I. J. Computer Network and Information Security, **2019, 6, 22-28**
Published Online June 2019 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2019.06.03

Blind Payment Protocol for Payment Channel Networks

Zhengbing Hu
School of Educational Information Technology, Central China Normal University, China
E-mail: hzb@mail.ccnu.edu.cn

Ivan Dychka, Mykola Onai, Yuri Zhykin
Faculty of Applied Mathematics,
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Ukraine
E-mail: {dychka, onay, zhykin}@pzks.fpm.kpi.ua

Received: 16 January 2019; Accepted: 24 March 2019; Published: 08 June 2019

***Abstract*—**One of the most important problems of modern cryptocurrency networks is the problem of scaling: advanced cryptocurrencies like Bitcoin can handle around 5 transactions per second. One of the most promising solutions to this problem are second layer payment protocols: payment networks implemented on top of base cryptocurrency network layer, based on the idea of delaying publication of intermediate transactions and using base network only as a finalization layer. Such networks consist of entities that interact with the cryptocurrency system via a payment channel protocol, and can send, receive and forward payments. This paper describes a formal actor-based model of payment channel network and uses it to formulate a modified payment protocol that can be executed in the network without requiring any information about its topology and thus can hide information about financial relations between nodes.

Index Terms—Bitcoin, cryptocurrency, networks, second layer protocols, micropayments, payment channels, privacy.

I. Introduction

Cryptocurrency networks have been around for almost 10 years and are quickly becoming an integral part of modern technological stack. Bitcoin, being one of the most advanced cryptocurrency systems, has the highest adoption level, usability characteristics and a large infrastructure, namely multiple implementations of "light" clients (i.e. "wallets") for different mobile operating systems, payment processors, exchanges etc. Further adoption of the Bitcoin technology is blocked by a number of problems, the most important of which are network throughput and transaction privacy.

A. Network Throughput

Network throughput is one of the most important unsolved problems in all modern cryptocurrency systems. Since all Bitcoin-derived networks achieve their immutability by storing complete history of transactions in a hash-chain of Merkle trees [1], their throughput T, measured by transactions per second, is bound by network parameters

$$T = \frac{S_b}{S_{tx} P_b},$$

where
S_b – block size, bytes,
S_{tx} – transaction size, bytes,
P_b – period between blocks, seconds.

Using this equation with Bitcoin network parameters $S_b \approx 1\,Mb$, $S_{tx} \approx 600\,B$, and $P_b \approx 600$ s, it's throughput can be approximately calculated as

$$T \approx \frac{10^6 B}{600\,B \cdot 600\,s} \approx 2.7 \text{ TPS},$$

which is extremely low compared to Visa's average 1700 TPS. In order to compete with such centralized payment systems, Bitcoin will have to increase its throughput by more than 600 times.

As will be shown in the next subsection, such scale cannot be achieved on the base network level without undermining some of the core properties of Bitcoin system.

B. Existing Solutions

This subsection describes existing scaling solutions in terms of their numeric efficiency and discusses their disadvantages in terms of undesired consequences.

During the last few years, the following solutions were proposed:

1. *Transaction size optimization*. Transaction size optimization is an obvious solution that gradually changes internal structure of transaction data to replace or remove redundant fields. One of the most important such optimizations is an

introduction of Segregated Witness protocol update, or SegWit [2], which allows to exclude large portion of transaction, namely its signature, from block data and store it in an external structure for accessing on demand. This also lays a groundwork for moving other data to external structures, for example transaction verification scripts, as described in MAST proposal [3]. Despite being very important for the network in general, this solution is sub-linear since its scaling effect has a lower bound (that of minimal amount of data required for a transaction to describe a transfer of funds within the system), and thus can hardly be used even to double the network throughput.

2. *Block size increase*. Increasing block size is another scaling solution that allows for more transactions to fit in the block. This solution is linear, since increasing the block size by a factor N increases the network throughput by the same factor. The main disadvantage is that the same factor applies to the speed at which global transaction database grows in size, and since this solution is linear, the factor will have to grow alongside the demand, gradually excluding network nodes that do not have enough storage to keep up and eventually leading to centralization of the network on a few most resource-rich nodes which will immediately start to play the role of "central banks", controlling and changing the rules of the system, thus undermining one of the most important properties of Bitcoin network.
3. *Block period decrease*. Decreasing block period is yet another linear scaling solution that is very similar in effect to block size increase, i.e. it has all the same problems of increased load on rule-verifying nodes in the network and eventual centralization as a consequence, but it also decreases the stability of the network consensus layer, since block propagation takes time and shorter periods between blocks lead to frequent reorganization of the block chain, which is a dangerous attack vector for the network.
4. *Second layer protocols*. The most promising solution to the scaling problem is a so-called "second-layer" payment protocol, which uses "first-layer" cryptocurrency protocol for settlement only, while processing transactions in a different way. Core property of a second-layer protocol solution is that it is super-linear, i.e. single first-layer transaction can be used to settle any amount N of second-layer transactions, where N does not have any upper bounds other than reasonable time spans. The only changes to first-layer protocol are various optional locking mechanisms which do not affect regular first-layer transactions in any way and the security of such a protocol is based on the security of the first-layer protocol and the stability of the locking schemes. The most important disadvantage of such protocol is the need for user to store intermediate, second-layer transactions, which imposes the requirement for intermediate data to be "non-toxic" in cryptographic sense, i.e. do not contain any information about the cryptographic secrets used.

II. Related Work

Few payment channel network models have been described in multiple publications on the topic: original Lightning Network publication [4] discusses implementation details in the Bitcoin context, Bolt anonymous payment channel proposal [5] and Eltoo simplified payment channel construction [6] provide formal specification of payment channel protocol, but none of them consider the network cross-channel protocols.

Lightning Network paper [4] describes source routing as the main routing scheme for the network. Source routing [7] is one of the optional routing schemes used in TCP/IP protocol suite and relies on sender explicitly specifying the addresses of the routers for every hop on the packet's intended path through the network. Source routing used in Lightning Network is extended with Sphinx [8] onion wrapping mechanism as described in [9].

Routing is an open problem in Lightning Network research, since every hop on the payment path represents a financial relation between two network entities and it is undesirable to expose any such information for financial system to be secure.

The routing protocol we propose in this paper eliminates the need to announce existing channels between entities by replacing the source routing mechanism with table-based routing mechanism [10] similar to the one used in TCP/IP by default, except the routing tables are one-time use only and constructed dynamically for every transaction that happens on the network. The PDF document should be sent as an *open file*, i.e. without any data protection.

III. Payment Network Model

We propose an actor-based model, which describes classes of network entities as well as their abilities in compromising security of other entities. This model provides a common terminological base and is useful both for design and analysis of the payment channel networks.

Payment channel network is a graph $(\mathcal{N}, \mathcal{C})$, where $\mathcal{N}$ is a set of entities interacting with a base cryptocurrency network, and $\mathcal{C}$ is a set of payment channels.

For each transaction executed within the payment channel network we assign particular roles to nodes in the network graph, namely, we call a path in graph a payment path, starting node of a payment path – a sender, ending node of a payment path – a receiver, nodes on the path – intermediaries and nodes outside the path – eavesdroppers. We call network graph with roles assigned actor graph and use this notion for some of the

security arguments. We describe network actors in details in subsection III.C.

There are two sets of distributed algorithms defined in the context of network graph model. The first set of algorithms represents the interactions between nodes directly connected by the payment channel edge and is described in subsection III.A. We call this set of algorithms payment channel protocol. The second set of algorithms represents the interactions between the nodes on the particular payment path. We call this set of algorithms payment protocol, since all the payments within the network are executed using this protocol even if more direct payment channel protocol would suffice. We describe payment protocol in details in subsection III.B.

Finally, we call the collective metadata about Payer-Payee relations and financial flows within the network payment topology, since this information can be derived from network topology viewed in the context of node/edge semantics.

A. Payment Channel Pprotocol

Payment channel protocol consists of three distributed algorithms, that are executed between two nodes in a network graph and involve interacting with cryptocurrency network, as well as with each other:

1. **SETUP** – initialization algorithm, which locks funds a_0 and b_0 from entities A and B on both sides of the channels respectively and creates a joint account which forms a channel state

$$S_0 = (a_0, b_0),$$

   which on high level can be described as a pair of positive numbers that represent the amounts belonging to each entity within this particular channel.
2. **UPDATE** – main component of the payment channel protocol, that is executed every time entity on either side of the channel wants to transact. The basic idea behind this algorithm is to update state of the channel

$$S_i = (a_i, b_i),$$

   to

$$S_{i+1} = (a_i + d, b_i - d).$$

   representing a transaction of the amount d from A to B or from B to A, depending on the sign of d, in such a way that state S_i can no longer be successfully used to perform **CLOSE**.
3. **CLOSE** – finalization algorithm that takes a current state $S_i = (a_i, b_i)$ and results in a transaction that sends the amounts a_i and b_i to entities A and B respectively, thus spending the funds locked during **SETUP** phase and terminating the channel contract.

B. Payment Protocol

Payment protocol consists of three distributed algorithms executed between peers, which are assumed to be connected with channels within the network, in order to create and propagate payments along the constructed payment paths:

1. **REQUEST** – initialization algorithm executed by the receiver of the payment that creates a payment identification structure, which can be passed to the sender to initiate a payment; the result of the algorithm is a request structure

$$r = (D_R, P_R),$$

   where
   D_R – destination of the payment,
   P_R – public protocol-specific value.
   of which the latter is used by all members of during forwarding;
2. **SEND** – initialization algorithm executed by the sender of the payment after payment request structure has been received in order to finalize the request and initiate the forwarding phase; the algorithm takes a request structure $r = (D_R, P_R)$ as an input, constructs a payment path

$$P = (I_0, I_1, \dots, I_n),$$

   and starts a forwarding process using a public part of the request structure P_R as a forwarding identifier;
3. **FORWARD** – main algorithm executed by each intermediary on the channel path during propagation of the payment that allows entities to transfer funds from one of their channels to another.

C. Network Actors

We distinguish four main classes of network entities relative to a process of propagating a single payment:

1. *Receiver* $\mathcal{R}$ – entity at the end of the payment path that requests a payment by executing a **REQUEST** network protocol and expects to receive funds from Sender for that request;
2. *Sender* $\mathcal{S}$ – entity at the start of the payment path that constructs a payment path and initiates a transaction by forwarding the payment to the first intermediary $\mathcal{I}_0$;
3. *Intermediary* $\mathcal{I}_i$ – entity on the payment path that performs a FORWARD network protocol thus transferring funds across channels from intermediary $\mathcal{I}_{i-1}$ to intermediary $\mathcal{I}_{i+1}$;
4. *Extern* $\mathcal{E}_i$ – entity outside of payment path that can monitor public network state as well as visible network traffic.

The figure 1 shows an actor graph representation of a small network: here $\mathcal{S}$ and $\mathcal{R}$ represent *Sender* and *Receiver*, while $\mathcal{E}_i$ and $\mathcal{I}_i$ – *Externs* and *Intermediaries* respectively.

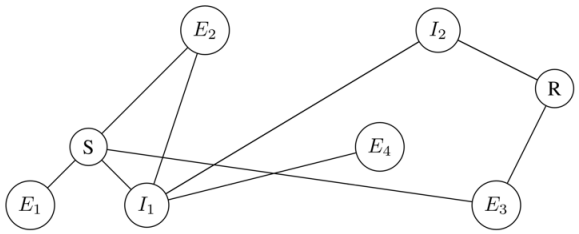

Fig.1. Network actor graph

IV. Blind Payment Protocol

The payment protocol model described in subsection III.B assumes that every node in the network knows its topology: this information is essential for the execution of the **SEND** algorithm of the payment protocol. While various approaches can be implemented for hiding the payment details [5][9], the network topology data remains accessible by all actors in the network and can be easily used for extraction of information like which entity is in direct economic relation with which (based on existence of payment channel) etc.

We presume that hiding payment topology is necessary in order to achieve payment security and confidentiality and can be directly implemented by hiding network topology. In this section we propose a payment protocol in the context of hidden network graph information. For each node on the payment path this protocol requires no knowledge of network topology other than its direct neighbors within the network graph. In order to do this, we first describe the hidden graph network model, using the terminology introduced in section III.

We call this protocol blind payment protocol due to the fact that neither Sender, nor any of the Intermediaries knows about the exact set of nodes on the payment path.

A. Hidden Graph Network Model

We call hidden graph network a network the topology of which is hidden from all of its members, i.e. connections are established either via some external medium or by using hiding peer-discovery protocol similar to that of limited directory querying, implemented in Tor network [11]. Every node in the network graph knows only about its direct peers and is assumed to keep this information private.

Considering a hidden graph payment network, it is apparent that every entity on the payment path knows only its direct predecessor and successor: since it doesn't know if either of those are Sender or Receiver respectively, or just Intermediaries, no further information can be inferred.

Figure 2 demonstrates hidden graph network as seen by node $\mathcal{E}_1$. Every node in the network knows only about its direct peers, so from its point of view, besides these peers and a potential receiver it knows about, there can be any number of other nodes as well as arbitrary connections between them.

Our payment protocol adapts the protocol described in section III to the hidden graph network in order to achieve payment topology hiding.

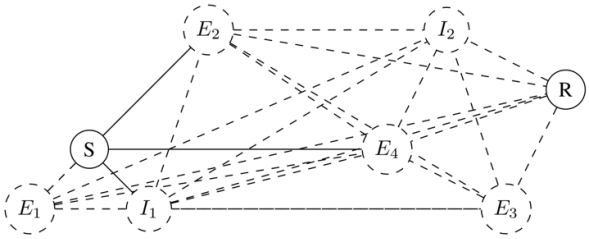

Fig.2. Hidden graph network

B. Payment Protocol with Blind Path Lookup

Modified payment protocol is defined as a set of algorithms that wrap algorithms of the underlying protocol. We move lookup component of the original **SEND** algorithm into a separate **LOOKUP** algorithm, since it no longer can be executed by the Sender alone and requires interaction of all the nodes on the payment path.

This **LOOKUP** algorithms performs two important functions:

1. Collect information required for construction of the routing header of the payment packet (generated cryptographic material).
2. Construct temporary lookup table entries for all the nodes on the ephemeral path – every node on the path extends its routing table with the information on where to send a packet if it comes from the sender of given lookup query.

Protocol algorithms are adapted from an existing protocol and described below:

1. **REQUEST** (figure 3) – non-interactive algorithm executed by *Receiver* to construct a data packet that allows for payment to be correctly forwarded to it:

   a) *Receiver* generates a random secret S_R, compute lookup token

   $$\mathrm{T_R} = \mathrm{H}(\mathrm{S_R}),$$

   where H is a secure hash function.

   b) *Receiver* executes underlying payment protocol's **REQUEST** algorithm computing the regular request packet R.

   c) *Receiver* provides the extended request packet (R, T_R) to the *Sender*. The values S_R and T_R must be stored until the payment for this request is received.

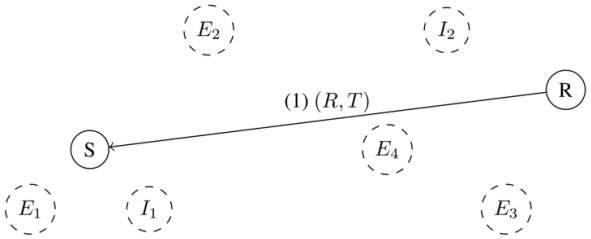

Fig.3. Blind payment, step 1 – Payment request

2. **LOOKUP** (figures 4 – 5) – interactive algorithm initiated by *Sender* to determine which of its peers can forward the payment to the destination:

   a) *Sender* broadcasts a path lookup message containing the lookup token T_R to all of its peers $\mathcal{P}_i$. Since it is reasonable to assume that if there exists a path at all, it is within some predefined radius N , standard peer-to-peer lookup techniques can be used, like giving a message an alive counter, initialized to N , which is decremented on each step in order to prevent message flooding.
   b) Every node within the lookup radius N decrements the lookup query liveness counter and forwards the message to its peers.
   c) The lookup succeeds when one of the peers of the given *Intermediary* can provide a value S such that $H(S) = T_R$, in which case the value S is reported back to the origin of lookup message as an indicator of successful search.
   d) Each *Intermediary* must store a reverse mapping $M: \mathcal{P} \to \mathcal{P}$ of which one of its peers provided the correct S_R for lookup message from which peer. During the reporting, every node extends its temporary routing table with record, that maps given lookup request to the neighbor that reported result.

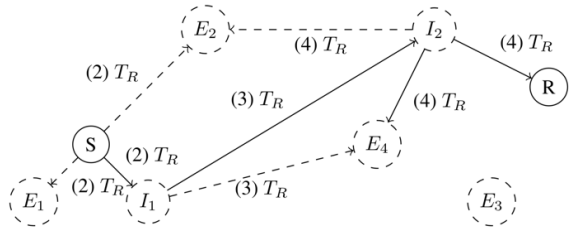

Fig.4. Blind payment, step 2 – Path lookup

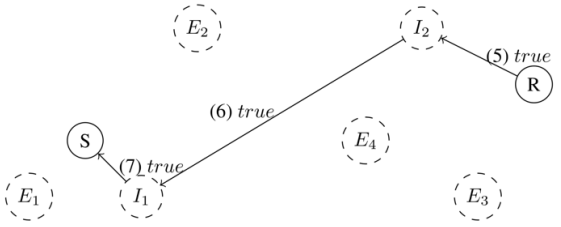

Fig.5. Blind payment, step 3 – Path report

3. **SEND** (figure 6) – interactive algorithm executed by *Sender* in order to construct a virtual payment path and initiate forwarding the payment:

   a) *Sender* executes **LOOKUP** algorithm to verify the existence of the path between it and the *Receiver* and ensure required mappings are created along the path.
   b) When the *Sender* receives S_R such that $\mathrm{H}(\mathrm{S_R}) = \mathrm{T_R}$, from its peer $\mathcal{P}_k$, it initiates the underlying payment protocol's **SEND** algorithm with that peer.

4. **FORWARD** (figure 6) – interactive algorithm executed by every Intermediary to forward the payment:

   a) *Intermediary* receives a forwarded payment from peer $\mathcal{P}_i$.
   b) *Intermediary* applies mapping M to it, obtaining next peer $\mathcal{P}_k = M(\mathcal{P}_i)$.
   c) *Intermediary* executes underlying payment protocol's **FORWARD** algorithm with it.

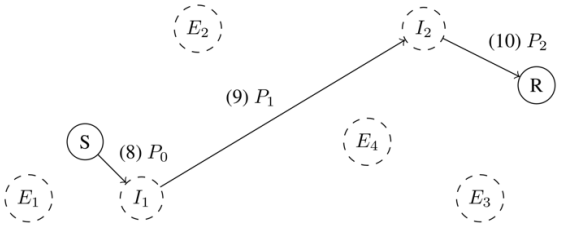

Fig.6. Blind payment, step 4 – Payment propagation

As can be seen from protocol description above, it can be easily implemented on top of existing payment network protocol by modifying the established structure of the payment request and introducing **LOOKUP** algorithm as part of the protocol.

In order to formulate payment topology hiding property, we define lookup function $L: N \times N \mapsto \{0, 1\}$ as follows:

$$L(n_i, n_j) = \begin{cases} 1, & \text{if there exists a path from } n_i \text{ to } n_j, \\ 0 & \text{otherwise.} \end{cases}$$

We claim that it is impossible to restore the graph structure given the proper subset of nodes in the graph $G' \subset G$ and a set of outputs $\{\mathrm{L}(\mathrm{n_i}, \mathrm{n_j}); \mathrm{n_i}, \mathrm{n_j} \in \mathrm{G}'\}$: since not all nodes are known, there is no way to tell whether the existing path between nodes n_i and n_j is direct or through some intermediary $n_k \notin G'$. If this claim holds, it means that within the hidden graph network model, network topology and hence directly related to it payment topology is undiscoverable.

V. Experimental Implementation

One of the existing implementations of Lightning Network protocol suite, LND [12], provides a high-quality modular Golang code base that facilitates experimentation and extension of the protocol. among the

features this particular implementation offers is a support for private channels, i.e. channels that are not advertised on the network and are not included in the network graph view available on all nodes. Private channels are used for direct payments and at node's own discretion during forwarding in order to balance the load if necessary.

We were able to use existing software to build a prototype network node software implementing proposed mechanism on top of private channels in order to test it and verify that all other protocol components continue to work with the new routing scheme in place. Required software changes are localized to packet structure, which is versioned and can be safely changed provided a different version number is used, and routing module itself.

Experiments with existing software show that Lightning Network protocol suite can operate in the payment channel network consisting entirely of private, unannounced channels using a proposed payment protocol with blind routing mechanism. Consequently, proposed protocol can be implemented as an opt-in extension for the existing network node software and doesn't require significant changes to the existing network structure, the only requirement being that enough nodes support this protocol, creating a sufficiently large network within the existing network.

VI. Conclusions

In this paper we described a graph-based model of payment channel network and introduced a hidden-graph model claiming that hiding payment topology (payer-payee relations, channel capacities etc.) is necessary for a secure payment network. Using this model, we proposed a payment protocol that hides payment topology by assuming the context of hidden graph network model.

Described network model provides a common terminology base which can be used to reason about network protocols in terms of graph-theoretic properties and entity interactions.

Proposed blind payment approach has several advantages, most important of which is hiding the payment topology. Proposed path lookup algorithm works in the hidden graph network model by dynamically constructing an inaccessible for the Sender virtual payment path and storing it fragmented on multiple nodes. Another advantage of such approach is the resilience to network changes: dynamically constructed network view does not rely on cached network state data and is consistent within reasonable expectations.

Introducing data flow hiding to the payment protocol requires some trade-offs. Most importantly, all entities within lookup liveness range learn the identifier of the potential Receiver of the payment. Another disadvantage is that dynamic peer-to-peer lookup is inherently slower than lookup in a static locally available routing table. This, on the other hand, is an advantage if our primary concern is payment metadata privacy. Finally, in order to achieve hiding of connections between nodes, constructed path must be stored in a fragment-per-Intermediary way, which imposes more state management on members of the network.

It is possible to implement proposed protocol as an extension of existing network node software and use it as an optional feature if enough supporting nodes are available without disrupting active network operation.

References

[1] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, 2008. [Online]. Available: http: / / bitcoin . org / bitcoin.pdf.

[2] E. Lombrozo, J. Lau, and P. Wuille, Bip 141: Segre- gated witness (consensus layer), 2015. [Online]. Avail- able: https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki.

[3] J. Lau, Bip 114: Merkelized abstract syntax tree, 2016. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0114.mediawiki.

[4] J. Poon and T. Dryja, The bitcoin lightning network: Scalable off-chain instant payments, 2016. [Online]. Available: https://lightning.network/lightning-network-paper.pdf.

[5] M. Green and I. Miers, Bolt: Anonymous payment channels for decentralized currencie, 2016. [Online]. Available: https://eprint.iacr.org/2016/701.pdf.

[6] C. Decker, R. Russel, and O. Osuntokun, Eltoo: A simple laye2 protocol for bitcoin, 2018. [Online]. Available: https://blockstream.com/eltoo.pdf.

[7] J. Postel, "Internet protocol," RFC Editor, STD 5, 1981. [Online]. Available: http://www.rfc-editor.org/rfc/rfc791.txt.

[8] G. Danezis and I. Goldberg, Sphinx: A compact and provably secure mix format, 2009.

[9] R. Russel, Bolt #4: Onion routing protocol, 2017. [Online]. Available: https://github.com/lightningnetwork/lightning-rfc/blob/master/04-onion-routing.md.

[10] F. Baker, "Requirements for ip version 4 routers," RFC Editor, RFC 1812, 1995. [Online]. Available: http://www.rfc-editor.org/rfc/rfc1812.txt.

[11] R. Dingledine, N. Mathewson, and P. Syverson, Tor: The second-generation onion router, 2004. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251375. 1251396.

[12] Osuntokun, C. Fromknecht, and J. T. Halseth, Lnd, 2019. [Online]. Available: https://github.com/lightningnetwork/lnd.

Authors' Profiles

Zhengbing Hu: Ph.D., Associate Professor of School of Educational Information Technology, Central China Normal University, M.Sc. (2002), Ph.D. (2006) from the National Technical University of Ukraine "Kyiv Polytechnic Institute". Postdoc (2008), Huazhong University of Science and Technology, China. Honorary Associate Researcher (2012), Hong Kong University, Hong Kong. Major interests: Computer Science and Technology Applications, Artificial Intelligence, Network Security, Communications, Data Processing, Cloud Computing, Education Technology.

Ivan Dychka: D.S., Professor, Dean of Faculty of Applied Mathematics, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Ukraine. Research Interests: Computer Systems and Networks Software, Automated Control Systems, Intelligence and Expert Systems, Databases and Knowledge Bases, Information Security Software for Computer Systems and Networks.

Mykola Onai was born on December 06, 1986. He received his Bachelor's Degree in Computer Engineering (June 2008) and his Master of Science Degree in Computer Systems and Networks (June 2010), both from the Department of Special Purpose Computer Systems at National Technical University of Ukraine "Kyiv Polytechnic Institute", Kyiv, Ukraine and the PhD degree in Computer Systems and Components in February 2018 from the Computer Systems Software Department at the National Technical University "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine. He is currently an Associate Professor in the Computer Systems Software Department at National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine. His main research interests are Finite Field Arithmetic, Public Key Cryptography, Elliptic Curve Cryptography, Computer Security, Network Security and Hardware Algorithms for Cryptography. Mykola Onai has authored and coauthored more than 40 scientific publications and is inventor of 4 patents.

Yuri Zhykin: Master's program student at National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute". In 2017 received Bachelor's Degree in Software Engineering from Department of A plied Mathematics at National Technical University of Ukraine "Kyiv Polytechnic Institute". Primary interests: Applied Cryptography, Programming Language Design, Compiler Development and Artificial Intelligence.

How to cite this paper: Zhengbing Hu, Ivan Dychka, Mykola Onai, Yuri Zhykin, "Blind Payment Protocol for Payment Channel Networks", International Journal of Computer Network and Information Security(IJCNIS), Vol.11, No.6, pp.22-28, 2019.DOI: 10.5815/ijcnis.2019.06.03

