Privacy-Preserving Multi-hop Payments with Constant Collateral

*Note: Sub-titles are not captured in Xplore and should not be used

1st Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

2nd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

3rd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

4th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

5th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

6th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—This document is a model and instructions for LaTeX. Test for pull. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

In recent years, permissionless cryptocurrencies, have emerged as a novel means to facilitate secure and reliable payments within a decentralized framework, garnering significant attention from both academia and industry. These cryptocurrencies employ a consensus mechanism to verify each transaction, which is then recorded on a publicly distributed ledger known as blockchain. Unfortunately, the widespread adoption of cryptocurrencies is hidered by notable scalability chanllenges. Complex consensus mechanisms, like Bitcoin's Proof-of-work(PoW), and the limited block size of the blockchain contribute to the issue. The theoretical throughput of Bitcoin stands at approximately 10 transactions per second(TPS), with a transaction confirmation time of around 1 hour. In contrast, traditional decentralized payment networks, such as Visa, boast the capability up to 47,000 TPS. Furthermore, the presence of high transaction fees renders small-value payments impractical for cryptocurrency users.

One promising solution proposed to tackle the issue of scalability is the implements of payment channels(PCs). PCs are off-chain payment protocols that enable two parties, who have established a channel,to conduct quick and validated transaction off-chain. To elaborate, the overrall process can be divided into three phases. Firstly, during the channel-openning phase, both users commit a portion of their coins to a shared address as initial funds, which is executed on-chain. In the subsequent channel-updating phase, the involved parties have the flexibility to engage in numerous off-chain transactions. They can adjust the allocation of funds between themselves by generating and exchanging signed transaction message. Ultimately, when the participants opt to settle the channel or encounter a dispute, they initiate the closing process by broadcasting the latest signed transaction to the blockchain. This transaction represents the most up-to-date distribution of funds within the channel.

II. Background

In this section, we provide an overview on the background and the notations used throughout the paper.

A. UTXO model

In this work, we assume the underlying blockchain, like Bitcoin, is based on the UTXO model. Transaction output is the fundamental component of Bitcoin transaction,which is an indivisible Bitcoin currency recorded on the blockchain and recognized as valid by the entire network. The Bitcoin complete nodes tracks all available outputs called *unspent transaction outpus* (UTXO). UTXOs can be any value and once generated are indivisible, a UTXO can only be consumed as a whole in a single transaction. The output consists of two parts,which we represent as a tuple $\theta := \{cash, \phi\}, \theta.cash$ is the output value, and the $\theta.\phi$ is the condition to spend this output, it also called *locking script*. Onesig(U) indicates that the condition required to spend this output is a digital signature. We say that a user U can spend an output only if $\theta.\phi$ contains only a signature w.r.t verification U's public key, if multiple signatures are required, we use MultiSig(U_1, U_2, ..., U_n).

Identify applicable funding agency here. If none, delete this.

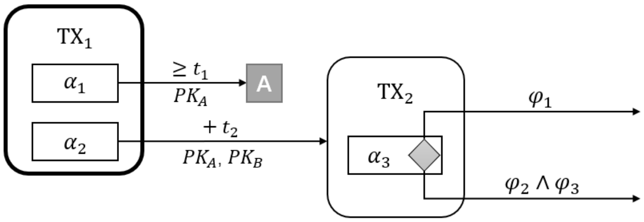

Fig. 1. Left transaction TX_1 is recorded on the blockchain, which has two output. Value α_1 can spent by A, with a transaction signed w.r.t PK_A after t_1 rounds, and the output of value α_2 can be spent by a transaction signed w.r.t PK_A and PK_B but only if at least t_2 rounds has passed through after the transaction being published on the blockchain. The right transaction TX_2 has one input which is the second output of TX_1 with the value α_1. TX_2 has only one output containing α_3 coins, which can spent by a transaction whose witness satisfies the condition $\varphi_1 \vee (\varphi_2 \wedge \varphi_3)$

Transactions consume previously recorded unused UTXOs and create new UTXOs available for future transaction, in this way, the transaction continues in the form of a chain of owners. on the chain, the input of the transaction corresponds to the output of the previous transaction. We denote a transaction as a tuple TX := {id,input,output,timelock,witness}, TX.id$\in \{0,1\}^*$ is the identifier of a transaction and TX.id = $\mathcal{H}$(TX.input,TX.output,TX.timelock), $\mathcal{H}$ is a hash function, modeled as a random oracle. TX.input and TX.output denotes the list of the inputs and the list of new outputs respectively. **TX**.timelock defined as the earliest time a transaction is valid and can be transmitted on the network or added to the blockchain, it defaults to 0 in most transactions. TX.witness$\in \{0,1\}^*$, also called *ScriptSig*, is part of the transaction input to address or satisfy the spending conditions set by the *locking script* on the output. Actually, before being recorded on the blockchain, transactions must go through the consensus mechanism of all nodes. During this period, each node will independently verify the transaction. For specific details please refer to [], we will only briefly describe the key parts: For a transaction(1) The sum of the input value cannot be less than the sum of the output value;(1)For each input, the quoted output cannot exist in any other transaction, it must exist and not be spent;(3) The *ScriptSig* for each input must be validated against the *locking script* for the corresponding output. To put it simply, the transaction must provide valid validation that satisfies the spending conditions of each input.

We use charts to visualize the transaction for a clearer illustration. Rounded rectangles represent transactions, thick-edged rectangles represent transactions that have already been published on the blockchain, and thin-edged rectangles for transactions to be published. The transaction contains at least one box to represent the output of the transaction, and the value in the box indicates the number of coins in this output. On the arrows coming from the output, are noted the conditions under which this output is spent. The public key below the arrow indicates who can use this output; Above the arrow is the timelock for the output(in this work only uses timelock as additional condition, which in practice could be any script supported by the underlying blockchain scripting language). There are two types of timelocks: *relative time lock* and *absolute time lock*, we use "+*t*" to represent the relative time lock, that is, the transaction only vaild at least *t* blocks has passed through after the transaction being recorded on the blockchain; The absolute time lock "$\geq t$" specifies the absolute time point, indicating that the first transaction has passed through *t* round after being recorded on the blockchain. Finally, we use a diamond to represent the relationship of "or" that the output conditions are different, expressed in the symbolic form as $\varphi = \varphi_1 \vee ... \vee \varphi_n$, where φ is the output locking script, written on the arrows of the output. A complete example is given in Fig. 1.

B. Payment channels

The payment channel is opened by two users locking some coins on the blockchain, and then the two parties can make as many quick instant confirmation transactions off-chain, that is, without waiting for transmitting the transaction to the blockchain, as long as the total amount of each transaction is less than the locked value. During the lifetime of the channel, only two transactions will be recorded on the blockchain, one is funding transaction TX_f which used to open the channel, the other is settlement transaction for close the channel. The funding transaction TX_f determines the balance of this channel, and the channel is open when TX_f is broadcast to the network and recorded on the blockchain, both parties update the channel balance by exchanging signed transactions off-chain, these transactions holding the reassigned balances are called state of the channel. Finally, the channel is closed by posting settlement transaction, which is the final state, on the blockchain.In a more detil, there are three operations of the payment channel: *open, update,* and *close*.

1) *Open*: Suppose that Alice and Bob want to create a payment channel, and to do so, Alice and Bob agree on a funding transaction TX_f. The output value of TX_f is $\alpha_A + \alpha_B$, which is the total amount of the channel, and there are two inputs to the TX_f are from Alice and Bob respectively, α_A represents the initial deposit provided by Alice in the channel, and α_B represents Bob's. When TX_f is published on the blockchain, it indicates that the payment channel has been opened.

2) *update*: After the channel is opened, to update the balance status of both sides of the channel, that is to make a transaction, parties exchange signatures on the commit transaction. Each time a new "commit transaction" is exchanged, the previous state of the channel is overwritten and updated to the newer state, while the old state is saved locally. Therefore, only the most recent commit transaction can be "executed". To ensure the security of payments, payment channel use a penalty mechanism to: 1) prevent malicious user from closing the channel by sending an outdated but self-advantageous state to the chain, and 2) enable honest user to acquire all the coins locked in the channel when dishonest user post outdated promised transactions. We use the recent work *generalized channels* [] as the basic structure, which split the "committ transaction" into two parts, the commit transaction

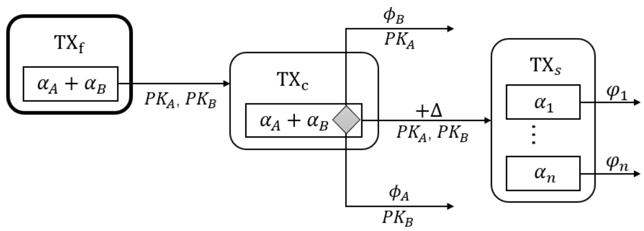

Fig. 2. Details of generalized channel, $(x_1, \varphi_1)...(x_1, \varphi_1)$ are the states of the channel. The output of TX_c is three different conditions, two of which are punishment mechanisms used to prevent dishonest users. The condition ϕ_B is punishment for B, represents the verification of B' revocation secret, When A discovers that B submit an old transaction, it can take all the money in the channel by B' revocation secret, which B has exchanged for A when the old transaction is updated. and in the same way, ϕ_A represents the verification of A' revocation secret that can punish A. The value of Δ is time upper bound that takes to publish a transaction on a blockchain.

TX_c responsible for penalizing and the split transaction TX_s that hold the actual output of the channel. assume that Alice wants to pay $x_A \leq \alpha_A$ coins to Bob. To do this, they create new commit transaction TX_c' and split transaction TX_s', at the same time, they pick the new *secret R'* to represent the commitment of the two parties to the new balance of the channel, then they exchange the *revocation secret r* of the of the previous round of transactions TX_c and TX_s, after that they make the new commit transaction TX_c' and split transaction TX_s' overwrite previous ones. TX_s' represents the updated balance of the channel, and both parties save the previous round of transactions (TX_c and TX_s) with the *revocation secret r* of the secret value in local memory.

3) *close*: Finally, assume both parties agree to close the channel, They can collaboratively submit $\overline{TX_c}$ and $\overline{TX_s}$, which represent the last transaction they agreed, to the blockchain to close the channel. If the dishonest one submit an expired promised transaction, the malicious party can be penalized by closing the channel through the revocation mechanism unilaterally.

C. Payment channel networks

The IEEEtran class file is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

III. Solution Overview

In this section, we present our key idea.

A. Security and privacy goals

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, ac, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

B. Key idea

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, ac, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

IV. Constrution

A. Building blocks

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, ac, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

B. Protocol description

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, ac, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

V. Analysis

A. Security

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, ac, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

B. High level functionality description

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, ac, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

VI. Evaluation

The implementtion and evaluation.

VII. Discussion

Some arguements

VIII. Conclution

Conclude the paper.

REFERENCES

Please number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first . . ."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English. citation first, followed by the original foreign-language citation [6].

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, "Title of paper if known," unpublished.

[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.