA are vulnerable to quantum computers by using Shor's algorithm [6].

B) **Lack Flexibility** Another issue with current versions of ZKP for blockchain is the lack of flexibility of these schemes that is restricting applying the same to many of the real use cases of blockchain. In scenarios where a transaction has to be performed based on a service and when tokenized assets has to be transferred based on a yes/no trivia, the existing model fails to achieve properties of zero knowledge. For example, consider a scenario where Alice tells Bob that she will pay him if he pays her electricity bill. Now Bob after paying the bill, Bob will get the receipt and constructs a proof P similar to one explained in Section II-D. Here when Bob gives a proof P to Alice that the bill is indeed paid, Alice know that the her is paid even before writing a smart contract. This is because in this case Alice need not see the receipt, rather getting the proof P that the bill is valid is enough for her to conclude that her electricity bill is paid. In such service based payments, the present models fail to achieve fairness. Most of the enterprise and military applications work on service based schemes and hence are not adopting the existing ZKP blockchains. (Refer Figure 2)

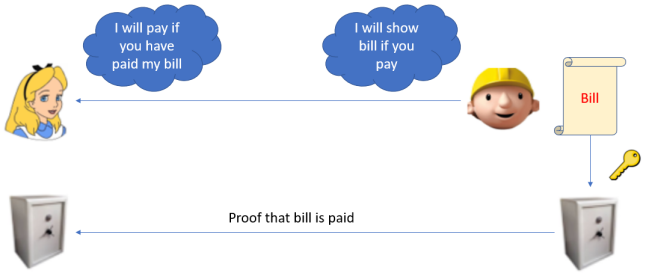

Fig. 2. Digital service payment in existing model failure

III. Important Cryptographic Terminologies

A. Scalable and Transparent ARgument of Knowledge (STARK)

STARK [7] systems can be executed with, or without, zero knowledge (ZK), and may be designed as either interactive or non-interactive protocols. We are using STARK with ZK as an interactive protocol to replace the current versions of ZKSNARK. Interactive proofs do not require a trusted setup phase and hence STARK overcomes one of the major disadvantages of SNARK based systems. Previous interactive ZKP systems were extremely slow in order to be used in real world scenarios. STARK on the other hand was built after massive improvements in the field of Interactive Oracle Proofs (IOP) that use Fast Reed-Solomon codes to achieve high scalability [8]. Also, STARK systems runs with a logarithmic complexity for both prover and verifier hence is advantageous in terms of throughput and also because each parties takes similar amount of time to perform computations, one cannot perform Denial of Service (DoS) on the other.

STARK systems are built on the basis of hashing and information theory whereas SNARK was using ECDSA. Hashing theory is thus far considered to be immune to post quantum attacks whereas public key cryptosystems are susceptible to Shor's quantum algorithm. (Refer Table I)

B. Zero Knowledge Proof

Zero knowledge proof protocol is a cryptographic technique where one user proves to another user that he/she knows a value v without revealing any information about v except the fact that he/she know v.

1) Definition 3.2.1:: An interactive proof of knowledge system for relation R is a pair of algorithms (P, V) satisfying:
1. Completeness: $\forall(x, w) \in R$,
$Prob(V_{P(x,w)}(x)accepts) > 1 - v(n)$
2. Soundness: $\exists M \quad \forall P' \quad \forall x \quad \forall w'$,

TABLE I
COMPARISION OF EXISTING ZKP METHODS

	Prover Scalability	Verifier Scalability	Transparency	Post-Quantum Security
hPKC	Yes	Only repeated computation	No	No
DLP	Yes	No	Yes	No
IP	Yes	No	Yes	No
MPC	Yes	No	Yes	Yes
IVC + hPKC	Yes	Yes	No	No
ZK-STARK	Yes	Yes	Yes	Yes

$Prob(V_{P'(x,w')}(x) accepts) < Prob(M(x,w';P') \in w(x)) + v(n)$

where x represents the common input and n is its length. The auxillary input of prover P is denoted by w and that of verifier V is denoted by y. $v(n)$ represents the negligible probability of the probabilistic Turing machine.

2) Definition 3.2.2:: Proof system (P, V) is zero knowledge (ZK) over R if there exists a simulator M which runs in expected polynomial time, such that for any probabilistic polynomial time V', for any $(x, w) \in R$, and any auxiliary input y to V', the two ensembles $V'_{P(x,w)}(x, y)$ and $M(x, y; V')$ are polynomially indistinguishable.

C. Witness Indistinguishable Proof (WIP)

A two party protocol in which participant a uses one of several secret witnesses to an NP assertion is witness indistinguishable if participant B cannot efficiently determine which witness A is using [9].

1) Definition 3.3.1:: Proof system (P, V) is witness indistinguishable (WI) over relation R if for any probabilistic polynomial time V', any large input x, any $w_1, w_2 \in w(x)$, and for any auxiliary input y for V', the ensembles, $V'_{P(x,w_1)}(x, y)$ and $V'_{P(x,w_2)}(x, y)$, generated as V's view of the protocol, are indistinguishable.

D. Indistinguishable Hash Functions

We are calling two hash functions $G()$ and $H()$ as indistinguishable if there exist no efficient method to distinguish distributions $G(x)$ and $H(x)$ for random x. Informally, this mean that given a hash output *Y*, the verifier should not be able to identify whether the hash was generated from *G()* or *H()*.

IV. PROPOSED SYSTEM

In this section we are going to propose a ZK protocol for service payments. Bob will write a Witness Indistinguishability proof to Alice proving that he know two values x and z, where x is the product and G is a random value. Bob also sends a hash output Y to Alice. If predicate $P(x = 1)$ is true, then $Y = G(z)$ otherwise $Y = H(z)$. In this model, Alice cannot determine if the predicate $P(x = 1)$ is true or not. Alice therefore performs a transaction saying to provide coins to the $G()$ pre-image of Y. (Refer Figure 3).

Here we achieve a fair exchange by using zero knowledge proof combined with computational indistinguishability between the two hash functions $G()$ and $H()$.

The merchant picks key *K* and encrypts the information S_p to S_e the buyer wishes to buy. Then the key *K* is hashed using a strong and secure hash function $G()$ to generate Y'. Also, the merchant takes a random garbage value R and hashes in with $H()$ to generate *Y"*, where $G()$ and $H()$ are indistinguishable.

Now using ZKSTARK system, the merchant interacts with buyer and proves a composite statement:

S_e is an encryption of an input that satisfies the Buyers program AND merchant know a value that will give the pre-image of Y.

Here Y is either Y' or Y''. Since Y' and Y" are indistinguishable, the buyer cannot determine which on it is. Note that the Key is encrypted using hash function *G* and a random garbage value is encrypted using *H* hash function. So the buyer initially wanted to buy an input for his program, but now want only the G hash pre-image of Y. By seeing the output Y, the buyer cannot determine whether the hash function used was $G()$ or $H()$ because of the witness indistinguishability characteristic of the two hash functions used. Hence the buyer writes the following smart contract:

The buyer makes payment to the following ScriptPubkey:

1. $OP\_G()$
2. $< Y > OP\_EQUAL$
3. OP_IF
4. $< SellerPUBLICKEY >$
5. OP_ELSE
6. $< block\_height{+}100 > OP\_CHECKLOCKTIMEVERIFYOP\_DROP$
7. $< BuyerPUBLICKEY >$
8. OP_ENDIF
9. $OP\_CHECKSIG$

The outcome of this script is that the merchant will receive the coins if he / she provides the G hash pre-image of Y and digitally signing with his / her key. To avoid locking the coins in the smart contract forever, the else statement is written to return the coins to buyers address if the merchant does not provide the *G* hash pre-image of Y within a particular time frame. Hence, when the merchant receives the payment, he/she has to disclose the information that the buyer needs in order to decrypt the answer, in this instance the G hash pre-image of Y.

Now consider the previously discussed electricity bill payment problem. By using the current schema, Bob, the prover will encrypt the bill B_p to B_e using a key K_1. He also takes a random garbage value G_p and encrypts it to G_e with the another key K_2. In the next step, Bob transfers either S_e or G_e to Alice. Bob, then hashes the key K_1 with a secure hash function *G()* to produce a hash output Y_1 and also hashes the key K_2 with a hash function *H()* to produce a hash output Y_2 where *G()* and *H()* are indistinguishable hash functions. Bob will send $Y \in Y_1|Y_2$. By seeing Y, Alice cannot determine

whether it is Y_1 or Y_2. The solution is valid if and only if the value received is Y_1. Now, Alice can write a smart contract to transfer to the address that provides the $G()$ pre-image of Y.

If a Non Interactive Zero Knowledge (NIZK) proof is used, the buyer could perform DoS attack on the merchant since the computation for proof is done before buyer writes the smart contract. This can be solved by using an interactive ZKP. Also from Section III-A, we understood more benefits of using STARK, an interactive protocol for Zero Knowledge systems. Hence in the proposed model, ZKSTARK based proofs are generated instead of ZKSNARK.

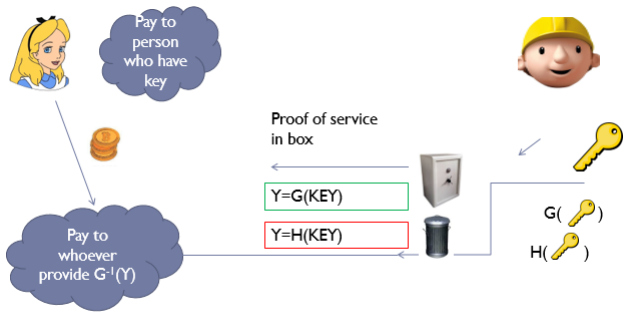

Fig. 3. Modified protocol using ZKSTARK proofs and indistinguishable hash functions.

V. Indistinguishable Hash Functions

In this section, we are performing analysis on various hash functions that are available in order to determine two hash functions that are highly indistinguishable from each other.

We carried out the task of determining indistinguishable hash functions in three stages by analyzing individual security of hash functions in the first round, followed by architectural and inner design comparisons in round two and finally evaluate performance of hash functions on different platforms that is required to be scalable in blockchain platforms where millions of transactions are executed every second.

STAGE 1: Individual Security of Hash Functions

Before going ahead with any hash function, it is of paramount importance to make sure that the hash function is secure individually. Choosing a weak hash function that is vulnerable to some common attack could mean that anyone could find the pre-image of output derived from that hash function. We have followed the SHA-3 hash competition test results and concluded the 5 finalists of SHA3 competition to be the most secure separately [10], [11]. They are Blake, Grostl, JH, Keccak and Skein. All five withstand most of the common attacks on hash functions and are relatively strong against side channel attacks as well.

STAGE 2: Design Analysis of Hash Functions

A) **Basic architecture**
In this stage, we went through the basic architecture of all the five shortlisted hash functions. All of them can produce varying output with size 224, 256, 384 and 512 bits. There were no internal patterns on the output that could provide some information about the hash function used.

B) **Inner design and side channel analysis**
Even if there is no internal pattern on the output, hash functions could be distinguished based upon the inner designs. For example, by performing a side channel analysis on CPU usage, two similar looking hash functions can have different CPU usage.
The inner designs of the hash functions are a major component in determining the side channel indistinguishability. Thus it is important to have our two indistinguishable hash functions to have same inner design. Among Blake, Grostl, JH, Keccak and Skein, Grostl uses the S-box from AES and is the only S boxed based design. JH and Keccak core are logical operations and Blake and Skein functions are ARX model [12], [13]. If we choose Grostl as one of the hash functions, it will have a distinguishable inner design with the other hash function chosen. Hence we eliminate Grostl and conclude in end of stage 2 to use either pair Blake and Skein as two hash functions or choose JH and Keccak.

STAGE 3: Performance Test

In order to meet the requirement of future applications of blockchain where millions of transactions will happen per second, it is important to use hash functions that provide high performance. Among the pair of {Blake, Skein} and {JH, Keccak}. Blake and Skein are providing faster performance when compared to JH and Keccak in different platforms for both long and short messages.(Refer Table II)

TABLE II
Performance of SHA-3 Finalists in different platforms

Platform	AMD64	X86	ARM-NEON	32bit RISC
BLAKE-512	High	Medium	High	Medium
BLAKE-256	Medium	High	High	High
Grostl-512	Medium	Low	Low	Low
Grostl-256	Low	Low	Low	Low
JH	Low	Medium	Medium	Low
Keccak-512	Low	Low	Low	Low
Keccak-256	Medium	Low	Medium	Medium
SKEIN	Medium	High	High	High

Thus after performing analysis on individual security, inner design and performance, we conclude to use Blake and Skein as the two indistinguishable hash functions. (Refer Figure IV)

VI. Conclusion

By using ZKP protocols, data confidentiality can be obtained in a blockchain environment. ZKSTARK does not require a trusted setup and is also resistant to post quantum attacks whereas ZKSNARK fails in both. Hence using ZKSTARK overcomes the major disadvantage of ZKSNARK based systems that require a trusted setup and also by not using public key cryptosystem, increasing the security of the

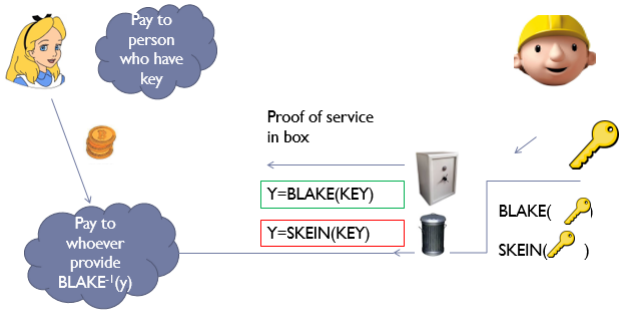

Fig. 4. Proposed Model with ZKSTARK proof and BLAKE and SKEIN as indistinguishable hash functions.

blockchain. Thus distributed ledger technology based platforms can be made implemented in most of the enterprise and military use cases by using two hash functions that indistinguishable along with ZKSTARK proof. Even though ZKSTARK is more secure, ZKSNARK is still much faster method. Improvements in Interactive Oracle Proofs (IOP) has to be made to make ZKSTARK based blockchain to scale for the demands of future use cases.

REFERENCES

[1] Pagnia, Henning, and Felix C. Grtner. On the impossibility of fair exchange without a trusted third party. Technical Report TUD-BS-1999-02, Darmstadt University of Technology, Department of Computer Science, Darmstadt, Germany, 1999.
[2] Irving, Greg, and John Holden. "How blockchain-timestamped protocols could improve the trustworthiness of medical science." F1000Research 5 (2016).
[3] Sasson, Eli Ben, et al. "Zerocash: Decentralized anonymous payments from bitcoin." 2014 IEEE Symposium on Security and Privacy (SP). IEEE, 2014.
[4] Ben-Sasson, Eli, et al. "Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture." USENIX Security Symposium. 2014.
[5] Mohr, Austin. "A survey of zero-knowledge proofs with applications to cryptography." Southern Illinois University, Carbondale (2007): 1-12.
[6] Schmidt, Arthur. "Quantum algorithm for solving the discrete logarithm problem in the class group of an imaginary quadratic field and security comparison of current cryptosystems at the beginning of quantum computer age." Emerging Trends in Information and Communication Security. Springer, Berlin, Heidelberg, 2006. 481-493.
[7] Ben-Sasson, Eli, et al. "Scalable, transparent, and post-quantum secure computational integrity." Cryptol. ePrint Arch., Tech. Rep 46 (2018): 2018.
[8] Ben-Sasson, Eli, et al. "Fast Reed-Solomon interactive oracle proofs of proximity." LIPIcs-Leibniz International Proceedings in Informatics. Vol. 107. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.
[9] Feige, Uriel, and Adi Shamir. "Witness indistinguishable and witness hiding protocols." Proceedings of the twenty-second annual ACM symposium on Theory of computing. ACM, 1990.
[10] Chang, Shu-jen, et al. "Third-round report of the SHA-3 cryptographic hash algorithm competition." NIST Interagency Report 7896 (2012).
[11] Mukundan, Puliparambil Megha, et al. "Hash-One: a lightweight cryptographic hash function." IET Information Security 10.5 (2016): 225-231.
[12] Aumasson, Jean-Philippe, et al. "Sha-3 proposal blake." Submission to NIST (2008).
[13] Ferguson, Niels, et al. "The Skein hash function family." Submission to NIST (round 3) 7.7.5 (2010): 3.