Verifying Payment Channels with TLA+

Matthias Grundmann, Hannes Hartenstein
Institute of Information Security and Dependability (KASTEL)
Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany

***Abstract*—A payment channel protocol does not only have to provide the payment functionality, it also has to fulfill security guarantees such as ensuring that an honest party receives their correct balance. For complexity reasons, it is typically difficult to assess the security of such a protocol or to find counterexamples in insecure protocols. In this poster, we present an approach to specify functional as well as security properties for a payment channel protocol in TLA+ and show that a Lightning Network-style protocol fulfills the required properties. In case a counterexample is found, we provide protocol developers with a graphical and intuitive output. We present the challenges we faced and our approach to meeting these challenges.**

I. Introduction

Payment channel networks improve the number of transactions performed per time unit by 'off-loading' transactions from a first layer, typically a blockchain, to a second layer. A payment channel is created by two parties locking funds in a shared account on the underlying first layer. Both parties store the state of how the channel's funds are distributed. The two parties can perform transactions off-chain by updating their shared state to a state with a different distribution of the channel's funds. At any time, each party can close the channel by publishing the latest state on the first layer. The security model of payment channels assumes that the counterparty is untrusted and adversarial. A dishonest party might close the channel in an outdated state that has a distribution of funds favorable for this party. In this setting, a payment channel protocol should guarantee that a party will finally receive their correct balance on the first layer if it follows the protocol. Developing a protocol for payment channels such as the Lightning Network [1] is a challenging task because the proposed protocols are complex and various edge cases need to be considered. Therefore, we address the following research question: How can one verify correctness and security properties of payment channel network protocols in a way that is accessible to protocol developers?

We approach this research question by making use of TLA+. The Temporal Logic of Actions [2] is a logic for specifying concurrent systems and is notated using the language TLA+ [3]. A specification in TLA+ specifies a set of initial states and actions that define state transitions and, thus, define possible successor states. To reason about a specification, TLA+ allows for specifying properties such as invariants that must hold in every possible state or properties that must hold in a series of states. A model checker (e.g., TLC [4], Apalache [5]) can be used to explore all possible states and validate whether the specified invariants and properties hold.

Related Work: Previous work made use of TLA+ to analyze a state channel protocol [6], to reason about the security of smart contracts [7], to analyze the synchronization of the Tendermint Blockchain [8], and to prove properties of a cross-chain swap protocol [9]. The security properties of the Lightning Network's protocol were shown in [10] using a different approach, namely the Universal Composability (UC) framework. While the UC proof can be considered more fundamental and comprehensive, we take an alternative path to provide an intuitive presentation of counterexamples found in insecure variants of a protocol during protocol development.

II. Payment Channels in TLA+

As a use case, we chose to specify a protocol for payment channels based on Bitcoin that is an abstracted version of the Lightning Network's specification [11]. Our specification has approximately 1,200 lines of code and is available online [12]. The protocol's security property is that an honest party finally receives the party's correct balance even if the other party cheats. In this section, we explain the challenges we faced and how we approached them: modeling the underlying blockchain [13] and transactions with hashes and signatures, specifying progress of time, allowing a dishonest party to deviate from the protocol while still keeping the state space explorable, and providing a protocol developer with an intuitive and understandable output in case a counterexample is found.

Specification of the Blockchain. To specify the construction of payment channels, we need a specification of transactions that are used in the protocol and the blockchain. The specification of this underlying layer must, on the one hand, model all aspects that are required by the payment channel protocol and, on the other hand, be as simple as possible so that the specification can be efficiently model-checked. Further, the specification of the blockchain and transactions must be an abstraction of Bitcoin so that counterexamples found using the specification can be transferred to the real world.

To meet these requirements, we follow the UTXO (unspent transaction outputs) model of Bitcoin. A transaction consists of inputs and outputs; inputs reference outputs that they spend; outputs impose conditions which must be met by an input spending the output. A condition can be the requirement to provide a signature matching a given key or to provide a

This work was supported by funding from the topic Engineering Secure Systems of the Helmholtz Association (HGF) and by KASTEL Security Research Labs.

978-1-6654-9538-7/22/$31.00 ©2022 IEEE

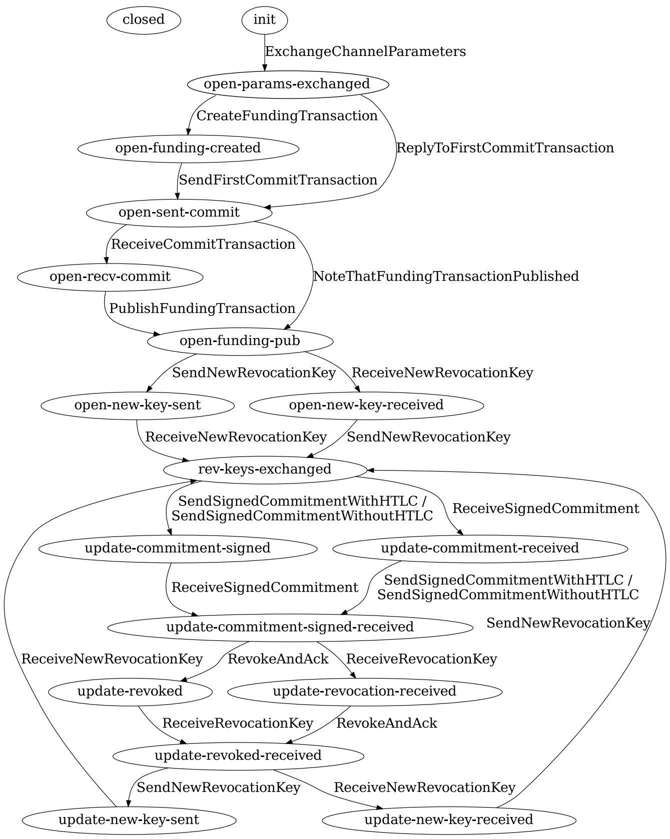

Fig. 1. Possible honest behaviors defined by the module PaymentChannelUser specifying the states (nodes) and actions (arrows) of a user inside a payment channel. For readability, actions leading to the state 'closed' are omitted.

preimage to a given hash value. To model these conditions, we developed an abstraction of signatures and hash functions that models the aspects of signatures and hash functions but is as simple as possible so that model-checking is possible. We model transaction IDs by assigning unique integer values as transaction ID to each newly created transaction and assign a new transaction ID if a transaction is changed. Having these components to model transactions, we model the blockchain as the set of all published transactions.

Specification of the Payment Channel Protocol. A payment channel protocol is a protocol between two users. The specification starts with an initial state in which the ledger contains only one transaction with an output that is spendable by one user. From that state on, the steps by both users allow for the creation, updates, and the closing of the payment channel. We specify the steps of the payment channel protocol by modeling the two users and specifying all possible actions that a user can perform (see Fig. 1). Each user stores the name of the current protocol state, variables and an inventory of transactions that the user can publish on the blockchain. The actions manipulate the user's variables or the ledger or they exchange messages between the users.

Adversarial Behavior. We model faults and adversarial behavior by specifying that one of the users can be dishonest. A dishonest user is explicitly modeled with the abilities that a cheating protocol participant has in the real world: The dishonest user can publish an outdated transaction, create transactions from keys and secrets known to the dishonest user, or stop acting at all. We do not model that a dishonest user sends invalid messages to the honest user because the honest user can detect such faulty behavior by comparing a message against the expected message according to protocol's specification. We also do not model actions that use side-channels that break assumptions of the protocol such as DoS attacks. During model checking, the model checker iterates through all possible executions of the protocol including all possible actions by the dishonest user and the model checker checks whether the specified security properties hold.

Time Flow. The specification uses a weak fairness property which means that users perform actions if they can, i.e. the execution stops only if no further action is possible. Without this property, we could not show liveness properties, e.g., that a channel will finally be closed. We model time as the current height of the blockchain. To model that the protocol can progress quickly or slowly with respect to the growth of the blockchain, we specify that the height of the blockchain grows at arbitrary times between actions of the users. The protocol for payment channels contains requirements that users need to perform an action before a certain point in time, e.g., punish a dishonest user before an outdated output becomes spendable. We model this by specifying that time does not advance further than such a point in time as long as the timely action is required. To reduce the number of states that are explored when model-checking, we specify the flow of time so that time advances directly to these points in time at which certain actions need to be performed.

Understanding Counterexamples. A counterexample is a series of actions that leads to a violation of a specified invariant or property. In a flawed protocol, for example, a dishonest user might be able to receive a part of the honest user's balance which breaks the security property. To support developers in understanding counterexamples and quickly checking a variant of the protocol, we developed a visualization of states of the payment channel protocol (see [14]). The visualization allows a developer to navigate through each intermediate state of the counterexample so that the developer can learn where the specification is flawed.

III. Results and Future Work

We measured the runtime of model-checking in a scenario in which one user opens a payment channel with ten coins, then pays seven coins to other party and then the other party sends back two concurrent payments of two and three coins. The explored state space contains 1,131,490 distinct states and the runtime of TLC is about two hours on a standard notebook. Most counterexamples are, however, found within a few seconds or minutes. The number of states is heavily influenced by the adversarial behavior: If both users are honest, the state space contains only 69,440 distinct states. As future work we would like to further improve the runtime and will address multi-hop payments.

REFERENCES

[1] J. Poon and T. Dryja, "The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments," Tech. Rep., 2016.

[2] L. Lamport, "The temporal logic of actions," *ACM Transactions on Programming Languages and Systems*, vol. 16, no. 3, pp. 872–923, May 1994. [Online]. Available: https://doi.org/10.1145/177492.177726

[3] ——, *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[4] Various. (2021) TLA+ Toolbox. [Online]. Available: https://github.com/tlaplus/tlaplus

[5] I. Konnov, J. Kukovec, and T.-H. Tran, "TLA+ model checking made symbolic," *Proceedings of the ACM on Programming Languages*, vol. 3, no. OOPSLA, pp. 123:1–123:30, Oct. 2019. [Online]. Available: https://doi.org/10.1145/3360549

[6] T. Close, "Breaking state channels with TLA+," Jun. 2020. [Online]. Available: https://blog.statechannels.org/breaking-state-channels/

[7] J. Kolb, J. Yang, R. H. Katz, and D. E. Culler, "Quartz: A Framework for Engineering Secure Smart Contracts," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2020-178, Aug. 2020. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-178.html

[8] S. Braithwaite, E. Buchman, I. Konnov, Z. Milosevic, I. Stoilkovska, J. Widder, and A. Zamfir, "Formal Specification and Model Checking of the Tendermint Blockchain Synchronization Protocol (Short Paper)," in *2nd Workshop on Formal Methods for Blockchains (FMBC 2020)*, ser. OpenAccess Series in Informatics (OASIcs), B. Bernardo and D. Marmsoler, Eds., vol. 84. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, pp. 10:1–10:8, iSSN: 2190-6807. [Online]. Available: https://drops.dagstuhl.de/opus/volltexte/2020/13423

[9] Z. Nehaï, F. Bobot, S. Tucci-Piergiovanni, C. Delporte-Gallet, and H. Fauconnier, "A TLA+ Formal Proof of a Cross-Chain Swap," in *23rd International Conference on Distributed Computing and Networking*, ser. ICDCN 2022. New York, NY, USA: Association for Computing Machinery, Jan. 2022, pp. 148–159. [Online]. Available: https://doi.org/10.1145/3491003.3491006

[10] A. Kiayias and O. S. T. Litos, "A Composable Security Treatment of the Lightning Network," in *2020 IEEE 33rd Computer Security Foundations Symposium (CSF)*, Jun. 2020, pp. 334–349, iSSN: 2374-8303.

[11] Various. (2021) Lightning Network In-Progress Specifications. [Online]. Available: https://github.com/lightning/bolts

[12] M. Grundmann. (2021) Specification of Protocol for Payment Channel in TLA+. [Online]. Available: https://github.com/kit-dsn/payment-channel-tla

[13] M. Grundmann and H. Hartenstein, "Fundamental Properties of the Layer Below a Payment Channel Network," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, ser. Lecture Notes in Computer Science, J. Garcia-Alfaro, G. Navarro-Arribas, and J. Herrera-Joancomarti, Eds. Cham: Springer International Publishing, 2020, pp. 409–420.

[14] M. Grundmann. (2022) Exemplary Output of Visualization of Counterexamples. [Online]. Available: https://github.com/kit-dsn/payment-channel-tla/tree/icbc22/visualization