CRYPTOBRICK

TODAY THE FUTURE IS BUILT

Executive Summary

Smart Contracts are HARD. It's hard to fully understand the blockchain concepts they are built on, it's harder to learn writing them right, it's even harder to make them safe and bug free. At the same time, more and more makers and entrepreneurs are designing products and services based on them, and we think the trend is only accelerating, thus the need to lower the entry barriers. We present a cross-chain platform to build well defined, secured, configurable Smart Contracts and make their life cycle easily manageable. Users can visually choose and define features, options and oracles they need by searching and scrolling through a curated catalogue of domain specific Smart Contracts. The platform deals with input validation, wiring and deployment, even across different blockchain technologies. Ancillary operational activities such as name resolution acquisition and usage monitoring are also fully managed. Smart Contracts are designed and developed by our team, paying careful attention to correctness and security. The platform further extends diversity and availability of categories and business domains allowing external developers to share and profit from their own Smart Contracts, all in a safe and validated marketplace environment.

CryptoBrick 2019, Dubai

Simply, Secure Smart Contracts

Published on 24 February 2019

Table of Contents

1. Introduction

Allow everyday business actors to leverage secure, peer-reviewed Smart Contracts on different blockchains, hassle-free. Help contract authors to build secure contracts and monetize their creations.

1.1 Problem Statement

We strongly believe that distributed ledgers and consensus technology (blockchain) hold the same disruptive power the Internet had back in the 90s, even daring to say that they will grow faster and cheaper. With the advent of Ethereum, a whole new concept comes into place: Smart Contracts[1], which enable anyone to encode arbitrary state transition functions that run on a built-in fully fledged Turing-complete machine[2].

The Ethereum Virtual Machine (EVM) executes a stack-based bytecode language; compilers generate bytecode from higher level languages that, while easing the task of programming, still require deep technical and conceptual knowledge of the platform. On top of this, we are witnessing the constant creation of new Ethereum-like platforms, supposed to address more sophisticated problems (NEO, Cardano, Icon, just to name a few). Smart Contracts are a powerful but complex[3] tool and their misuse can have dramatic financial consequences.

Smart Contracts are:

Hard: coding skills are required on top of a profound understanding of security concepts. Making a mistake when drafting a contract can *cost* huge amount of money.

Public and final: correctness / security issues in deployed contracts can lead to *loss of capital*. Moreover, current Smart Contract programming languages have significant quality and security issues that make it harder even for experienced developers not to make any mistakes. See Dao[4] recursive send attack and Solarstorm[5] just to mention two recent cases.

Specialized: customization requires *non-trivial* code changes.

Custom: every blockchain has its own quirks - coders need a deep knowledge of each blockchain *internals* in order to produce meaningful and bug free code.

This leads to Smart Contracts being deployed only as part of vertical service platforms. Self-provisioning of Smart Contracts is infrequent. User base is limited to knowledgeable operators and security issues are frequent. *CryptoBrick* aims to build a secure, user friendly Smart Contracts marketplace in order to waive their complexity and allow an easy deployment on the supported blockchains. Contracts are generalized by means of **templates**, supporting parameters to custom fit the user needs; furthermore, a wide range of oracles can be consulted and included in the contract to allow custom actions based on the results of real world events.

1.2 Value Proposition

Despite their potential, the current complexity of Smart Contracts holds back a lot of potential users and therefore a good share of the market. It is our belief that this technology can and should achieve mass adoption on global scale.

CryptoBrick aims to unleash this potential by allowing everyday business actors to leverage secure, peer-reviewed Smart Contracts on different blockchains, hassle-free:

- Choose from a curated, secured, peer-reviewed portfolio of Smart Contracts.
- Configure each contract instance in a clear, user-friendly interface - no other tools required.
- Custom fit the behavior of the contract by setting the parameters exposed by the template.
- Employ oracles to react on world events directly from the contract.
- Deploy the contract on the desired blockchain.
- Integrate and monitor the contract within end user applications through API and tools.
- Accept payments in multiple crypto-currencies, independently from the chosen blockchain.

An ever increasing *CryptoBrick* library of pre-configured and generalized Smart Contracts will be made available for the user to choose from, on top of a user driven marketplace that will provide even more variety of contracts. Users can check contract reviews weighted by our rating protocol

and stored on the blockchain to guarantee their integrity, feedback from other users, and contract documentation.

CryptoBrick plans to help not only authors to build secure and robust contracts by providing a specific platform for development and testing, but also allowing them to monetize their creations. This is made possible using facilities such as:

- External integrations via oracles.
- Testing API.
- A simple, secure, cross-blockchain development environment.

Additionally, an ecosystem of professional services is available from day one:

- Architecture definition and security reviews.
- Porting Services among blockchains.
- Integration services.
- Maintenance and operation of Smart Contracts.
- A marketplace for certified partners to further monetize their creation.

The platform allows easy deployment on either Test and Main net. API access allows an easy integration of Smart Contract state and data into external applications.

1.2.1 Testing and Learning

CriptoBrick wants to encourage the utilization of Smart Contracts, thus, any deployment of Smart Contracts on testnet incurs no cost. In this way *CryptoBrick* will set the foundation to be the go-to playground for testing and become itself the main learning platform for Smart Contracts.

1.3 Blockchain Evolution

Blockchain stack has evolved at an astonishing speed since its foundation, the main reason being the improvement of flexibility and broadening of its usage for different purposes other than value transfer. Another driver of the evolution has been security: breaches of the caliber of DAO and Parity[6] may have had disastrous consequences and catastrophic results and pushed the development even further.

In the beginning was *Bitcoin*, the 1^{st} generation blockchain; its commitment has always been the transfer and storing of value and, being a novel technology, it only supports a minimum set of information for the purpose: from and to whom, when, and how much.

The need of additional information to be included in transactions led to the creation of 2^{nd} generation blockchains. This development was initially started by *Ethereum* followed in recent time by other projects, like Antshare (NEO). Ethereum adds metadata support; metadata can store information such as the terms and conditions of the transfer, why was the money transferred, and what were the involved entities. In the Ethereum ecosystem, the connection between the transferred value and its metadata is called Smart Contract; unfortunately, storing metadata on the blockchain may prove inefficient in the long run as transaction costs go higher and higher with the use.

Currently, 3^{rd} generation blockchains, such as *Cardano*[7] decouple the value transfer from metadata using different layers. The value layer is responsible for the token economics and the balances of all user accounts, the control layer manages Smart Contracts as well as regulation and digital identity functions.

CryptoBrick aims to interface with all these blockchains to provide a seamless experience for contracts deployment regardless of the language "spoken" and their limitations and quirks.

2. Roadmap

Rome was not built in a day — Although the roadmap is ambitious, at *CryptoBrick* we believe it is a feasible one. To support our thesis, we have developed a proof of concept[8] available for testing to anyone who wishes to try it. Milestones mentioned below refer only to technical achievements.

2.1 Proof of Concept - PoC

At *CryptoBrick* we have already developed and publicly deployed a proof of concept[8] focused on allowing the user to customize and deploy an example escrow contract on Ethereum Rinkeby testnet. The PoC also includes a management page and a basic Metamask[9] wallet integration easing interactions with the contract by the involved parties.

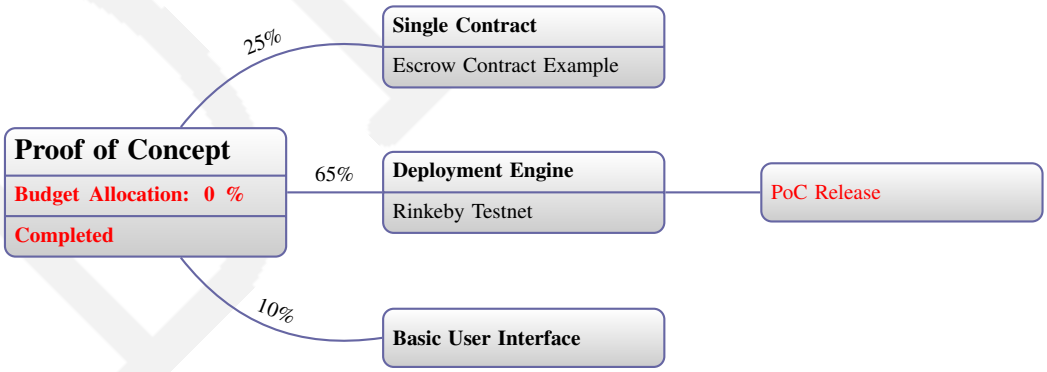

Figure 2.1: Proof of Concept Budget Allocation

2.2 Milestone 1

The first Milestone includes full integration with external wallets to facilitate the use of **BRIK** tokens. It also covers the foundation of *CryptoBrick* template library as well as the Deployment Engine on Ethereum. Utilities such as single sign on[10], two factors authentication[11] and analytics are included in this release.

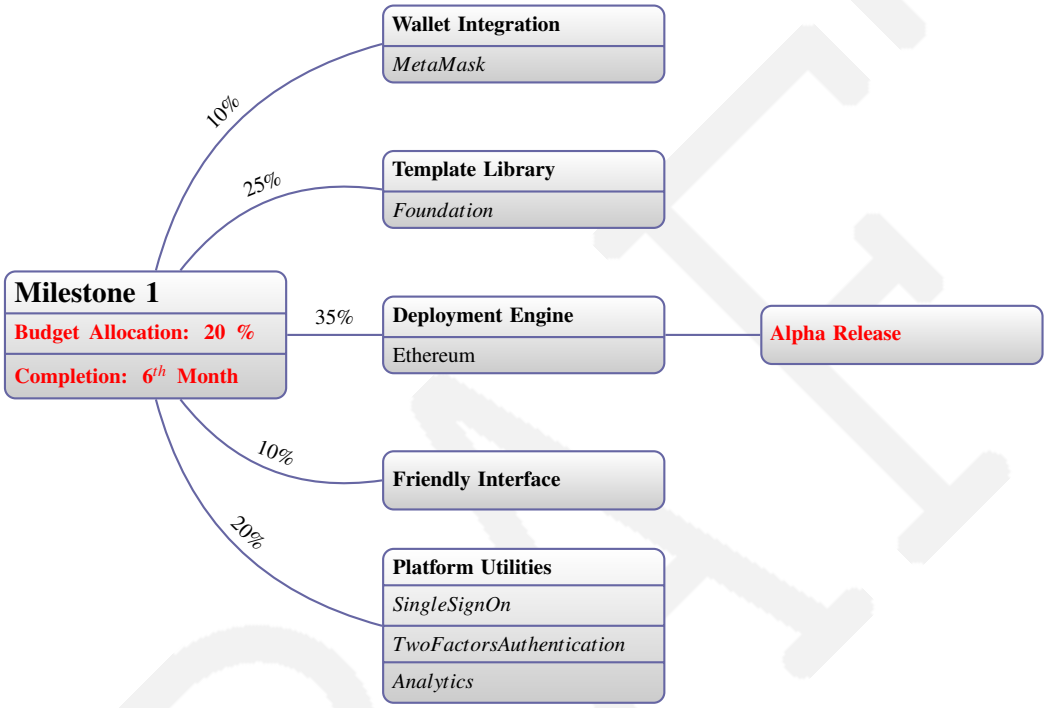

Figure 2.2: Milestone 1 Budget allocation

2.2.1 Template Library - Foundation

CryptoBrick supports an ever-growing Smart Contract templates library to choose from, configure and deploy on the preferred blockchain. The library contains templates covering many different domains and provides filtering and searching capabilities to easily browse them. Each template can have one or more implementation language to be deployable on different blockchains, available in subsequent milestones. Examples of available templates domains are:

- Escrow.
- Derivatives.
- Notarization.
- Auctions.

Escrow Service

A powerful use case for Smart Contracts is the trustless decentralized escrow where the contract act as a third party to enable value transfer between parties. *CryptoBrick* voids all the unnecessary complications of coding and managing the Smart Contract; the user defines the customizable parameters while the platform notifies to the other involved parties about the deployment, giving them access to a friendly user interface to monitor and manage operations on it.

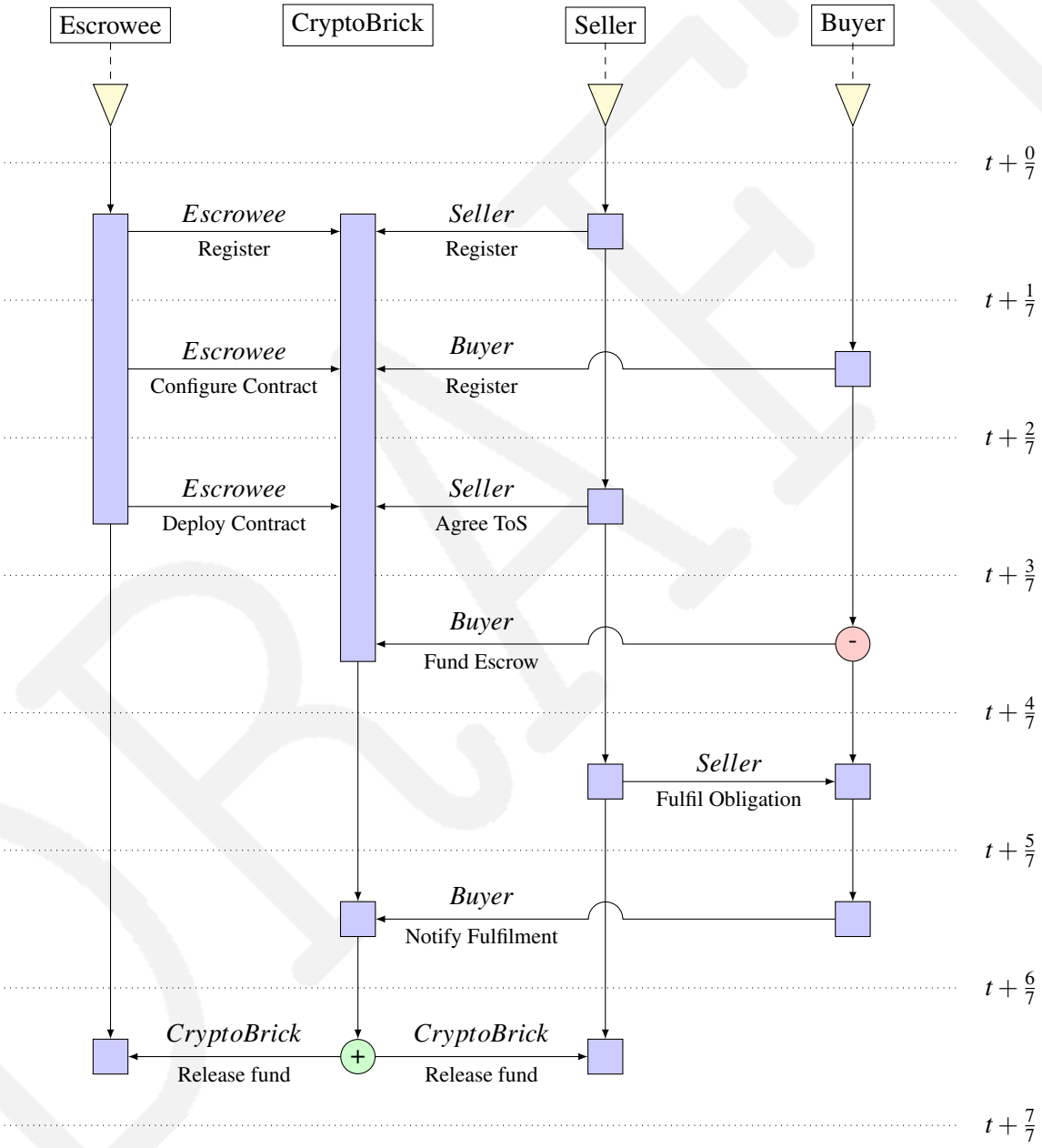

Figure 2.3: Escrow Contract Sequence Diagram

The Finite State Machine[12] that describes the behavior of the escrow contract is illustrated below. Every contract in *CryptoBrick* comes with a live representation of its finite state machine in order to simplify user understanding and awareness about what exactly is going on.

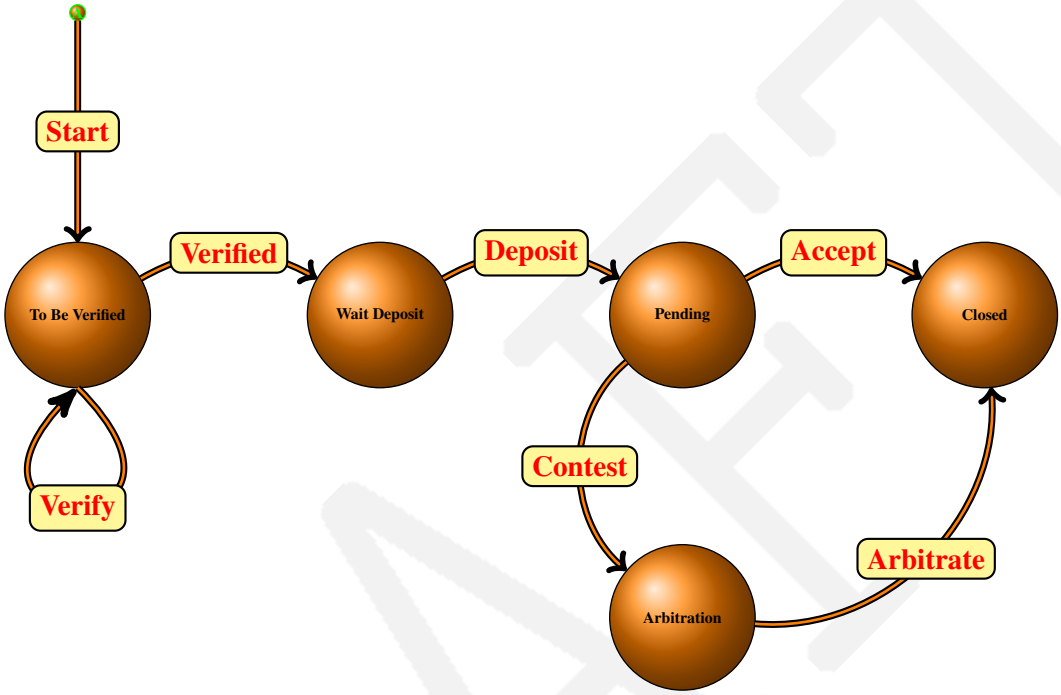

Figure 2.4: FSM Escrow Contract

2.2.2 Two Factors Authentication

CryptoBrick is focused on delivery of secure applications; before performing an executive transaction, a second layer of authentication is needed. Leveraging standard two factor authentication, a second device (mobile, tablet, etc.) produces a secret key, only valid within a configurable time frame, used to complement the password and execute the transaction. The reliability of the second factor is equal to the reliability of the used device.

2.3 Milestone 2

During milestone number two *CryptoBrick* adds the integration of oracles queryable from a Smart Contract and introduces atomic swap to allow seamlessly trustless cross-chain atomic trading. On top of expanding the template library of Smart Contracts, *CryptoBrick* also deploys its own wallet while introducing the concept of the marketplace.

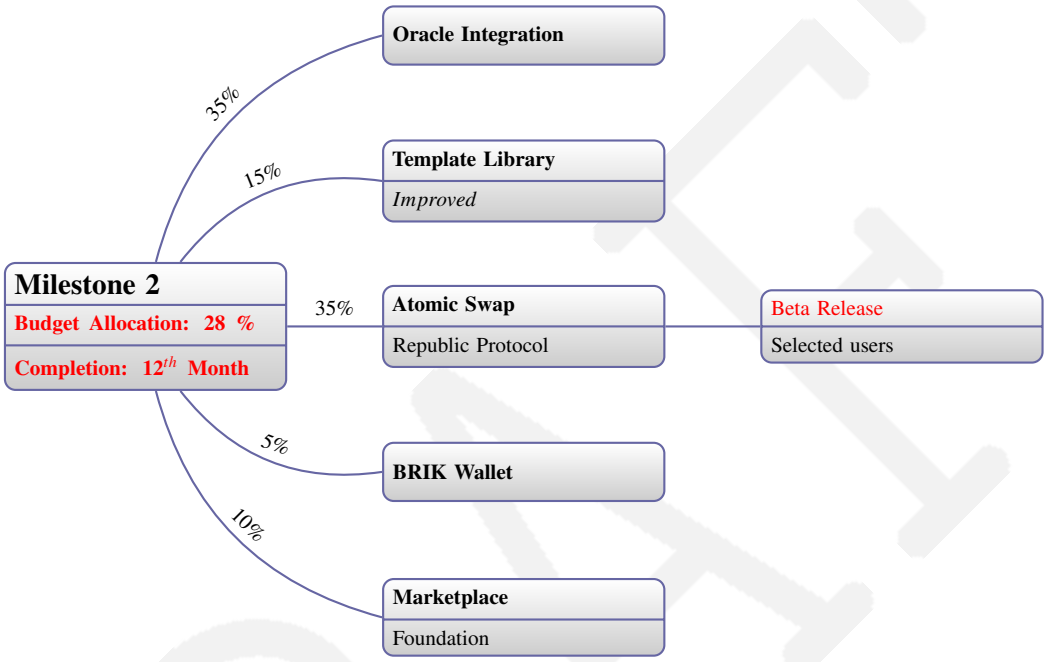

Figure 2.5: Milestone 2 Budget Allocation

2.3.1 Oracle integration

A Smart Contract without the ability to get input from external sources is severely limited in its functionality, fully depending upon actions from other actors to do anything. Sometimes, to be useful, a contract needs the ability to refer to values provided by the external environment, such as the result of a sport event or the price of a physical asset. Being the blockchain trustless by definition, this cannot be achieved by the contract alone. Oracles come in handy for this task. However, a decentralized contract that requires trusting a single outside data source is a bit of a contradiction. To decide whether a provided fact is true or not, oracles essentially require the implementation of a N-of-M consensus mechanism. There are already several projects that attempt to bring external data into the blockchain in a decentralized way, Oracolize[13] being one of these. The oracle is deployed as a N-of-M Smart Contract on Ethereum whose M parties are service providers from within the platform. The oracle commits to answer a Yes/No question, within a limited timeframe, by weighting replies and send back the answer to the Smart Contract once available. The user who wishes to query the oracle from his Smart Contract needs to stake **BRIK** token in order to do so. These tokens are then shared between all the N parties who reply, within the time limit, to the question. On *CryptoBrick* oracle judgments may also be further questioned through the arbitration protocol.

2.3.2 Atomic Swap

Cross-chain atomic swap allows for trustless exchange of **BRIK** for supported crypto-currencies[14]. It can be implemented using a hashlock, that locks the transactions on both blockchains under the same value. *CryptoBrick* will implement atomic swap support using the best standards available at the beginning of the milestone.

2.3.3 BRIK Wallet

CryptoBrick allows users to both use platform-managed wallets and interface with client own wallets via web3js and the Ethereum Generic JSON RPC API. The same approach is extended to other supported blockchains.

2.3.4 Marketplace

Nowadays, marketplaces increasingly extract value from their users as their network grows. For example, skilled developers on Upwork marketplace earn much less than they did in the past, yet the cost to hire them has not decreased. If the demand for skilled programmers increases, the developer should benefit when prices reach their equilibrium, however established marketplace platforms exploit these shifts to capitalize on the value added by their community. As such, the value is mostly extracted by shareholders which are extremely rewarded when shares pick up value. Early adopters of the platform, although essential at the beginning, receives only decreasing revenue compared to their initial added value. On the other hand, **BRIK** tokens create a huge benefit for early adopters and token holders, thereby aligning user incentives toward further growth of the platform. CryptoBrick gives every certified partner the opportunity to earn tokens by publishing increasingly better and more efficient Smart Contracts on the platform, charging 0% commission fees for the transaction. We charge only a minimal fee for the gas used in addition to the nominal transaction cost. Because the **BRIK** token is used for all transactions, there are no external transaction fees or currency conversion fees. Meticulous security tests are performed before a third party is accepted as a certified party. Furthermore, every contract deployed undergoes a manual security code audit performed by our experts team before being released to the public. Users can always request independent security and functional audits of Smart Contracts by leveraging other third parties available in the marketplace.

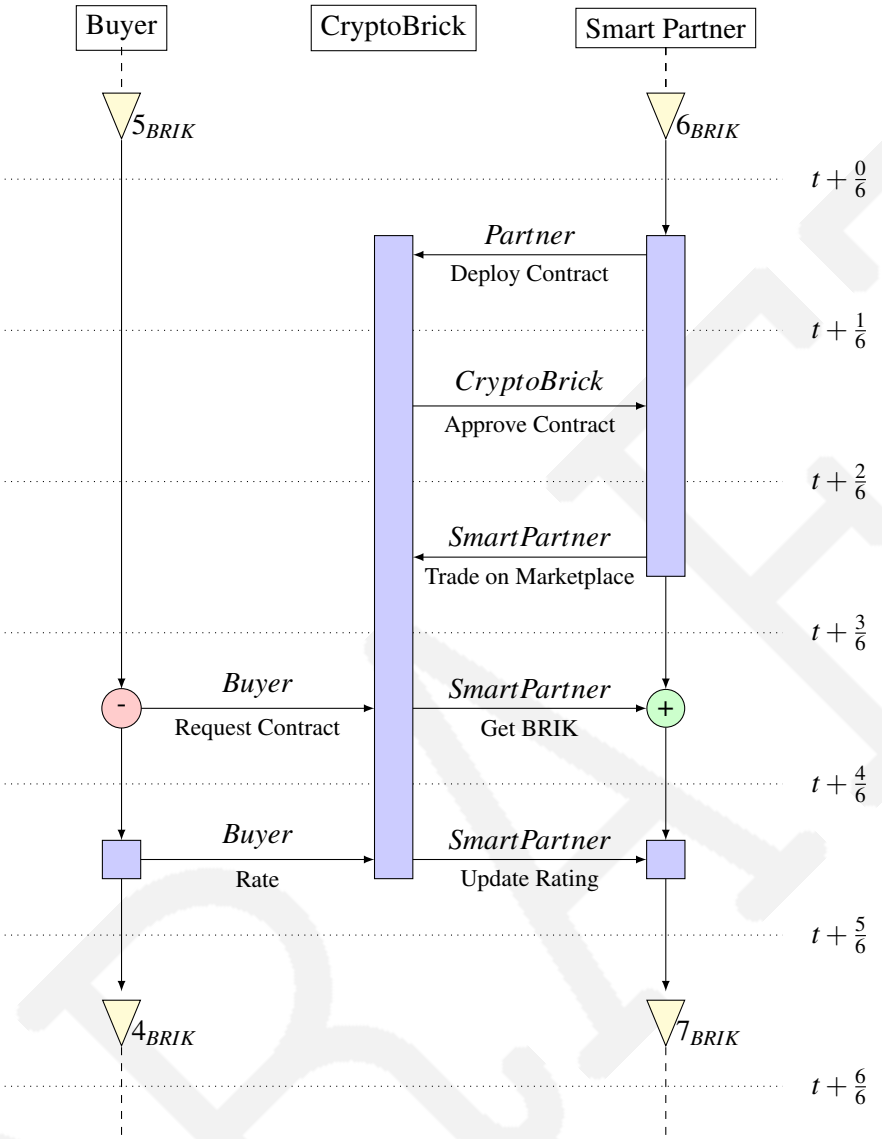

Figure 2.6: Marketplace Sequence Diagram

Smart Contract Best Practices

It is of the utmost importance to guarantee a fair, honest and immutable review of the contracts listed in the marketplace. *CryptoBrick* encourages users to review the contracts they choose and deploy. Reviews are stored on the blockchain through the Rating protocol and linked to the user identity. To ensure privacy and keep gas costs at a minimum, reviews are stored off-chain with only an hash of the review being posted on the blockchain. This ensures immutability and prevents post-hoc modification of any review. Furthermore, each Smart Contract made by our Certified Partners is thoroughly assessed before being put into the marketplace. The assessment takes into consideration best practices of the development language, known attacks, and the engineering of the contract.

Only a limited number of patterns would generally be approved before releasing a Smart Contract, paying utmost attention to external calls made. Dynamic analysis on Testnet on top of manual and automatic assessment with static analysis tools are performed. In order to improve code quality, *CryptoBrick* enforces rules for style and composition, making code easier to read and review. Code will need to be resubmitted if errors are discovered or if improvements need to be made. *CryptoBrick* and certified parties have the ability to trigger the circuit breaker to stop the execution of a Smart Contract if errors are discovered. Furthermore, Smart Contracts are subject to review from users who possess a rating score of minimum 3. The rating protocol weight the review using the score level of the user.

2.4 Milestone 3

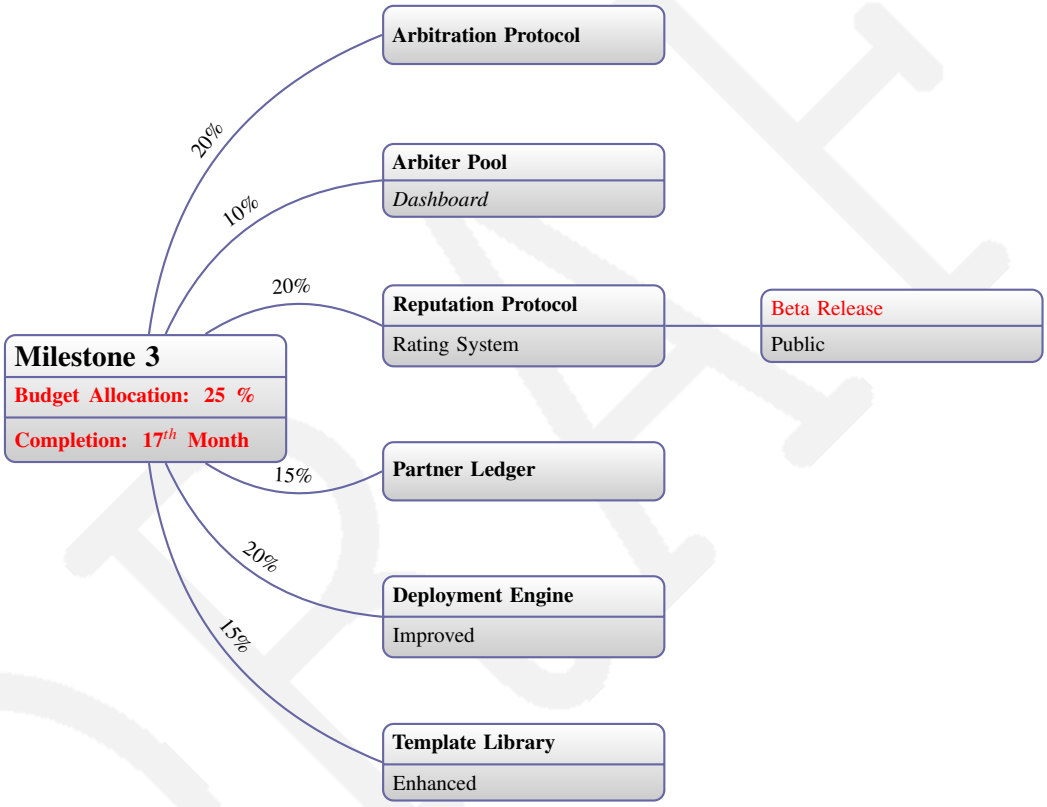

Figure 2.7: Milestone 3 Budget Allocation

2.4.1 Arbitration Protocol

In some usage scenario, a third party (arbiter) may be needed to arbitrate disputes that arise. This protocol defines the rules for the arbitration and rating score changes. Arbitration begins when a user sends a request and stakes **BRIK** tokens to the protocol. The accused party has to stake the same amount. Arbiters are selected from a pool of the Partner Ledger users; they need to stake **BRIK** tokens as well, which are feed into an algorithm responsible to select the moderators. The algorithm takes into account a few factors such as the current reputation and the stake amount.

The Arbiter gets a reward in **BRIK** token for accurate decision while he gets penalized for failing to render a decision within the time limit. He does not get penalized for incorrect decisions, however, his rating is lowered making him less appealable for the algorithm the next time a dispute arise. Once a decision is made, the contract gives an aggregate of the majority voters' reward offering to the winning party, and the rest goes to the losing party. All parties are paid once the dispute is resolved and arbiters become eligible to judge the next dispute.

2.4.2 Reputation Protocol

The reputation protocol computes and maintains scores of platform users on *CryptoBrick* partner distributed ledger. Triggers fire automatically when the score of a user needs to be updated, such as when he goes through the KYC process or when he leaves a review. Due to the way distributed applications are built, the score is backed by transparent algorithms defined in Smart Contracts and accessible to everyone; any existing distributed application can fetch, through our REST API, the reputation of any user.

Rating System

The User score is handled by the reputation protocol which updates and gets values whenever it is needed, e.g. in case it needs to check if the user is allowed to perform a specific operation. Score goes from 0 (lowest) to 10 (highest).

Certified Partner

Users are eligible to become certified partners if they possess at least a rating score of 7. These users, should they decide, undergo a specific test before reaching the certified status. The test is designed to assess the technical knowledge of the certified partner with a specific focus on application security. Certified partners may participate in arbiter and oracle pools.

2.4.3 Partner Distributed Ledger

Rather than an open permission-less system such as Ethereum, *CryptoBrick* hosts an Hyperledger Fabric[15] blockchain. The main reason for this choice is to avoid unknown identities to participate in our certified partner network and at the same time authenticate them through a Membership Service Provider (MSP). The MSP acts as a Certification Authority to issue certificates and authenticate partners. *CryptoBrick* distributed ledger is used to authenticate partners and manage their permissions on the platform. Arbitration and Reputation protocol interact with the ledger both to choose arbiters from the pool and to store the rating of users and contracts. Access control list is enforced to provide additional layers of permission through authorization of specific operations while correlating identities weighted by the rating protocol. For example, a specific user could be permitted to invoke a chaincode application, but blocked from deploying a different one. Chaincode is essentially what a Smart Contract is on Ethereum; they encode logic that is invoked by specific types of transactions on the channel. Fabric does not require a built-in cryptocurrency as consensus is not reached via mining; the consensus method is based on a Proof of Stake algorithm called Byzantine Fault Tolerance[16]. Certified Partners are part of the consensus mechanism of Fabric.

2.5 Milestone 4

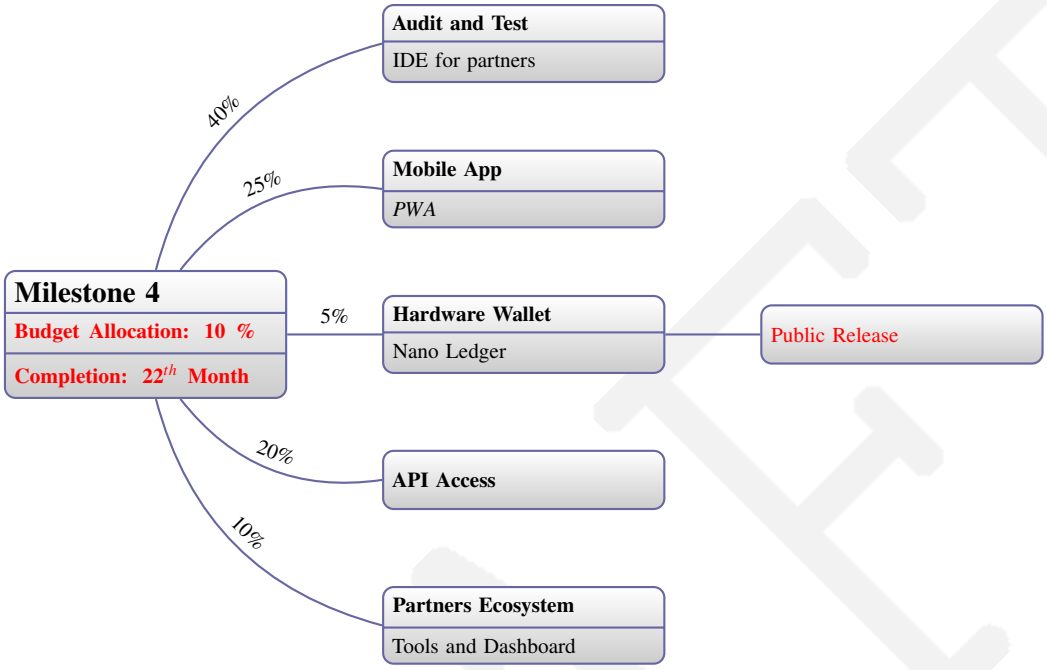

Figure 2.8: Milestone 4 Budget Allocation

2.5.1 Mobile Application

At CryptoBrick we have pondered the idea of building a mobile App using a state of the art stack. Progressive Web Application (PWA) fixes the gap between mobile application and web application through the use of Web API. Traditionally browser mobile web applications could not perform operations such as sending push notification or allowing to work offline, however PWA brings the features expected from native apps to the mobile browser experience in a way that uses standard-based technologies and run in a secure container accessible to anyone on the web. The core of PWA functionalities relies on service workers who power offline functionality, push notifications, background content updating, content caching, and more.

2.5.2 REST API

The platform supports REST JSON APIs allowing enterprise customers and users to access all functionalities programmatically, such as browsing the contract library, deploying and managing contracts and accounts.

2.6 Milestone 5

Figure 2.9: Milestone 5 Budget Allocation

2.6.1 Loyalty

In order to incentivize the utilization of the platform, CryptoBrick has reserved 2% of the total supply of **BRIK** tokens to be redistributed through a loyalty program. Users are eligible for a cash back of 0.2% of the transaction cost in **BRIK** for any additional service they will use after the eligibility condition is met. The program runs until the overall Loyalty Program token pool will be available and will be based on a "first in first served" policy.

2.6.2 Advanced Testing IDE

Contract development is currently made harder by the lack of a cross-chain integrated development environment. *CryptoBrick* will develop tools and plugins to enhance IDEs to help creating a complete ecosystem for contract development, review and monetization that will allow the platform to grow.

2.7 Technicalities

CryptoBrick is based on the most established and proven technologies. Integration with the Ethereum and other blockchains is done using official clients and standard blockchain protocols, such as the JSON RPC API, in order to simplify updates and ensure maximum compatibility. Key technologies are be Java, Spring Framework, Linux as operating system and Docker for containerization. Architecturally, simplicity will be the driving factor in every decisions, to guarantee a clear, maintainable, secure and scalable solution.

THIS IS NOT A SOLICITATION FOR INVESTMENT

This document is not a solicitation for investment and does not pertain in any way to an offering of securities in any jurisdiction. This document provides an overview of the functionality and a technical description of the Platform and the issuance of **BRIK** tokens.

DO NOT PURCHASE **BRIK** TOKENS IF YOU ARE NOT AN EXPERT IN DEALING WITH CRYPTOGRAPHIC TOKENS AND BLOCKCHAIN-BASED SOFTWARE SYSTEMS.

Before releasing the token to the user, *CryptoBrick* will perform a Know Your Customer process to prevent fraud and money laundry. US citizens who wish to proceed into purchasing **BRIK** token must be accredited investor[X] by SEC standard.

CryptoBrick 2019, Dubai

Simply, Secure Smart Contracts

Published on 24 February 2019

Bibliographie

[1] Even without any extensions, the Bitcoin protocol actually does facilitate a weak version of a concept of "Smart Contracts" expressed in a simple stack-based programming language. https://bitcoin.org/bitcoin.pdf

[2] https://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/turing-machine/one.html

[3] The part of the protocol that actually handles internal state and computation is the *Ethereum Virtual Machine* (EVM), a stack based machine which is therefore responsible to run Smart Contract code. Assuming someone wants to "simply" add the number 1 and 3, he needs to feed the EVM the following code: PUSH1 0x01 PUSH1 0x03 ADD STOP. A POP from the stack would deliver the desired result...Needless to say, more complex statements requires additional interaction with the EVM.

[4] http://vessenes.com/more-ethereum-attacks-race-to-empty-is-the-real-deal/

[5] https://pdaian.com/blog/chasing-the-dao-attackers-wake/

[6] https://www.coindesk.com/brain-freeze-parity-bug-continues-no-easy-solution-sight/

[7] https://cardanofoundation.org

[8] https://poc.cryptobrick.io

[9] https://metamask.io

[10] https://en.wikipedia.org/wiki/Single_sign-on

[11] https://en.wikipedia.org/wiki/Multi-factor_authentication

[12] https://en.wikipedia.org/wiki/Finite-state_machine

[13] http://www.oraclize.it/

[14] https://republicprotocol.com/

[15] https://hyperledger-fabric.readthedocs.io/en/release-1.1/

[16] http://www.comp.nus.edu.sg/~rahul/allfiles/cs6234-16-pbft.pdf

[17] https://github.com/ethereum/wiki/wiki/ICAP:-Inter-exchange-Client-Address-Protocol

[18] https://www.investor.gov/additional-resources/news-alerts/alerts-bulletins/investor-bulletin-accredited-investors#.VNI7IdLF9Qg

List of Figures