CHAPTER 6
SMART CONTRACTS

From : Mastering Blockchain

By – Imran Bashir

www.packet.com

History

Smart contracts were first theorized by *Nick Szabo* in the late 1990s, but it was almost 20

years before the true potential and benefits of them were truly appreciated.

Smart contracts as described by *Szabo*

- *A smart contract is a*
  - *computerized transaction protocol that executes the terms of a contract.*
- *The general objectives are*
  - *to satisfy common contractual conditions (such as payment terms, liens, confidentiality, and even enforcement),*
  - *minimize exceptions both malicious and accidental, and*
  - *minimize the need for trusted intermediaries.*
- *Related economic goals include*
  - *lowering fraud loss, arbitrations and enforcement costs, and other transaction costs.*

In Bitcoin

- This idea of smart contracts was implemented in a limited fashion in bitcoin in 2009,
- Where bitcoin transactions can be used to transfer the value between users, over a peer-to-peer network
- Where users do not necessarily trust each other and
- There is no need for a trusted intermediary.

Definition

- There is no consensus on a standard definition of smart contracts. It is essential to define what a smart contract is, and the following is the author's attempt to provide a generalized definition of a smart contract.
- ***A smart contract is a secure and unstoppable computer program representing an agreement that is automatically executable and enforceable.***

Working

- Smart contracts usually operate by managing their internal state using a state machine model.
- This allows development of an effective framework for programming smart contracts, where the state of a contract is advanced further based on some predefined criteria and conditions.
- Smart contracts can be fully or partially automated.

Properties

1. Automatically executable } Minimum required
2. Enforceable
3. Semantically sound
4. Secure and unstoppable.

- Deterministic(by nature and the language) - This property will allow a smart contract to be run by any node on a network and achieve the same result.

Ricardian contracts

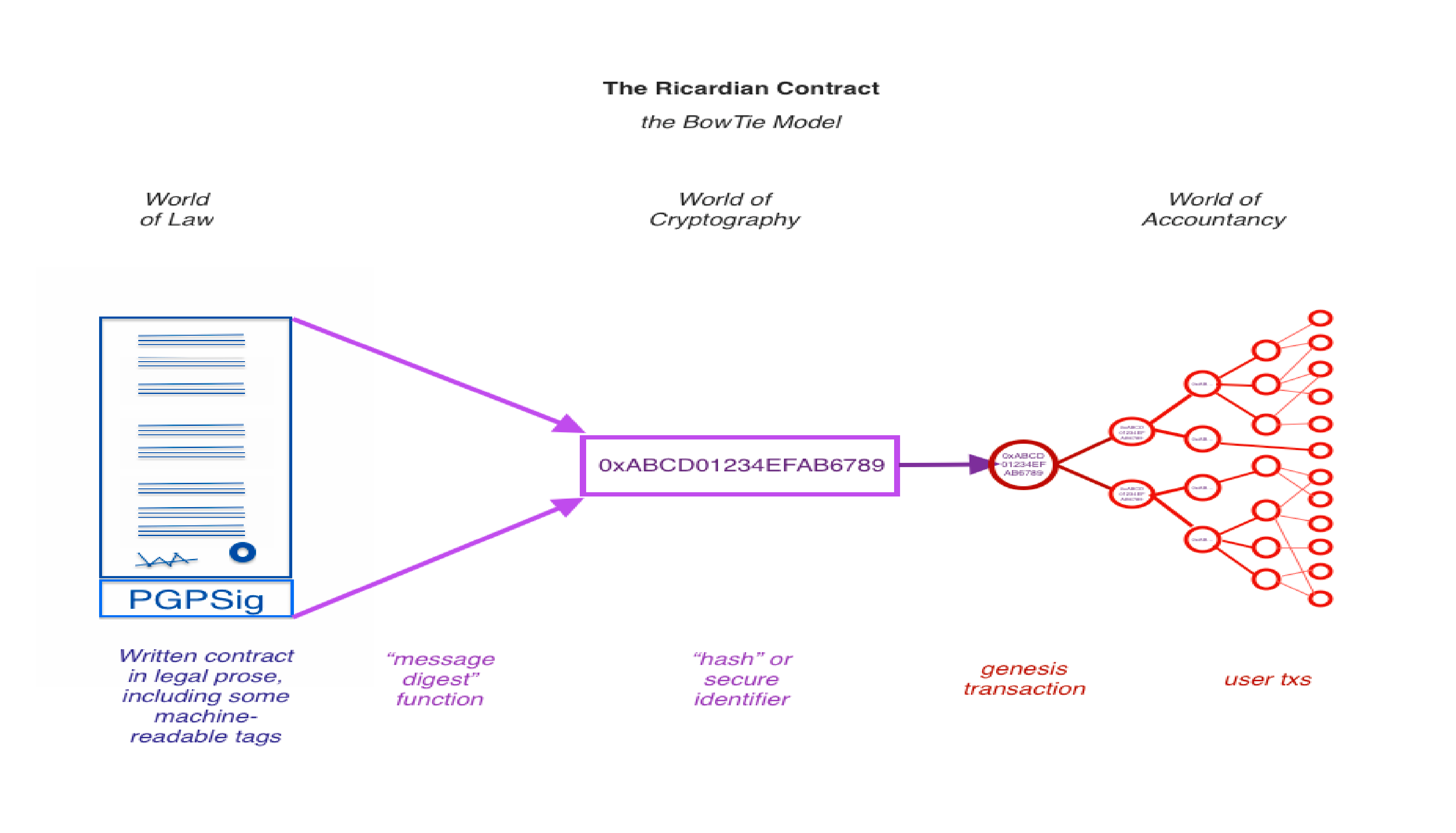
- Ricardian contracts were originally proposed in the *Financial Cryptography in 7 Layers* paper by *Ian Grigg* in late 1990s.
- These contracts were used initially in a bond trading and payment system called **Ricardo.**
- The key idea is to write a document which is *understandable and acceptable by both a court of law and computer software.*
- It identifies the issuer and captures all the terms and clauses of the contract in a document in order to make it acceptable as a legally binding contract.

Recardian Properties

http://iang.org/papers/ricardian_contract.html, a Ricardian contract is a document that has several of the following properties:

1. A contract offered by an issuer to holders
2. A valuable right held by holders, and managed by the issuer
3. Easily readable by people (like a contract on paper)
4. Readable by programs (parseable, like a database)
5. Digitally signed
6. Carries the keys and server information
7. Allied with a unique and secure identifier

Ricardian contracts, bowtie diagram



The Ricardian Contract

the BowTie Model

World of Law

World of Cryptography

World of Accountancy

Written contract in legal prose, including some machine-readable tags

"message digest" function

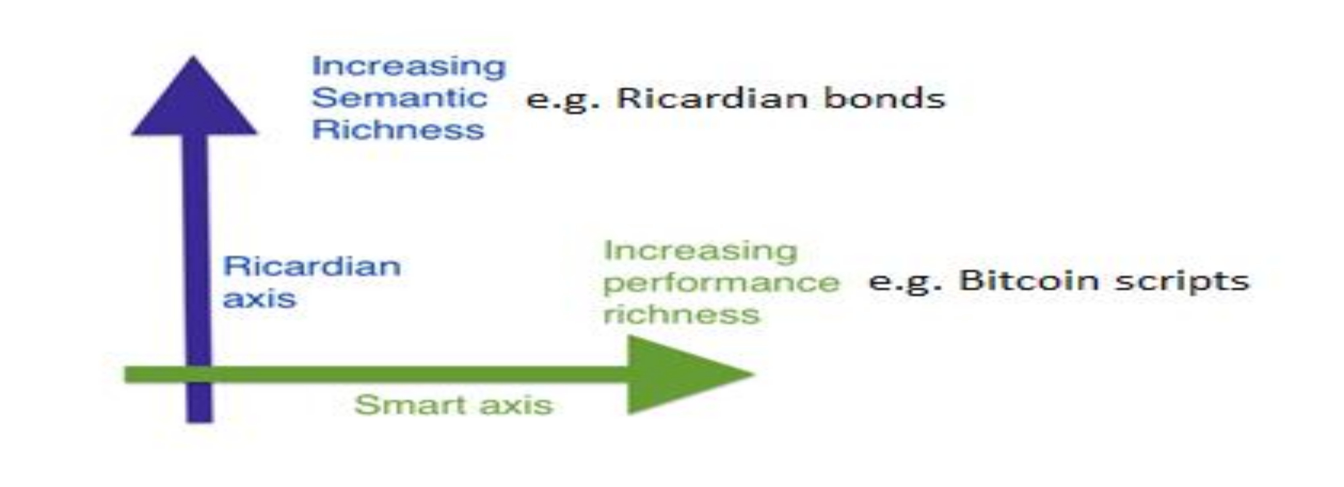
"hash" or secure identifier

genesis transaction

user txs

Difference

- A smart contract does not include any contractual document and is focused purely on the execution of the contract.
- A Ricardian contract, on the other hand, is more concerned with the semantic richness and production of a document that contains contractual legal prose.



Semantic Contract Types

- The semantics of a contract can be divided into two types:

1. Operational semantics(performance) – Defines the actual execution, correctness and safety of the contract.
2. Denotational semantics(legal semantics) - Concerned with the real-world meaning of the full contract.

A smart contract is made up to have both of these elements (performance and semantics)embedded together, which completes an ideal model of a smart contract.

Representation and Examples

- A Ricardian contract can be represented as a tuple of three objects, namely *Prose, parameters* and *code*.
  - Prose represents the legal contract in regular language;
  - Code represents the program that is a computer-understandable representation of legal prose;
  - Parameters join the appropriate parts of the legal contract to the equivalent code.
- Ricardian contracts have been implemented in any systems, such as CommonAccord, OpenBazaar, OpenAssets, and Askemos.

Smart Contract Templates

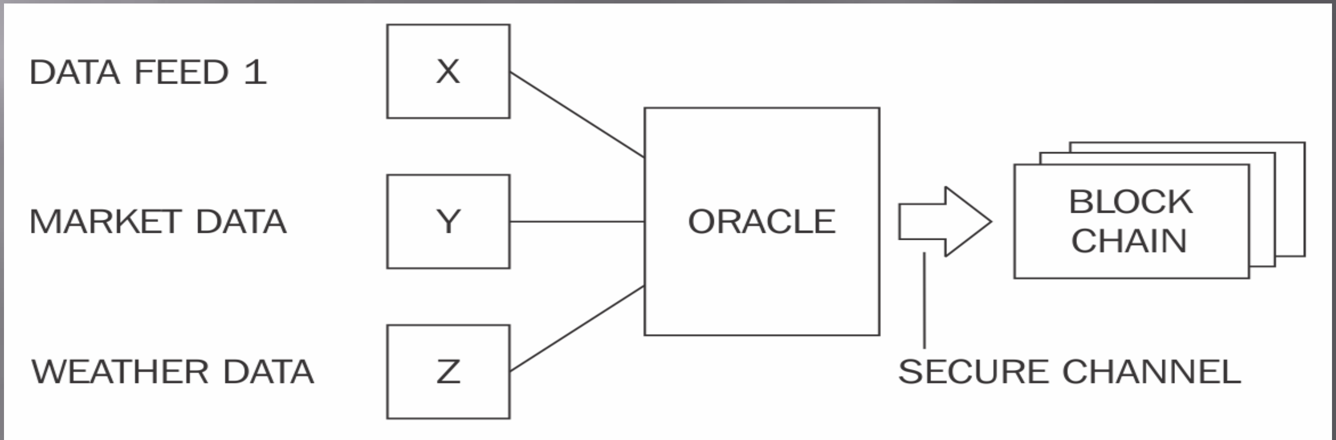
- The idea is to build standard templates that provide a framework to support legal agreements for financial instruments.
- This was proposed by *Clack et al* in their paper named *Smart Contract Templates: Foundations, design landscape and research directions*.
- The paper also proposed that domain specific languages should be built in order to support design and implementation of smart contract templates. e.g. Solidity and Serpent.

Oracles

- Oracles are an important component of the smart contract ecosystem.
- The limitation with smart contracts is that they cannot access external data which might be required to control the execution of the business logic.
- For example, the stock price of a security that is required by the contract to release the dividend payments.
- Oracles can be used to provide external data to smart contracts.

Oracles and Smart Contracts

- An Oracle is an interface that delivers data from an external source to smart contracts.
- Depending on the industry and requirements, Oracles can deliver different types of data ranging from weather reports, real-world news, and corporate actions to data coming from **Internet of Things (IoT)** devices.
- Oracles are trusted entities that use a secure channel to transfer data to a smart contract.



Reliability and Trust in Oracles

- Oracles are also capable of digitally signing the data proving that the source of the data is authentic.
- Oracle is a subscription based data push or pull service.
- It is also necessary that Oracles should not be able to manipulate the data they provide and must be able to provide authentic data.
- Even though Oracles are trusted, it may still be possible in some cases that the data is incorrect due to manipulation.
- This validation can be provided by using various notary schemes

Types of Oracles

1. *Standard or Simple* Oracles – *Centralized*
2. *Decentralized* Oracles - These types of Oracles can be built based on some distributed mechanism. It can also be envisaged that the Oracles can themselves source data from another blockchain which is driven by distributed consensus, thus ensuring the authenticity of data.
3. *Hardware* Oracles - Introduced by researchers where real-world data from physical devices is required. This can be achieved by using tamper-proof devices.
4. *Smart* Oracles - Smart Oracles are basically entities just like Oracles, but with the added capability of contract code execution. Smart Oracles proposed by *Codius* run using Google Native Client. https://www.codius.org/.

Data Communication Methods

- In bitcoin blockchain, an oracle can write data to a specific transaction via an OP_RETURN Opcode, and a smart contract can monitor that transaction and read the data.
- Various online services such as http://www.oraclize.it/ and https://www.realitykeys.com/ are available that provide oracle services.
- Also, another service at https://smartcontract.com/ is available which provides external data and the ability to make payments using smart contracts.

Oracle Data Authenticity

- In order to prove the authenticity of the data retrieved by the Oracles from external sources, mechanisms like TLSnotary can be used which produce proof of communication between the data source and the oracle.
- This ensures that the data fed back to the smart contract is definitely retrieved from the source.
- More details about TLSnotary can be found here https://tlsnotary.org/.

Deployment on Blockchain

- Smart contracts may or may not be deployed on a blockchain but it makes sense to deploy them on a blockchain due to the distributed consensus mechanism provided by blockchain.
- Ethereum is an example of a blockchain that natively supports the development and deployment of smart contracts.
- Smart contracts on Ethereum blockchain are usually part of a larger application such as **Decentralized Autonomous organization (DAOs).**

In Bitcoin

- In bitcoin blockchain the lock_time field in the bitcoin transaction can be seen as an enabler of a basic version of a smart contract.
- The lock_time field enables a transaction to be locked until a specified time or after a number of blocks, thus enforcing a basic contract that a certain transaction can only be unlocked if certain conditions (elapsed time or number of blocks) is met.

Attack on Smart Contract

- The DAO is one of the highest crowdfunded projects, and started in April 2016.
- This was basically a set of smart contracts written in order to provide a platform for investment.
- Due to a bug in the code this was hacked in June 2016 and an equivalent of 50 million dollars was siphoned out of the DAO into another account.
- This resulted in a hard fork on Ethereum in order to recover from the attack.

Learning from the Attack

- This attack highlights the dangers of smart contracts and the absolute need to develop a formal language for smart contracts.
- The attack also highlighted the importance of thorough testing.
- There have been various vulnerabilities discovered in Ethereum recently around the smart contract development language.
- Therefore it is of utmost importance that a standard framework is developed to address all these issues.
- Some work has already begun, but this area is ripe for more research in order to address limitations in smart contract languages.

Summary

- This chapter started by introducing a history of smart contracts. As there is no agreement on the standard definition of a smart contract, we attempted to introduce a definition that encompasses the crux of smart contracts.
- An introduction to Ricardian contracts was also provided, and the difference between Ricardian contracts and smart contracts was explained.
- The concept of smart contract templates was discussed, on the subject of which high quality active research is currently being conducted in academia and industry.
- Some ideas about the possibility of creating high level domain-specific languages were also discussed to create smart contracts or smart contract templates.
- Later, the concepts of Oracles was introduced followed by a brief discussion on the DAO, and security issues in DAO and smart contracts.