Cogito Technical Whitepaper

The invention of Bitcoin in 2008 symbolized the advent of cryptocurrency. The succeeding years witnessed a dramatic growing of the blockchain technology. The introduction of Ethereum opened the path for supporting complex business behaviors with cryptocurrencies. The overall momentum of development, however, is now facing a series of key challenges. These include: (1) slow transaction speed; (2) programming barrier of smart contracts; (3) lack of security in smart contracts; and (4) inflexibilities in managing and updating blockchains.

Designed to be the new generation blockchain, cogito leverages the latest artificial intelligence technology to resolve the abovementioned challenges. The fusion of blockchain and AI technologies enables cogito to build a revolutionary cryptocurrency, which support significantly boosted transaction speed, superior accessibility to general users, enhanced security under malicious attacks, and highly flexible operations.

Objectives of COGITO

The challenges facing today’ s blockchains have to be resolved before the ideal of cryptocurrency can really become reality. We believe the artificial intelligence technology, which has received an unprecedented growth in the past decade, provides out-of-box solutions to address the challenges. cogito is designed to be an intelligent chain to unleash the potential power of the blockchain technology. In this section, we briefly review the objectives of cogito.

1.1 Automatic generation of smart contracts

Although smart contracts give blockchains the essential capability to handle scaled commercial behaviors, they need the users to be able to write programs in a given programming language. With cogito, no programing expertise is needed any more for designing smart contracts. The unique code generation technique of cogito allows automatic conversion of an abstract description of a smart contract into an executable program. cogito only requires users to input the core elements (e.g., input, output, and transaction conditions) of a contract with a scripting language. Then a code generator based on a deep neural network is able to automatically convert the script into an equivalent program.

1.2 Secured smart contracts

Smart contract programs may call functions offered by the host system and/or third-party libraries. Also, programs running on different computers in a distributed framework do not provide any guarantee for execution time. Such openness and decentralization are the reflection of the essential spirit of blockchains, but give birth to various sources of security threats. In fact, the lack of security is plaguing the smart contracts. The cogito blockchain is equipped with a power AI security engine consisting of four major components, 1) a rule- based semantic and syntactic analysis engine for smart contracts, 2) a formal verification toolkit to prove the security properties of smart contracts, 3) an AI-based detection engine for transaction model identification and security checking, and 4) a deep learning based platform for dynamic security verification and enhancement.

1.3 High speed transactions

Today all public chains are suffering from the problem of long transaction latency and low transaction throughput. Specifically, it takes over 30 minutes for Bitcoin to finish one transaction, while the transaction throughput of Ethereum is only 10 Transfer Per Second (TPS). In fact, a blockchain depends on a P2P network to validate transactions. Since a transaction needs to be broadcasted to all nodes in a network, the overall latency has to increase as long as more nodes are joining the network. cogito resolves the problem by dynamically selecting a delegation network in which all nodes are voted as delegates of others. All Proof-of-Work (PoW) processing is only allocated inside the delegation network, which only incurs a much smaller latency due to the smaller number of nodes. The selection process is random in the sense that a node is selected with a probability proportional to its Proof-of-Stake (PoS). The online version of cogito will support a throughput of 100,000 TPS.

1.4 Flexible blockchain management

cogito is designed to be a highly flexible blockchain. The flexibility is twofold. First, cogito offers access control and routing services so as to allow seamless integration private chains into a common public chain. Such a feature meets the requirements many industry and government players for authorization, while at the same time allows necessary information flow from a public chain to a private one and vice versa. Second, cogito uses a reinforcement learning framework to optimize its parameters (e.g., consensus mechanism, and transaction configuration) in an evolutionary manner. The optimization paradigm ensures dynamic updating of parameters for near-optimal performance without the risk of incurring hardfork.

1.5 Value adding mining

Perhaps the most criticized part of cryptocurrency is the “waste” of energy in the mining computations. Although it is essential to attach physical value to the cryptocurrency, the mining process does not make any sense out of the world of digital currency. The problem is even worse when now over 70% of the total computing power around of world is dedicated to mining Bitcoins and others. cogito introduces a new mining mechanism in which miners perform the Markov Chain Monte Carlo (MCMC) computation, which is an essential tool for Bayesian reasoning. MCMC based Bayesian computing plays a fundamental role in numerous big data applications such as gene regulatory network, clinical diagnosis, video analytics, and structural modeling. As a result, a distributed network of MCMC computing nodes provide the power for solving real-world compute-intensive problems and thus build a bridge between the values in the physical and virtual worlds.

2. A Brief Review of Technological Innovations in cogito

The design of cogito depends on a large number of innovations in both AI and blockchain technologies. Table 1 summarizes the essential technologies of cogito. Thanks to these technologies, cogito distinguishes itself from its predecessors and initializes a new generation of blockchain.

Table 1. Technological innovations of cogito

Category	Technological Innovations	Objective
Base protocol	1. Random clustering based voting for a network of delegates	Reducing transaction latency
	2. Introduction of a separate control chain	Enabling the interaction between a public chain with private chains and the deployment of security control
	3. Evolutionary parameter optimization	Adapting blockchain design to external usage patterns and environment
Smart contract	1. Automatic program generation	Breaking the barrier of

generation	for smart contracts	programming and making smart contracts accessible to general users
Security	1. Formal verification and deep learning based auditing including smart review of contracts, relational review of contract elements, formal verification of contract security, and transaction arbitration	Identifying potential loopholes and malicious intentions
	2. Binary code checking	Identifying potential loopholes and malicious intentions
	3. Credit score-based trusted gateway and proxy including online-offline data proxy transfer (ORACLE), credit score of public users, and multi-chain-based data routing	Maintaining credibility record for the network
	4. Dynamic security verification and enhancement of smart contracts with generative adversarial network	Ensuring robustness under high-intensity attack
	5. AI secured custody of contracts, mainly for	Enabling long-term financial derivative transactions
Transaction	1. Pattern match	Allowing data exchange and identification of matched contracts for multi-party transactions
	2. Transaction data search engine supporting permission-based index, privacy-based user transaction tracking, and smart grouping based on behaviors of multiple users	Assisting historical data tracing and data mining
Mining	1. Bayesian and deep learning as the PoW computation	Creating universal values from mining

3. Key Technologies of cogito

AI plays an essential role in shaping cogito as an intelligent blockchain. Moreover, cogito is also the product of extensive optimization and extension of the blockchain technology. In this section, we review the key technology breakthroughs made by cogito.

3.1 Hybrid PoS + PoW consensus based on stochastic network contraction

The fundamental reason leading to the excessive transaction latency in today's blockchains is the transmission overhead in a P2P network cannot scale with the number of nodes. The problem is inherent to the P2P based PoW consensus since every transaction is to be broadcast to the whole network. In fact, the latency has to worsen as long as the number of nodes increases. The recently proposed lightning network is devised to solve the above problem. What it does, however, is no more than creating a private channel for the payer, receiver and transferring nodes so that transaction can be completed once a consensus is reached. Such a process lacks a guarantee for the safety, especially when the channel is under malicious attacks, because it takes place off the blockchain.

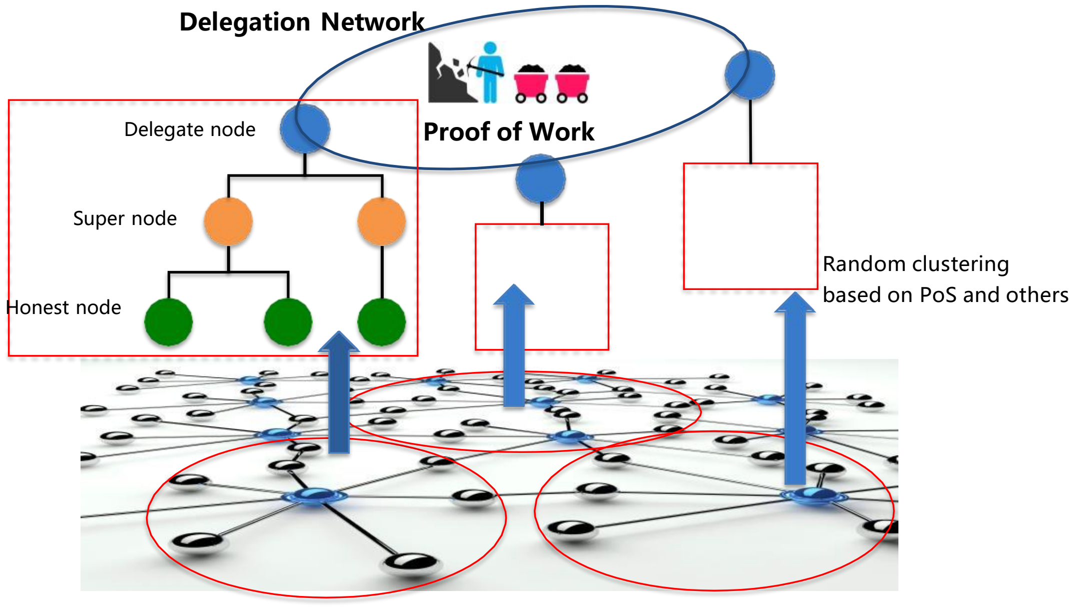


Figure 1. Random clustering based PoS + PoW consensus

An effective out-of-box solution is to introduce a hierarchy into the P2P network so that the broadcast overhead can be contained. The key idea is to contract a network into smaller ones (i.e., create a hierarchy) and perform PoW computation in the contracted network with fewer nodes. The basic idea of cogito’ s proprietary algorithm is illustrated in Figure 1. The hierarchy is created with a distributed random clustering process without centralized control. When the clustering is completed, each node will have a delegate for itself. The selected representatives constitute a new network, coined as delegation network, which has an adjustable number of nodes. Then the transaction is only broadcasted inside the delegation network. The PoW is allocated to the delegate nodes and one such node can further partition its work into smaller jobs and assign these jobs to the nodes voting for it. Note that the clustering is iteratively performed. The probability of a node to be selected as a delegate is proportional to its PoS and other factors.

The distributed random clustering based consensus algorithm of cogito is designed as follows.
A node determines whether it wants to be affiliated with another node based on PoS or other factors.
The node sends a request to neighboring nodes asking them to join it as a cluster, together with a link list of proof certificates to sign on and a request for deposit. If other nodes agree to join the cluster, the certificate will be used for verification and they will be marked as “affiliation committed”. If they do not agree, they will be marked as “competitors” and will not receive information from this cluster.
After becoming affiliated to a cluster, a node calculates the connectivity of neighboring nodes and elects a group of candidates according to certain rules. With a random method the best candidate will be elected and the respective message is distributed to other nodes.
When one node has an enough number of affiliated nodes, it will be marked as “fully- loaded”. Such nodes will compete to incorporate other clusters. Nodes will send invitations to neighboring nodes, asking the latter to join them. Through a multi-signature mechanism, they jointly generate a random seed, which can be used to initialize a Markov Chain procedure. An independent third party will judge who wins the competition by looking at the seed, public and private keys published by both parties, and the results of Markov computing. Both parties have to confirm the result and put a signature on the result, which is then published for book-keeping.
Nodes will keep incorporating others until they become “fully-loaded”(meaning when the number of nodes reaches 255 or exceeds the total number of nodes N/128 in the previous round) or until all other nodes in its neighboring area are fully-loaded. It will then be marked “isolated”.
When there are only “fully-loaded” and “isolated” nodes in the network, the next round of absorption starts. “Fully-loaded” will only absorb “isolated” nodes and vice versa.

“Fully-loaded” nodes will become “Super” nodes (meaning they have more than 2^16 nodes in the respective cluster, or more than the total number of nodes N/128 in the previous round), or “Isolated Super Absorbing Nodes (meaning the number of nodes falls below 32768, or there are less than 128 “fully-loaded” nodes left).
When there are more than 255 “Super” nodes or “Isolated Super Absorbing” nodes, a next round of absorption begins and nodes will become core nodes, meaning they have more than 2^24 nodes in the respective cluster, or have more than the total number of nodes N/128 in the previous round. When the number of core nodes exceeds the total number of nodes N/128 in the previous round, the absorption process ends.

The above procedure continues until there are less than 255 clusters. The super nodes of these clusters then become “delegate nodes”.

3.2 Deep learning based code generation for smart contracts

A significant advantage of cogito is that users of smart contracts no longer need to know how to do coding with a programming language. As a matter of fact, cogito only requires a user to type in the purpose of the contract, i.e., input, output, and transaction conditions, as a script. Then a deep neural network based code generator will identify the basic transaction patterns and convert them into a program capturing the behavior of the target smart contract. In the future, we are going to use pure natural language as the input frontend.

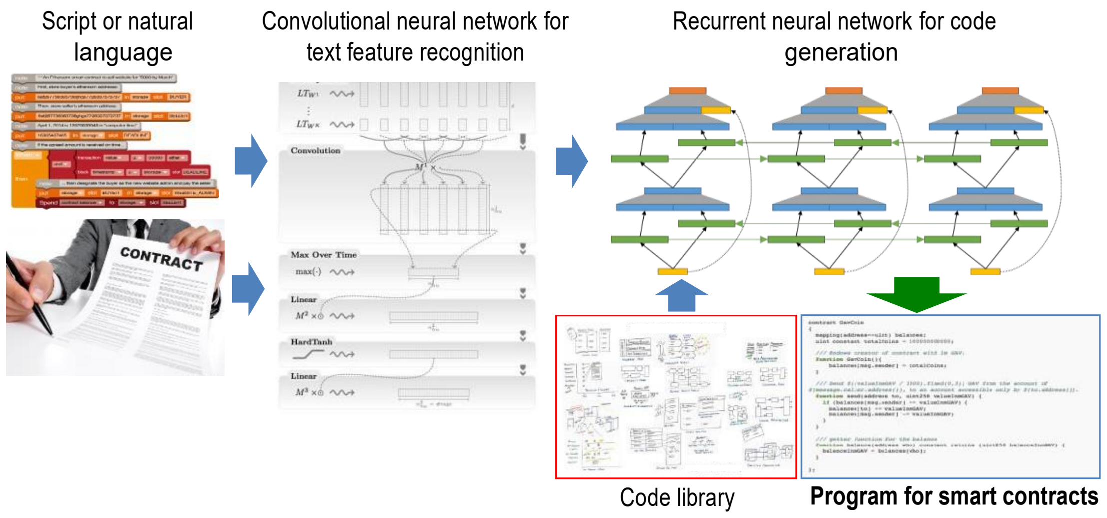

Script or natural language
Convolutional neural network for text feature recognition
Recurrent neural network for code generation
CONTRACT
Convolution
Max Over Time
Linear
HardTanh
Linear
Code library
Program for smart contracts

Figure 2. Deep learning based automatic generation of smart contract programs

Recent development of the deep learning technology proves the feasibility of automatic code generation. First, the maturing of the convolutional neural network (CNN) enables extracting high quality features from language and text. Second, the recurrent neural network (RNN) is mathematically proven to be a Turing complete computer that is able to generate any sequence. These deep neural networks offer the key techniques to analyze the purpose of a contract and generate a respective program.

An illustration of the code generation process for smart contracts is shown in Figure 2. The script capturing the purpose of a contract is fed to a convolutional neural work, which is trained with a large number of labeled samples, to identify the underlying transaction patterns and data attributes. The discovered patterns are organized in a sequence and then given to a recurrent neural network. The RNN has its parameters trained with typical patterns of smart contracts. It transforms the input patterns into a program for the target program with the help of a code library containing codes for various design patterns of smart contracts.

Besides offering dramatically enhanced accessibility, the code generation technology of cogito can be readily integrated with automatic security validation and enhancement techniques. In fact, the whole process can be made as a close-loop for iterative improvement. During the operation of cogito blockchain, the underlying models can be continuously updated for better code generation quality.

3.3 AI-enabled security validation and enhancement

cogito is designed to boost the security of blockchain to an unprecedented level. The overall security framework of cogito consists of four major components, 1) a rule-based semantic and syntactic analysis engine for smart contracts, 2) a formal verification toolkit to prove the security properties of smart contracts, 3) an AI-based detection engine for transaction model identification and security checking, and 4) a deep learning based platform for dynamic security verification and enhancement. Figure 3 depicts the framework.

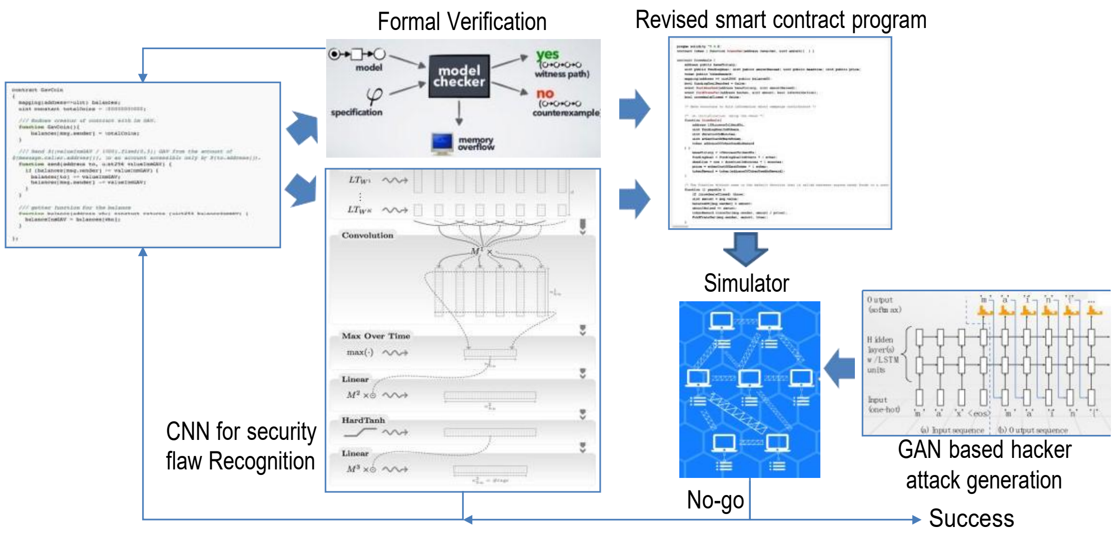


Figure 3. Deep learning based security validation and enhancement of smart contract programs

3.3.1 Syntactic and semantic analysis

Given a program of smart contract, cogito' s built-in compiler constructs a BNF-based AST as an internal representation. For smart contracts that have been compiled into Bytecode documents, cogito first disassembles the binary code and then produces a corresponding BNF. Based on a rule library, which is built with domain knowledge and historical experience, the compiler uses recursive descent parsing to check the AST for any security vulnerabilities.

At the syntactic level, cogito' scompiler identifies the respective finite state machine and data flow graphs from the program. It then performs rule based checking and code revision. Typical examples include: (1) supplementing all conditional clauses to prevent execution problems due to incomplete conditions; (2) Analyzing all public members and functions that are called to determine the level of exposure of contracts; and (3) checking whether transactions steps are complete to make sure that condition descriptions are complete.

At the semantic level, cogito' s compiler provides contextual check to determine operations that do not satisfy rules or are not safe. Typical examples include: (1) checking objects and methods that have to be exposed to external environment to check their necessity and potential flaws; (2) validating whether contract branches or processing of ORACLE are completed and whether there are other abnormal operations when a contract is called; (3) checking the same condition in different options to avoid anomaly as a result of different call sequences.

3.3.2 Formal verification of smart contracts

The above static syntactic and semantic analysis is able to identify logical flaws due to human factors. However, logical problems in runtime cannot be detected. For instance, a user may define contracting conditions when the target contract is under complex constraints. Also, security bugs can arise due to the fact that a contract is executed in a distributed environment and each node sees a unique execution sequence. As a result, the abnormal execution of a contract may leave a loophole for other programs to change its internal state. The cogito blockchain is equipped with a formal verification framework validate the security properties of smart contracts. Based on a functional programming language, the framework integrates a SMT solver and has a multitude of models and tools. It has been used for verification of various software and encryption programs.

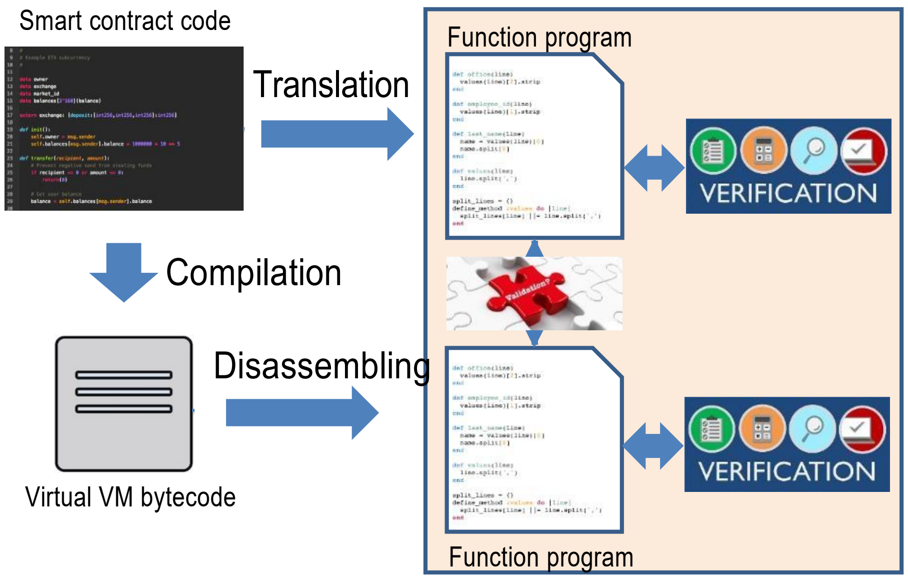


Figure 4. Formal verification of smart contracts

Figure 4 shows the formal verification flow for smart contracts. The verification tool-chain is capable of processing contracts at both source-code and bytecode level. The source code will be translated into an equivalent program in a functional programming language. The adoption of the functional programming model is to expose the hidden logic and ease the succeeding formal operations. The bytecode to be run on the cogito virtual machine is disassembled and transformed into an equivalent functional program. An equivalence checking and other consistency checking can be accomplished on the two functional programs. With the functional programs, a set of property checkers and theorem provers can be applied validate various security properties (e.g., whether the return value of send()

function has been checked).

3.3.3 AI-based verification of smart contracts

The abovementioned formal verification identified loopholes and bugs that can be captured with explicit formal rules. On the other hand, it is challenging to define a complete set of security properties that cover all possible situations. Accordingly, cogito has a deep learning based framework to discover the hidden intention of smart contracts and detect complex patterns of security vulnerabilities.

cogito uses a convolutional neural network to extract textual features and detect interesting patterns. These patterns can be syntax or structure patterns (or a combination of the two). The former usually contains grammar and function features, while the latter structural features. The CNN is trained with Ethereum' s open-source smart contracts, which are manually labeled.

One key characteristic of cogito is its use of AI to automatically identity program syntax to detect typical models and then automatically produce properties that satisfy security requirements. Given a program of smart contract, cogito' s AI engine will automatically detect similarity partial matching and complete matching to predict the behavior model of codes. Based on such models, the AI engine will produce a set of relevant constraints for in- depth formal verification.

3.3.4 Dynamic verification and security optimization based on deep neural network

The security verification techniques covered in the previous three sub-sections are static ones. However, there are loopholes that can only be exposed during dynamic execution.
Table 2 lists the Ethereum smart contracts' vulnerabilities at three levels, namely, high-level programming language, bytecode, and blockchain. It can be seen that most problems in Table 2 only happen during dynamic execution in a distributed environment.

Table 2.Vulnerabilities of Ethereum smart contracts

Level	Vulnerabilities	Attacks targeting vulnerabilities	Responses
Solidity	Call to the unknown	1. The DAO attack	Fast increase of call stack, depletion of balance

	Gasless send	2. “King of the Ether Throne”	Fast consumption of Gas
	Exception disorders	“King of the Ether Throne” GovernMental	Fast increase of call stack, fast consumption of Gas, fast increase of data stack
	Reentrancy	5. The DAO attack	Exhaustion of call stacks, depletion of Gas
	Keeping secrets	6. Multi-player games	Characterization of codes and bytecodes
EVM	Immutable bugs	GovernMental Rubixi	Characterization of codes and bytecodes
	Ether lost in transfer	N.A.	Receiving address is an “orphan” address
	Stack size limit	3. GovernMental	Fast increase of call stack
blockchain	Unpredictable state	GovernMental Dynamic libraries	fast increase of data stack, access deleted or updated modules
	Generating randomness	N.A.	Random distribution results in code execution
	Time constraints	3. GovernMental	Fast increase of call stack, increase of contract balance, and change of timestamp

To address these issues, cogito resort two dynamic approaches, generative adversarial network (GAN) based security verification and distributed concurrence-based dynamic model verification.

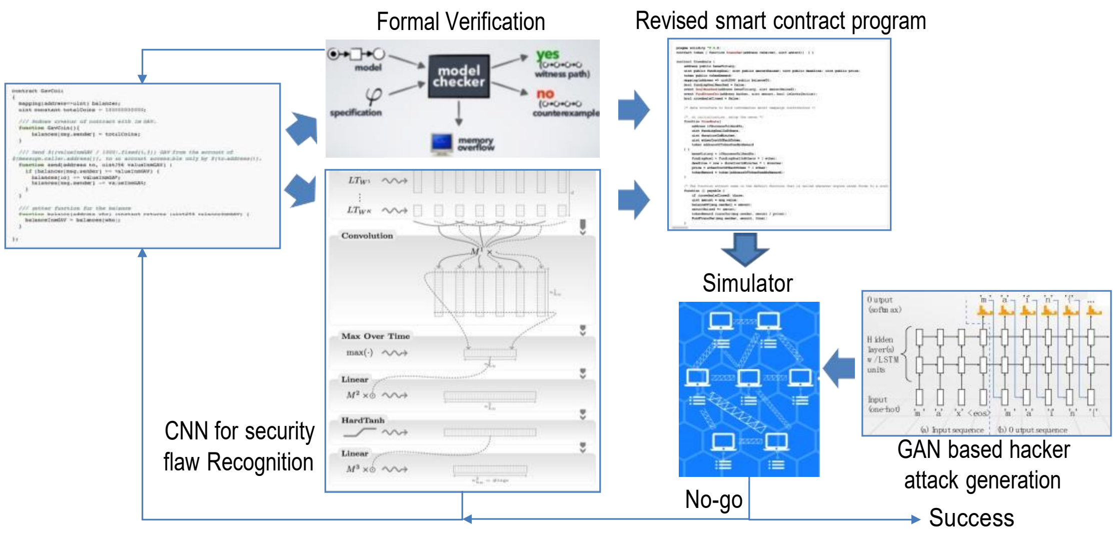


Figure 5. Deep learning based dynamic verification of smart contracts

3.3.4.1 Generative adversarial network based security verification

cogito adopts the recently developed generative adversarial network to accomplish dynamic security verification. As illustrated in Figure 5, the dynamic verification procedure can be coupled with the code generation framework as a closed loop. The GAN framework consists of two RNNs. One RNN is used to revise existing programs for smart contracts, while the other learns to generate hacker programs from random samples from a given probability distribution. After smart contract programs are generated, they will be deployed in the "sandbox simulation network (one that simulates a blockchain and where experiments can be conducted in a controlled manner)", together with the corresponding hacker codes. The cost functions of these two networks are tied together so that the overall optimum is achieved when the whole system reaches a Nash-Equilibrium. At this point, the revised program for smart contracts has the highest level of security.

3.3.4.2 Distributed concurrence-based dynamic model verification

Besides the above general security verification and enhancement techniques, cogito also deploys customized tools for attacks as follows.

Contract sequence attack
This attack takes advantage of the fact that the execution of smart contracts is asynchronous and subject to dynamic change. Even if a contract is statically secure, it is still prone to dynamic attacks, unless contract is designed as dynamically immutable. cogito uses machine learning techniques to defend contracts against such attacks. These techniques include relation

checking of the contract sets to identify relational contract transactions. cogito also provides an asynchronous simulator to help identify anomaly indicators of this type of attack.

Timestamp dependence attack
The root cause of this type of attack is due to excessive discretion of miners. cogito uses AI to dynamically check timestamp dependence or random number dependence to avoid such behaviors.

Mishandling of exceptions and reentrancy attacks
These attacks are in essence caused by anomalies triggered by the function calls of smart contracts. cogito uses a deep neural network to find the coding patterns leading to such vulnerabilities.

3.4 Highly flexible blockchain architecture

cogito is designed to enable an exceptional level of flexibility in blockchain management and operations. Figure 6 illustrates a reference architecture of cogito. As shown in the dotted box, a cogito blockchain consists of six types of networked nodes, standard cogito nodes, cloud access nodes, cloud storage nodes, cogito trustworthy gateway, storage nodes for external data sources, and AI service nodes.

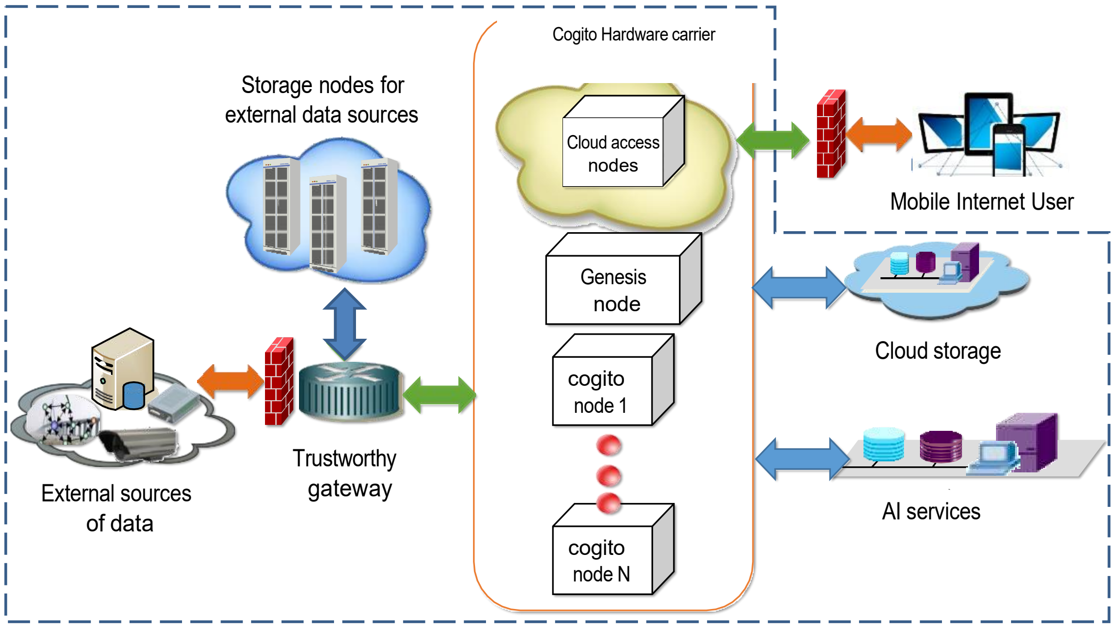


Figure 6. cogito system structure and deployment model

Standard cogito nodes constitute one distributed collaborative control chain and at least one data chain. Each node should have the computing power to run a virtual machine. The distributed collaborative control chain, abbreviated as control chain hereafter, is designed by completely following the distributed and decentralized principle of blockchains.

Cloud access nodes are devised to facilitate the access of mobile devices to cogito’ s cloud service so that all devices connected to cogito have sufficient computing power.

The block data of a cogito blockchain are copied in real time to the cloud storage nodes. Instead of serving for consensus, the cloud storage is designed to make it easier for users to view data offline. Users can obtain the transaction records for validation, data mining, and other purposes.

cogito trustworthy gateway is the key interface to communicate with external data sources. This device will obtain all the external data sources requested by cogito and determine their credibility. When using these data, cogito nodes check the data consistency to determine the level of confidence of each data source and give a credit rating to each external node. cogito trustworthy gateways are also organized with a P2P topology.

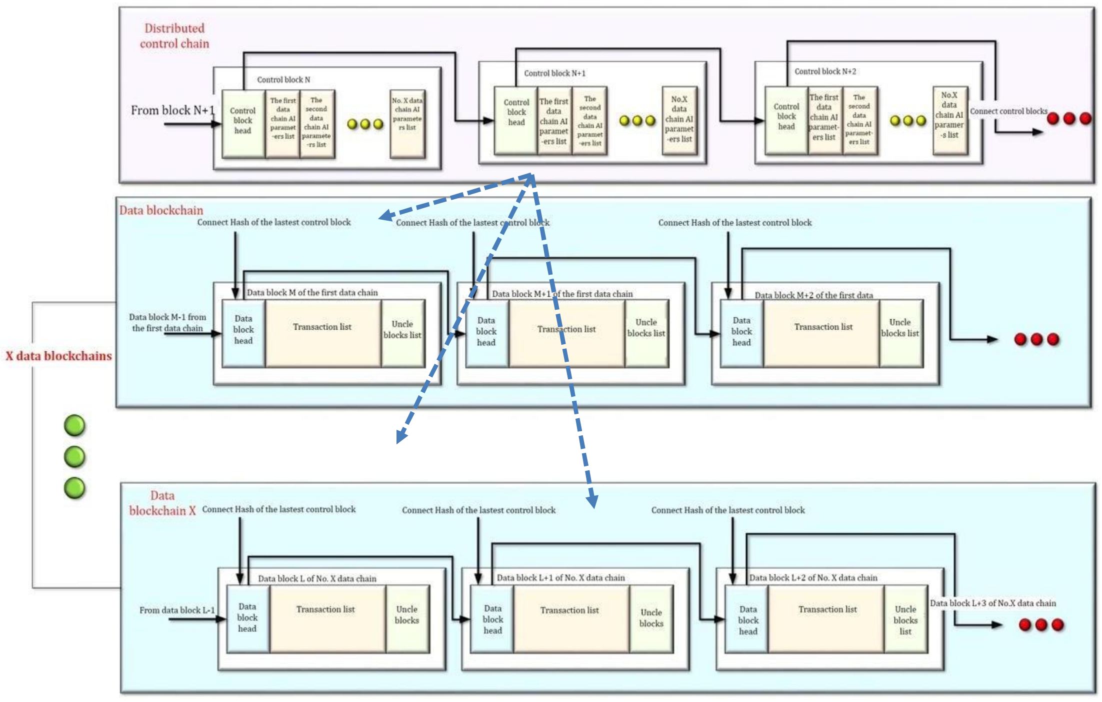
Storage nodes for external data sources from a pool for data obtained via cogito trustworthy gateways. The data have to pass an AI based validation procedure before being checked in. The pool will also store private data of users, but such data are encrypted to ensure that only authorized users have access. All data inside the pool can be verified by checking their signature without accessing their content to prove whether they have been tempered.

AI service nodes serve two functions, supporting the optimization of the entire system and initiating various AI services on cogito. It can service external users by initiating cogito as a service provider and also arrange cogito nodes to offer services to internal users. The AI service nodes depend on affiliated hardware to provide computing power.

3.4.1 Multi-chain structure of cogito

Seeing the strong need to run public and private chains on the same platform, cogito is a blockchain support multi-chain integration. It allows the integration and interoperability of public chains being completely public and private chains being coordinated with a multitude of security access and control mechanisms. Figure 7 depicts a reference multi-chain platform enabled by cogito technologies. It consists of one control chain and multiple data chains.
cogito’ s support to public and private chains is mainly realized through a control chain to enable designated access control and security mechanisms. The control chain consists of control blocks for access control, data storage safety, and multi-user security. The way every

data chain block is decoded and its parameter configuration are determined by the configuration of control blocks. There are also control blocks defining the configuration parameters of the P2P network. Thanks to the control chain, cogito adopts a three-tier security guarantee mechanism composed of block storage key, node security key and user security key. It uses group keys and node keys to encrypt transmission information and supports the distribution, management and security certification of security keys in the form of private chains.
The idea of integrating multiple interoperable chains give cogito significantly greater freedom to support real-world scaled commercial applications. For instance, it is now feasible to have one data chain that supports Bitcoin and one Ethereum data chain. The two chains can exchange data and tokens through cogito’ s secure smart contracts.


Distributed control chain
From block N+1
Control block N
Control block head
The first data chain AI parameters list
The second data chain AI parameters list
No. X data chain AI parameters list
Control block N+1
Control block N+2
Connect control blocks
Data blockchain
Connect Hash of the lastest control block
Data block M-1 from the first data chain
Data block M of the first data chain
Data block head
Transaction list
Uncle blocks list
Data block M+1 of the first data chain
Data block M+2 of the first data
X data blockchains
Data blockchain X
From data block L-1
Data block L of No. X data chain
Uncle blocks
Data block L+1 of No. X data chain
Data block L+2 of No. X data chain
Data block L+3 of No.X data chain
Figure 7. cogito's multi-chain structure

3.3.2 Natural evolution of blockchain parameters

A blockchain typically has a multitude of parameters such as block size, permission algorithm, consensus mechanisms, and mining algorithms. The configuration of these parameters has a significant impact on the overall performance of a blockchain. It is generally impossible to choose an optimal group of parameters for all situations. However, a dramatic change of these parameters incurs great risks for hardfork of cryptocurrencies. The introduction of the multi-chain integration complicates the problem because consensus mechanisms and other parameters of different chains can vary significantly.

A reinforcement learning based parameter optimization engine is embedded into cogito blockchains. A so-called value network continuously learns the long-term rewards of fine- tuning various parameters. After learning, the network can generate best parameter-tuning decision given the current structure and environment of a given blockchain. Figure 8 shows the natural evolution process of cogito. Since the cogito chain learns to optimize itself, he whole process is designated as natural evolution.

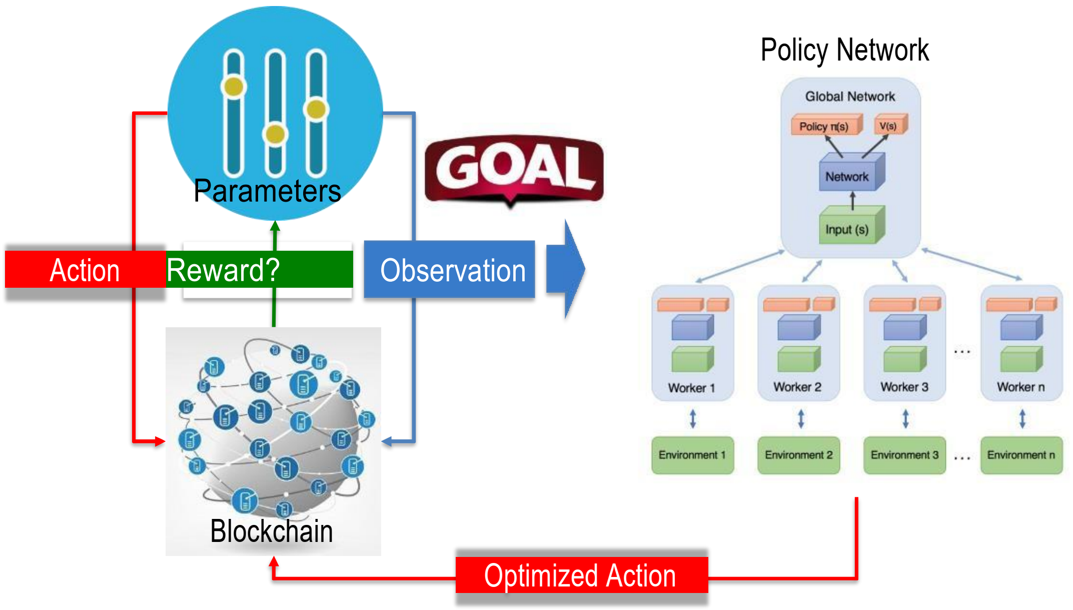
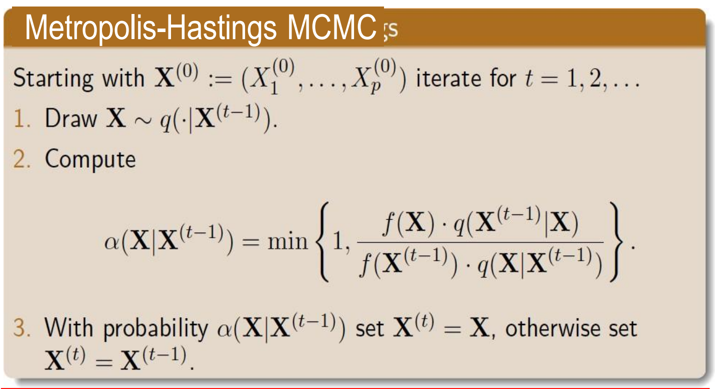


Figure 8. Natural evolution of cogito parameters through reinforcement learning

3.5 Proof-of-work with Markov Chain Monte Carlo computations

Adopting a Hybrid PoS + PoW consensus mechanism, cogito revolutionarily introduces a value-adding computation to replace the traditional Hash computations. The idea is to use the Markov Chain Monte Carlo (MCMC) computation as Proof-of-Work. MCMC has wide applications in such scientific and engineering applications as personalized medicine, finance modeling, human cognition modeling, and social network analysis. As a result, the mining process can be unleashed for real-world applications and generate added value. In fact, we believe the MCMC based mining provides a stronger bridge between the assets in virtual and physical worlds.

MCMC constructs a Markov chain to draw samples from a target probabilistic density. As one of the most influential algorithms invented in the 20th century, it opens the path for enabling Bayesian reasoning with real-world data. MCMC is actually a family of sampling algorithms and the upper half of Figure 9 lists the pseudo-code of a well-known derivative, the Metropolis-Hastings (MH) algorithm, to sample a posterior distribution (i.e., the distribution of a hypothesis given observed data). The MCMC algorithm has a group of essential features that make it appropriate to serve as a mining workload. First, it is compute intensive. The convergence of a complex distribution can take tens of millions of samples. The structural learning problem, which needs to sample a huge number of graph topologies, can take days to finish even on supercomputers. Second, the distribution of samples after convergence is unknown a priori. Such a feature makes it extremely hard to cheat by forging final results. Third, it is possible to evaluate the closeness to final convergence, although the final results are unknown. The lower half of Figure 9 illustrates one trajectory of sampling. The shaded rings in Figure 9 represent the target posterior distribution.



Metropolis-Hastings MCMC

```
Starting with $\mathbf{X}^{(0)} := (X_1^{(0)}, \ldots, X_p^{(0)})$ iterate for $t = 1, 2, \ldots$
1. Draw $\mathbf{X} \sim q(\cdot|\mathbf{X}^{(t-1)})$.
2. Compute
   $$\alpha(\mathbf{X}|\mathbf{X}^{(t-1)}) = \min\left\{1, \frac{f(\mathbf{X}) \cdot q(\mathbf{X}^{(t-1)}|\mathbf{X})}{f(\mathbf{X}^{(t-1)}) \cdot q(\mathbf{X}|\mathbf{X}^{(t-1)})}\right\}.$$
3. With probability $\alpha(\mathbf{X}|\mathbf{X}^{(t-1)})$ set $\mathbf{X}^{(t)} = \mathbf{X}$, otherwise set
   $\mathbf{X}^{(t)} = \mathbf{X}^{(t-1)}$.
```

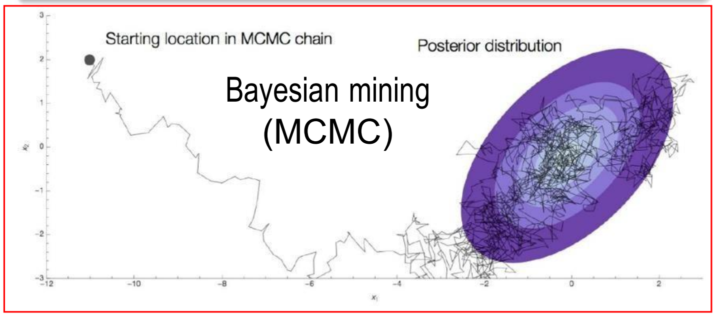


Figure 9. Markov Chain Monte Carlo computations

The MCMC computing pattern is not friendly to either CPU or GPU. In fact, MCMC involves various probabilistic computations such as random number generation and sampling probabilistic distributions. Such computations are not natively supported by current computers and thus incur significant overhead. In our previous work, we developed a Bayesian computing architecture for efficient MCMC computations. The overall hardware architecture is illustrated in Figure 10. It is organized as multiple stochastic multiprocessors to support multi-tasking. One or more stochastic multiprocessors can be allocated to execute a single task of Bayesian reasoning. A stochastic multiprocessor is equipped with a set of 8 sampling units. Note that these sampling units can be invoked in a SIMD manner. A sampling unit is composed of a random number buffer and stochastic logic circuit with the purpose of drawing random samples of a given set common distributions (e.g. uniform and binomial distributions). A commercial IP core is used to generate true random numbers. A programmable controller is designed to coordinate the parallel sampling according to the given MCMC program. The architecture is equipped with an on-chip exemplar memory, which is actually a hybrid computing/memory module to manipulate exemplar operations. Each stochastic multiprocessor also has a scratchpad memory for fast access of general- purpose data. The above architecture serves as a prototype for mining hardware and can be implemented as a dedicated integrated circuit.

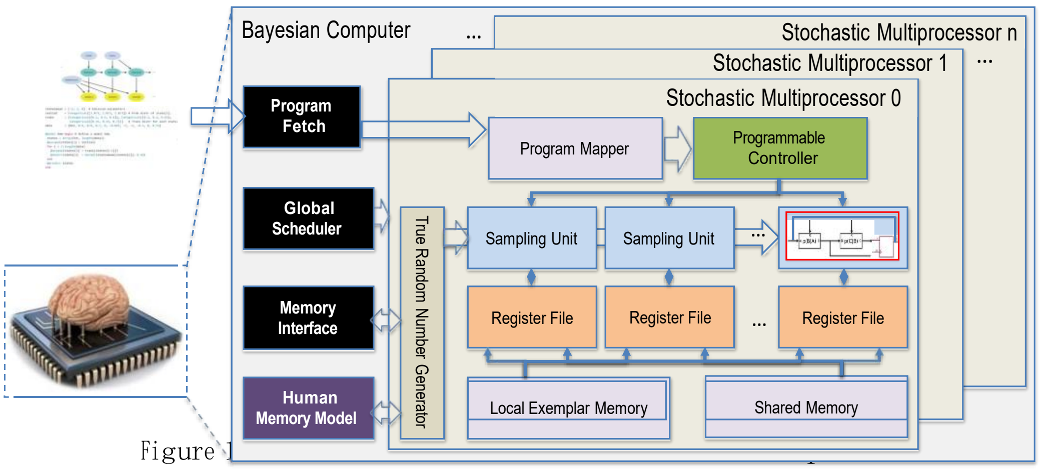


Figure 1

4. Roadmap of cogito

Age of Genesis

2022.5- Initialization
*Infrastructure
*Private chain
*Inter-chain transaction

Age of Speed

2023.12: Light Speed Network
*AI-enabled PoS+PoW consensus
*Random generation of delegates nodes
*Evolutionary parameter optimization

Age of Civilization

2024.1: AI-secured Intelligent Contracts
*Formal verification
*AI based proactive protection
*AI created autonomous constitution

Age of Wonder

2024.4: Mining & More Apps
*Mining ICs
*Computing/mining facility
*Big data applications

As shown in the roadmap above, the development of the cogito blockchain is organized into four stages. In the first stage, coined as the age of Genesis, we are going to build the infrastructure consisting of both a control chain and data chains. The blockchain infrastructure also supports the interoperability between a public chain and multiple private chains. In the second stage, the age of Speed, the focus is on developing a so-called light speed network. With the random hierarchy generation scheme, cogito will allow a transaction speed over 100K transactions per second. The third stage, the age of Civilization, it will witness the rise of AI-enabled protection over blockchains. Besides offering ever stronger security protections, AI potentially allows the autonomous emergence of a constitution, which defines the rules of behaving ethically in the world of cryptocurrency but exhibits itself in a distributed manner. The objective of the final development stage is to build mining ICs, deploy mining facilities, and deliver big data applications. We designate this stage as the age of Wonder because for the first time the computing power of cryptocurrency can be leveraged by real-world data driven applications.

5. Conclusion

With the integration of cutting-edge technologies,We introduce a new dimension, artificial intelligence, into the world of blockchain and cryptocurrency. As the blockchain technology already connects the world as one, cogito’ s AI-enabled intelligent blockchain will give us unparalleled capabilities of understanding the past and predicting the future.