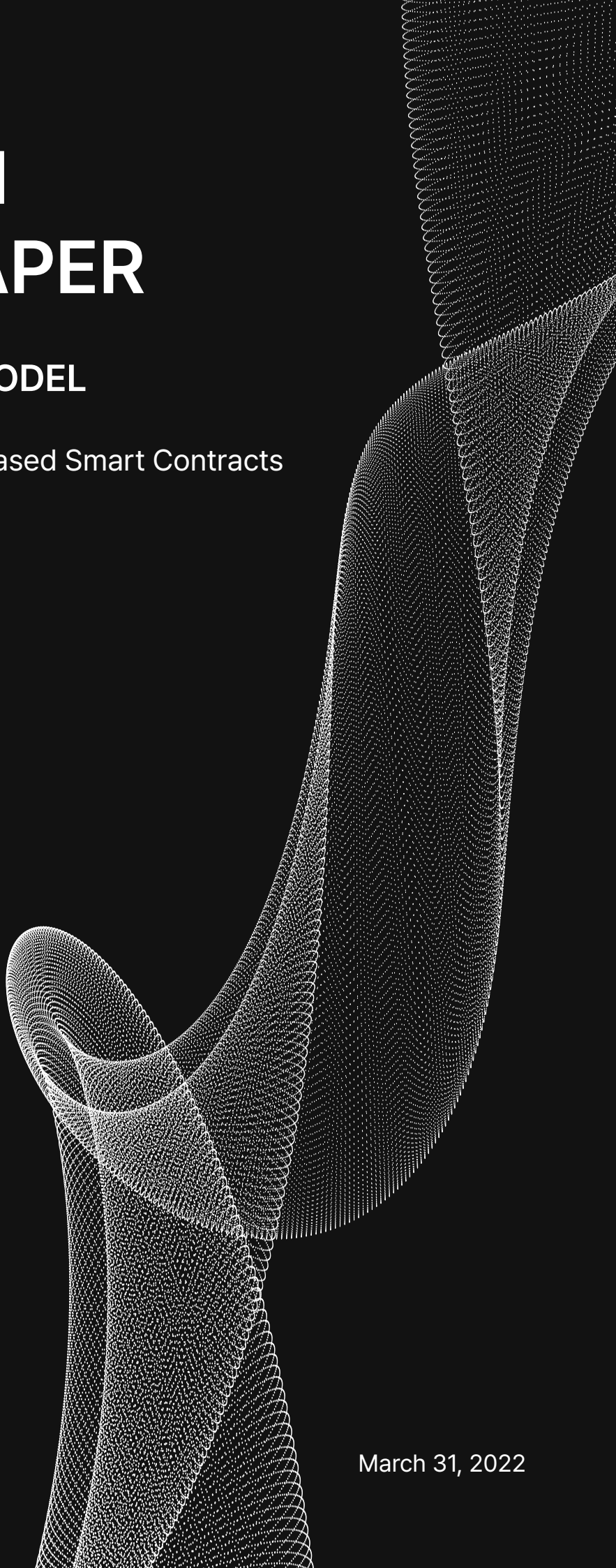
PROTOCON YELLOW PAPER

PART 1 : CONTRACT MODEL

On ‘Contract Model’, Model-based Smart Contracts



Choi, Yoon-il
Bae, Min-hyo

March 31, 2022

Table of Contents

1. Overview

This document defines Protocon's[1] 'Contract Model', which corresponds to smart contracts of general blockchains.
Protocon's Contract Model (hereinafter 'the Contract Model') is a concept similar to smart contracts of general blockchains and is a novel smart contract development methodology which takes every feature of the existing smart contract while supplementing its shortcomings. Such a novel approach serves as the foundation for providing reliability and security to numerous application systems and application services linked to blockchain. Moreover, unlike programming language-based smart contracts, the Contract Model is also excellent in terms of blockchain usage cost, development efficiency and security.
This Yellowpaper begins by introducing the Contract Model, the specification and features of the model. Chapter 2 presents the backgrounds and the need for adopting the concept of 'Model' on the smart contract, and Chapter 3 defines the Contract Model through detailed specification. Chapter 4 describes the features of the Contract Model, and Chapter 5 offers the conclusion of this Yellowpaper.

2. Model

2.1 Model-based Development

Model-based Development or Model-based systems engineering[2] is a methodology mainly applied in developing systems which are highly complex and at the same time require high reliability and security (e.g., aircrafts, military hardware, etc.).

In recent years, a massive amount of capital flowed into NFT or DeFi projects. Such market conditions demand provision of services with high stability and availability, in particular new smart contracts that can be secured from security threats such as hacking. Therefore, an advanced smart contract development methodology which is more stable, reliable and efficient than the VM (Virtual Machine) structure proposed by Ethereum is required. Protocon's 'Contract Model' is a novel smart contract development methodology which meets such needs.

In the model-based development methodology, a model is defined through mathematical specification and a 'Unit Contract Model' is developed according to this definition. Unit Contract Models undergo rigorous testing and screening in terms of reliability and security. These models can be combined to create 'Composite Contract Models', using which numerous reliability and security-qualified application systems and services can be established.

---

[1] Protocon is the blockchain project which aims to establish a protocol-based self-operating digital economy, and is founded on 'Mitum' Blockchain which newly defined PBFT algorithm and smart contract.
[2] "Model-based systems engineering for aerospace,"
https://resources.sw.siemens.com/en-US/e-book-model-based-systems-engineering-aerospace.

2.2 Model-based Smart Contracts

The concept of smart contracts was introduced by Nick Szabo in 1996[3], but it is an undeniable fact that Ethereum is the trigger for the term and concept of smart contracts to become widely known.

Upon being launched in 2015, Ethereum blockchain network continued to operate in an extremely stable manner ever since. While most mainnets which proclaimed their ambition to surpass Ethereum such as Solana or Klaytn experienced network outages due to overload or DDoS attacks, Ethereum maintained extremely stable operation, and so Ether, the cryptocurrency of Ethereum has so far established itself as the second-largest cryptocurrency after Bitcoin.

Of course, large-scale hacking attacks did take place in the Ethereum network; a representative case would be the hacking attack on The DAO fund worth 150 million dollars on February 2016[4]. There are recent cases including hacking attacks on various DeFi projects on the Ethereum network such as Uniswap and dForce. However, it is worth noting that this hacking accident was not caused by a defect in the Ethereum mainnet itself, but mainly by a defect in the Ethereum smart contract.

Ethereum's smart contract was developed using the language named Solidity, which has the following six vulnerabilities.[5]

1) Overflow and Underflows
2) Message Calls and Access Control
3) Reentrancy
4) Short Address Attack
5) Forcing Ether to Contract
6) DoS Attack

Thus, the issue of Ethereum smart contracts having security vulnerabilities continues to be raised. A research jointly conducted by the National University of Singapore and University College London in 2018 discovered over 34,000 vulnerable smart contracts in the Ethereum network.[6] Newly discovered vulnerabilities of smart contract is continuously reported to the Ethereum community's bug bounty program[7]. Also, according to a research paper published

---

[3] Szabo, Nick. "Formalizing and securing relationships on public networks," First Monday, 1997.
[4] After the The DAO hacking attack, Ethereum was hard forked from Ethereum Classic.
[5] "How to Secure Your Smart Contracts: 6 Solidity Vulnerabilities and how to avoid them," https://medium.com/loom-network/how-to-secure-your-smart-contracts-6-solidity-vulnerabilities-and-how-to-avoid-them-part-1-c33048d4d17d.
[6] "34,200 Ethereum Contracts Vulnerable to Hackers, Containing Millions in Ether," https://u.today/34200-ethereum-contracts-vulnerable-to-hackers-containing-millions-in-ether.
[7] "ETHEREUM Bounty Program," https://bounty.ethereum.org.

by IBM researchers under the title ZEUS[8], (based on the analysis using their model check framework) there exists one or more vulnerabilities in 95% of all smart contracts deployed. The problem is, these vulnerabilities do not just remain as vulnerabilities, but may lead to actual hacking attacks worth at least million to billion dollars.

Security issues of smart contract is not restricted to Ethereum. Solana blockchain, which is known for the operation of numerous NFT and DeFi projects, develops smart contract using a language named 'Rust'. On February 2nd, 2022, Solana's cross chain bridge service Wormhole was hacked, resulting in $320 million in harm, and post analysis showed that the hacker took liberty of unpatched Rust smart contracts.[9]

Such issue of smart contract is a extreme danger and sever threat, as the ideals and objectives of blockchain - in which data cannot be forged or falsified, and the uniqueness, ownership and value of digital assets are guaranteed - are destroyed by none other than 'smart contracts' which was created to allow utilization of blockchain for various purposes.

To provide a solution to this situation, we intend to introduce model-based system engineering to the blockchain industry, which is generally applied in the development of systems which require high reliability and security such as aircrafts or military hardware, as the development methodology to provide a more stable and reliable smart contract. We have named this the 'Contract Model.'

3. Protocon Contract Model Specification

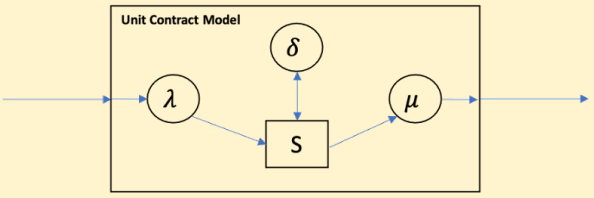
Protocon's Contract Model consists of the following two definitions, the 'Unit Contract Model' and the 'Composite Contract Model.' Unit Contract Model is the basic model which cannot be divided, and Composite Contract Model refers to a model generated through combination of more than one Unit Contract Model.

[8] Sukrit Kalra et al., "ZEUS: Analyzing Safety of Smart Contracts," IBM research, 2018.
[9] "Solana's Wormhole Hack Post-Mortem Analysis," https://extropy-io.medium.com/solanas-wormhole-hack-post-mortem-analysis-3b68b9e88e13.

3.1 Unit Contract Model

$$M=<X, Y, S, \lambda, \delta, \mu>$$

- X: a set of inputs
- Y: a set of outputs
- S: a set of states
- $\lambda: X \rightarrow S$, input transform function
- $\delta: S \rightarrow S$, state transition function
- $\mu: S \rightarrow Y$, output function



[Figure 1] Unit Contract Model Specifications

Unit Contract Model 'M' consists of three sets and three functions.

The three sets indicate input, output and model state. Model state refers to the assembly of data which constitutes the model.

The three functions consist of input transform function, state transition function, and output function. Input transform function transforms input data (parameters) into elements of the state set. State transition function is a function which performs the calculation of transitioning the state through calculation on model state data. Output function is the function which outputs the value in accordance with the data content and format required by state data.

A common model performs a single operation through continuous operation of three functions(λ, δ, μ) of input transform function, state transition function and output function.

Here, we will use 'Mitum Currency'[10] Model as an example to help understand the structure and operation of Unit Contract Model.

'Mitum Currency' Model is a Unit Contract Model which performs operations such as generating tokens to be used on Mitum blockchain and transfer tokens between accounts.

The state set of the Mitum Currency model is a set of data that pairs the number of tokens with the account that holds the generated tokens.

[10] "Mitum currency's documentation," https://mitum-currency-doc.readthedocs.io/en/latest/.

Account	Token Account	Difference
0xa374	258	0
0xb28d	27	0
0xdb8d	102	0

[Figure 2] Example of Mitum Currency's Set of States

For instance, when performing the operation to send 25 tokens from Account 0xa374 to Account 0xb28d, the input data would be

$$x = (0xa374, 0xb28d, 25) \in X$$

, and the result of performance of input transform function λ is as follows.

Account	Token Account	Difference
0xa374	258	-25
0xb28d	27	25
0xdb8d	102	0

[Figure 3] Set of States After the Performance of Input Transform Function λ

The result of performance of state transition function δ is as follows.

Account	Token Account	Difference
0xa374	233	0
0xb28d	52	0
0xdb8d	102	0

[Figure 4] Set of States After the Performance of State Transition Function δ

Depending on the demands, output function μ may choose either the success or failure of operation, or token count of modified accounts as output data.

3.2 Composite Contract Model

$$C=<X, Y, \{M_i\}, \{C_j\}, R>$$

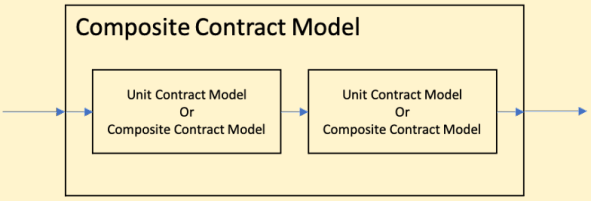
- X: a set of inputs
- Y: a set of outputs
- $\{M_i\}$: a set of unit contract models
- $\{C_j\}$: a set of composite contract models
- R: $< Rxx, Rxy, Ryy >$, relations between contract models

where

$$Rxx \subseteq X \times \bigcup_{k=i\ or\ j} X_k$$

$$Ryx \subseteq \bigcup_{k=i\ or\ j} Y_k \times \bigcup_{l=i\ or\ j} X_l$$

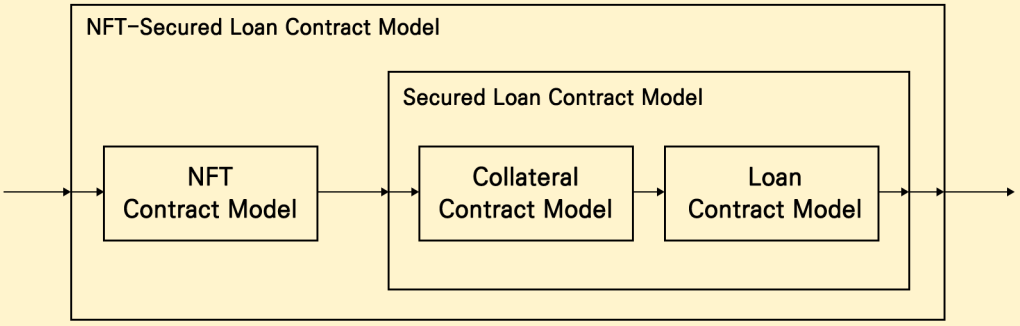
$$Ryy: \bigcup_{k=i\ or\ j} X_k \rightarrow Y^{\Phi}$$



[Figure 5 Composite Contract Model Specification]

Composite Contract Model C consists of 4 sets and 3 relations. The sets are sets of inputs, outputs, unit models constituting the Composite Contract Model and children Composite Contract Models. 3 relations consist of mapping data input from outside the Composite Contract Model to input of internal component model, mapping inputs and outputs between internal models and mapping output of internal model and outputs of Composite Contract Model.

The following is an example of a secured loan Composite Contract Model taking NFT as collateral.



[Figure 6] Example of a Composite Contract Model

'NFT-Secured Loan Contract Model' consists of 'NFT Contract Model' that receives NFT as input and holds NFT in the contract account within the model and 'Secured Loan Contract Model' that generates a loan based on the collateral. Again, 'Secured Loan Contract Model'

consists of ‘Collateral Contract Model’ that generates collateral securities based on the collateral and ‘Loan Contract Model’ that delivers cryptocurrencies to the borrower in accordance with the loan result based on the collateral securities. ‘NFT Secured Loan Contract Model’ includes input and output relations indicated by arrows in the figure above.

As shown above, Contract Model enables creating diverse Composite Contract Models that fulfills market and customer needs by combining sufficiently verified Unit Contract Models.

4. Features of Protocon Contract Model

In this section, we will learn about the features of the Protocon Contract Model.

4.1 Model-based Development

Smart contracts on many blockchains are built with individual programming languages. Solidity, Rust and Haskell are used on Ethereum, Solana and Cardano, respectively. Due to these structural characteristics, all developers are required to learn the appropriate language designated in each blockchain to develop smart contracts, even though they are software development experts. Until the developer learns the concerned language and is equipped with a certain level of development capability, multiple trials and errors are inevitable. Diverse human errors occur during this process, and reliability issues and vulnerabilities occur consequently. Most of Ethereum smart contract vulnerabilities mentioned above have occurred in such processes.

On the other hand, Protocon’s model-based smart contract, in other words, the Contract Model, enables development by stacking up verified Unit Contract Models as components, like making diverse works with Lego blocks. When a new Unit Contract Model is needed, the model can be developed to become independent from the programming language through development by defining the model’s specification, six components of <X, Y, S, λ, δ, μ>.[11] Furthermore, it can greatly prevent the chance of vulnerability from immature or faulty use of the programming language.

For the Contract Model developed in such a manner to be distributed on the blockchain, it shall pass verification and consensus process by nodes constituting the blockchain’s consensus group. This prevents imprudent distribution of Contract Models and allows it to secure the reliability through verification. Consequently, reliability and security are enhanced because only sufficiently verified smart Contract Models are distributed and adopted.

In Ethereum’s smart contract, contracts performing the structurally same function are stored and executed redundantly in multiple blocks for each project. However, contracts requiring

[11]With a GUI environment which would be developed later, it would become able to develop Contract Models without separate programming languages.

the same function call the same unit model in the Contract Model system, consequently enhancing the computing resource efficiency. Above all, it can prevent vulnerability caused by imprudent hardfork of codes.

Furthermore, after these Unit Contract Models are sufficiently developed, it is possible to provide more complex services by combining various functions using unit models through provision in the form of a GUI tool.

By reaching this level, various blockchain application functions can be stably developed through GUI tools without going through complex programming. Through this new development methodology, we propose a solution for the blockchain industry that makes it easier and safer for people to access the blockchain.

4.2 Baremetal

Another differentiated feature of Protocon Contract Model is that smart contract is directly executed on a hardware (Baremetal) without using a virtual machine. Usually, executing a smart contract on a blockchain requires using a virtual machine specified by each blockchain. Blockchains use their own virtual machines, such as EVM (Ethereum Virtual Machine) of Ethereum[12], LLVM (Low Level Virtual Machine) of Solana and KEVM (K Ethereum Virtual Machine) of Cardano.

On Ethereum, Solana and Cardano, codes are written and compiled in different smart contract programming languages of each blockchain, and they are converted into machine codes in the form of bytecodes (virtual machine instruction). And then, these machine codes are executed by an emulator on each virtual machine in the unit of instruction. As is well known, an emulator is a software that imitates functions of a CPU operation of the hardware, and using an emulator is much slower than working with an actual hardware.

On the other hand, the contract model runs on hardware rather than virtual machines, providing better performance than other blockchain smart contracts. Since the Contract Model does not use a virtual machine, hardware performance can be utilized as it is. This alone can dramatically improve the processing power of the blockchain.

4.3 Independence of Programming Language

Protocon's Contract Model enables developing smart contracts without dependency on a programming language.

---

[12] "ETHEREUM VIRTUAL MACHINE (EVM)," https://ethereum.org/en/developers/docs/evm/.

If a developer is developing a new Contract Model, the developer shall first define the model specification, <X, Y, S, λ, δ, μ>, and implement it with a preferred programming language in accordance with the defined specification. Developers who are sufficiently proficient in a particular language will be familiar with the characteristics and security precautions of the language, so they can implement the program much more efficiently and safely than studying and developing a new language. This is a highly important feature that differentiate Protocon Contract Model from other smart contracts that require specific languages.

5. Conclusion

As mentioned above, a massive amount of capital flowed into NFT or DeFi projects. In terms of dealing with digital data assets nature in blockchain, securing reliability and safety is the top priority above all. In this context, a highly stable and available service is required, specifically a smart contract ensuring safety against security threats, including hacking attacks.

Protocon offers the concept of Contract Model in terms of responding to such demands. Already, we have successfully developed most of the functions currently used in the blockchain industry, such as token models, DID models, data models, and NFT models, by applying model-based smart contract development methodologies. And the developed models have been tested on TestNet and are being applied to various projects based on Protocon networks. We will keep introducing diverse new features while reflecting various demands of industries requiring the blockchain.

In conclusion, we present the Protocon Contract Model as an alternative to overcoming fatal disadvantages of VM-based smart contract. This will provide an optimal technical foundation for projects requiring high reliability, fortified security, and outstanding processing performance.

Project Information

Homepage : https://protocon.io/

Github : https://github.com/ProtoconNet

Contact : contact@protocon.io