# **Smart Contract Programming Languages on Blockchains: An Empirical Evaluation of Usability and Security**

BLOCKCHAIN is an emerging technology that is quickly getting momentum in both the community and industry worlds. A blockchain is basically a chain made up of sequential lists of data called blocks that are connected and safeguarded via encryption. Many peers on a network keep copies of this chain. Blockchain technology has play significant impact on a few areas, including finance, cloud computing, privacy, security, etc. Smart contracts is the meaning of the agreement between multiple parties are directly written into lines of code. Smart contracts give trusted transactions and agreements between unidentified groups. Smart contract also facing dares. because its not fully mature. More than 2 Million (40 Million USD) worth of ether were stolen from the DAO network in June 2016 by attacking weaknesses in the DAO design. And also in 2017 Nov, a developer attempted to repair a fault that allowed attackers to steal 32 million dollars from a few multi-signature wallets. we judged the usability and security weakness phases of three domain-specific languages namely Solidity, Pact and Liquidity. The results indicated that Solidity is the most user-friendly language for beginning developers to write smart contracts,

SMART CONTRACT PROGRAMMING LANGUAGES talk about the popular smart contract languages like Solidity, Pact, and Liquidity. Solidity is a programming language designed for creating smart contracts which has syntax close to Ecma (JavaScript). Ethereum is a decentralized, open-source platform for creating smart contracts. Ethereum has its own IDE that called Ethereum Virtual Machine (EVM). Solidity contracts run on EVM. A peer-to-peer network protocol is part of Ethereum. Several networked nodes are responsible for maintaining and updating the Ethereum blockchain database. The same set of instructions are carried out by every node on the network while the EVM is running. Pact is a programming language used to create smart contracts that the Kadena blockchain will carry out. Pact enables developers to design transactional logic that is reliable, fast, and secure for task business operations. For Tezos, Liquidity is a high-level typed smart-contract language. It is a fully typed functional language that closely meets with the security requirements set forward by Michelson.

EMPIRICAL EVALUATION is the study of empirical objective is to identify the usability and security flaws of the smart contract programming languages covered in the past segment. The formal descriptions of three smart contracts were used as the basis for the study for analyzing the usability and examining the kinds of flaws and security gaps that developers might remove from the contracts

1. Experimental Planning (2) Experimental Execution (3) Experimental Results and Analysis (4)Threats to Validity (5) Discussion

RELATED WORK is comparing the programming languages for smart contract development. There isn’t much research that are relevant to evaluating smart contract programming languages or platforms. First research is to compares the platforms Eris, IBM Open Blockchain, Intel Sawtooth Lake Block Stream, Ethereum, and IBM Open Blockchain. According to this research, Ethereum is the best platform for scalability, development, documentation, and support. Another research is that study the use of platforms for smart contracts from different angles. The study compares a sample of 6 platforms. For the two platforms—Bitcoin and Ethereum—a sample of 834 contracts was examined, with each one being identified by application domain. They found that nine design patterns proposed in the study are used by around 80% of Ethereum contracts. The vast majority of the research that is included in this area compares and analyzes various smart contract platforms. There is a lack of empirical information on the potential interactions between developers and domain-specific smart contract programming languages, as well as on the relative merits of their usability and security.

CONCLUSION is the study of smart contract programming is still in its infancy, and there is still a long way to becomes mature. An experiment was conducted to examine the usability and security vulnerabilities of the three domain-specific languages, Solidity, Pact, and Liquidity, as part of the evaluation given. Results of the experiment showed that while Solidity is the easiest language for a beginner developer to code smart contracts