# Distributed Pseudonymous Blockchain Oracles for Smart Contracts

Matthew Whittington
matt@factom.com

Contributors: Tiana Laurence, Pete Harris, Andrew Yashchuk

With the advent of smart contracts there is a need for external data to be made accessible to those contracts from the outside. These are generally referred to as Oracles. Acquiring data from an Oracle may involve personal knowledge of an individual, an account set up on a website or an established business relationship. The reasons for needing real world knowledge of these Oracles are trust and a process in place to convey the questions that need to be answered. The costs of this trust is almost always capital, identity information or both.

### This proposal has five parts:

**I. Why should I use a blockchain instead of traditional system?**

A blockchain implementation of transactional data avoids counterparty data risk and avoids the onboarding costs associated with customer acquisition.

**II. Where do I put a question?**

A blockchain centric process for submitting queries r tasks and receiving answers in a pseudonymous fashion.

**III. Who Is answering my question?**

A pseudonymous (Hash Identity) method for identifying the Oracle.

**IV. Why should I trust the Oracle?**

A method for building a reputation based on the Oracles hash identity.

**V. When an Oracle answers, how do I know it is really them?**

A method for verifying an Oracle's identity in their answers.

**VI. What is in it for an Oracle?**

A method for allowing micropayment incentives for Oracles.

**Additional items**

Bitcoin and Blockchain are synonymous. However, several core issues have become overwhelmingly clear to those wishing to utilize the bitcoin blockchain at scale. Entering data into the Bitcoin blockchain is prohibitively expensive at volume. In addition to this, the Bitcoin blockchain cannot handle high transactional volumes. A Blockchain that does account for these issues is Factom. The Factom blockchain offers an attractive alternative. It is orders of magnitude less expensive and has orders of magnitude more capacity for transaction volume. It also has built-in layers of redundant security that other blockchain do not offer. The Factom Blockchain anchors itself into the Bitcoin blockchain, among others, to take advantage of the security of Bitcoin's hash rate. The layering effect of security ensures the immutability of its blocks. Another feature that the Factom blockchain offers, that helps those wishing to do large volumes of transaction, is it also has a theme tracking capability. What this enables is the ability to chain together data you care about and forget the rest of the data set. Unlike the Bitcoin Blockchain that requires you have all of the data to prove any of the data, Factom allows you to prove your data set without needing all the data in Factom.

In reference to the technical accuracy of this section, I could use many terms for what I am calling Questions in this proposal. I am going to use 'Question' throughout for consistency. If I write 'ask a question,' it could mean post a query or submit a document. I am stating it as asking and answering for consistency, not for technical accuracy. Technically accurate terms will be used in the technical parts of the document.

On another note, a blockchain doesn't 'do' anything. It is an immutable list of entries. This proposal implies at least a rudimentary process that uses the blockchain entries. This front end will have at least one Oracle and one lookup for Oracles to function.

# Why should I use a blockchain instead of traditional system?

With a traditional two party system, verifying transaction volumes or details requires the participation of both sides of the transaction. In cases where the opposing side of a transaction is transitory in nature, this can be difficult or impossible. This causes counterparty data risk with any transactional system you may have in place today. By nature, public blockchains are not controlled by a single entity. The transaction details for any business that utilizes them can be audited at any time even if one of the parties is no longer present. Verifying transactions that you have taken a part in does not require you to get in contact with the opposing entity and be subject to their willingness or ability to provide you with transaction details from their systems.

Data or service providers that do not require extensive regulatory processes can easily use the method described herein to provide their services at microtransaction levels without lengthy customer onboarding to ensure payment. This also allows for detailed transaction and payment information at any time in the future with no input from a 3rd party entity or payment delays for services rendered.

If your organization does require regulatory compliance, the Identity aspects of this system can be used to provide proof of compliance.

# Where Do I Put A Question?

To have a system that receives and answers questions, there needs to be an agreed upon location or method to ask these questions. This implementation in the Factom Blockchain will have one chain that is to be watched for questions (chain id). The questions will be posted as entries into that chain. Answers and possible Offer Entries (see part V) will be posted to the same chain. Each Oracle will have its question and answer chains defined in its chain creation. The chain will be registered in the Acolyte Oracle Registration Chain (da7ac1717e0b2e0c4609cdc2c78289e841239d2779fd23632ab226510184aae4) so it can be located by the service . That is the Oracle Registry Chain. To be recognized as an Oracle to this system, the entry will have minimum formatting requirements (appendix A). It will also have to be signed by an Identity (see part III) recognized by this system. In addition to defining what chain will be watched, the initial entry will also define the format of questions and answers. This could be anything from free form text, XML, JSON, encrypted multipart posts, a link to a file in a blockchain file storage system, or any other format that is demanded by the Oracle provider. For this proposal, all of our Oracles will use json documents. Third party Oracles that wish to leverage this project's Identity (part III) and reputation (part IV) implementation certainly can. Or, they can implement their own front end processes to handle the blockchain entries themselves. One advantage of using blockchain entries in this fashion is that, as a distributed ledger, neither the Oracle or the submitter needs to present an identifiable location or identity to a centrally controlled point in the system.

# Who Is Answering My Questions?

Identities in this system involve the creation of 2 Factom chains and an entry into the Oracle Registry Chain mentioned in the last section. The first chain that is created contains the public keys that comprise the Identity. This chain is to be used to maintain the Identity itself. The second chain is created to hold reputation entries (see next section III) that concern that Identity. It also may contain entries that concern the Identity but do not change the Identity itself. These may be keep alives, heart-beats or other functions that the Oracle itself may need for maintenance. It also holds changes to the Oracle such as formatting changes of requests and responses.

Identity creators may stop with no other identifiable information than the public keys. They may add additional information such as web sites, DUNS numbers or other information that identifies the entity beyond just the keys.

The Identity format is based on the one that is used by Factom to identify the Federated Servers that is described at:

https://github.com/FactomProject/FactomDocs/blob/master/Identity.md

The primary difference between the Identity creation described in the above link and the oracle Identity creation is that HEX encoded values the Factom Identity Chains begin with ‘da7a’ instead of ‘888888’ as expected by the Factom Federated Server keys.

Each Identity has a public key that all of its entries (see section IV) should be signed with.

# Why should I trust the Oracle?

The second chain that was created for an Identity is used for reputation entries. Entries into this chain contain the entry hashes for the question asked, the answer given, the positive or negative rating upon the answer of that question. This rating is to be signed with the private key of the requester of that question.

Only entries that match questions, answers and signature keys will be included in reputation calculations. The determination of trustworthiness is up to the Questioner of the Oracle. The default reputation determination (ie. Our implementation) includes:

I. the length of time since oracle creation
Ii. the number of questions answered
Iii. the number of requestors
iv. the identity age of the requestors
v. the reputation of the Questioner

Identities must be used to be an Oracle in this system. They are optional for the submittal of questions IF the oracle concerned chooses to answer them without an Identity. Only requestors with Identities can make valid reputation entries.

Reputation Entries about Oracles are used to gauge the worthiness or accuracy of an Oracle. Reputation Entries about Questioners are used to gauge the value of their opinion about Oracles.

# When an Oracle answers, how do I know it is really them?

As with anything placed into a blockchain, if you wish to prove that it is an authentic entry, signatures should always be used. In this project, the first External ID is a signature of the content. The only exception to this is the original creation of Oracle or Identity chains that can use the first External ID to provide the public address of the key used to sign all of the proceeding entries. This number CAN change. In the case of a compromised private key or just routine account maintenance by an Identity holder, those changes can be found in the Identity Chain of that Identity. The format of the Identity chains is described in Appendix A.

Any entry used in this system that has an API Key will be treat as an Identity Chain and will expect the content to be signed. That is the expectation of the Oracles set up in this implementation. The expectations and handling of these signatures are up to the Oracle creators and Questioners implementation.

# What is in it for an Oracle?

There are a few process steps that can be used for an Oracle. The steps I am about to describe allow for transactional payments for data. It is up to Oracles to determine which of these steps are to be implemented.

A Full Oracle transaction works like this:

I. A Questioner posts a Question entry to a watched Chain.
II. One or more Oracles post a bid for the answer along with a payment method.
III. The Questioner determines which Oracle to use and pays them.
IV. The paid Oracle posts a receipt for the answer
V. The Oracle posts a reputation entry into the Questioner's Identity Management chain
VI. The paid Oracle posts the Answer
VII. The Questioner enters a reputation entry into the Identities management chain.

The first implementation of this will only implement steps I and VI.

**Appendix A**

Oracles require interaction with 6 chains. Four are generated by the Oracle creator and two are pre existing and watched by Factom Acolyte.

Examples provided below will use the following private keys. They are 32 byte arrays.
Key 1: 0000000000000000000000000000001
Key 2: 0000000000000000000000000000002
Key 3: 0000000000000000000000000000003
Key 4: 0000000000000000000000000000004

Signing Key: 0000000000000000000000000000005

If an Identity has not been created, there are two Chains and two entries to be created. Many Oracles can be attached to one Identity or an Identity can be created for each.

# Identity Chain

An Identity Chain has external IDs with the values 0, “ACOLYTE IDENTITY KEY”, 4 public (keys as Factom Entry Credit Addresses) and a numeric value that will cause the chain ID to begin with ‘da7a’ when the chain is created. Chain IDs are created when the initial make chain command is sent. The way the ID itself is derived is to sha256 each external id in the entry, then append those hashes and create an sha256 of that appended byte value.

Identity Chain ID will equal: (+ means append in the following)
sha256(
    sha256(“0”) + sha256(“ACOLYTE IDENTITY KEY”) + sha256(public key1) +
    sha256(key2) + sha256(key3) + sha256(key4) + sha256(number)
)

**For example:**
**Version**: 0
**Type**: ACOLYTE IDENTITY CHAIN
**Key** 1: EC2LwDdXFJ3148i3XSzJi1qpxzUrWiHv153xaro3PacUCE9wGEu2
**Key** 2: EC2eJJVx4MvaCYKHc37dnAQghxG5k9D7hU9WvdoKA5hjsBqMMzVQ
**Key** 3: EC3cPnuGWV8R36ufrZCmJfPeAd3w5bkaqWBS7MX7L2YrqPGreDJB
**Key** 4: EC3giNbxZqhBefDszymnxkXs9v4ADa6UezLnwMQKyEmB3xMJxHcB
**Nonce**: 56660

**The sha256 of the appended values:**
DA7A34162343565E246D48007D620AF01A47AA92271C2C4EA954B619757713E4

**Factom-cli example of actual chain creation:**

factom-cli mkchain -e 0 -e "ACOLYTE IDENTITY CHAIN" -e
EC2LwDdXFJ3148i3XSzJi1qpxzUrWiHv153xaro3PacUCE9wGEu2 -e
EC2eJJVx4MvaCYKHc37dnAQghxG5k9D7hU9WvdoKA5hjsBqMMzVQ -e
EC3cPnuGWV8R36ufrZCmJfPeAd3w5bkaqWBS7MX7L2YrqPGreDJB -e
EC3giNbxZqhBefDszymnxkXs9v4ADa6UezLnwMQKyEmB3xMJxHcB -e 56660

**Chain ID Created:**
DA7A34162343565E246D48007D620AF01A47AA92271C2C4EA954B619757713E4

# Identity Signing Key:

Now we need to make an entry assigning this identity's Signing Key. This Entry will be signed by Key1 (private key) of the Identity. This same entry (format) can be made for future changes to the Identities signing key. The date stamp keeps it unique (per day) against a replay attack.

The Entry has 4 values
**Version**: 0
**Type**: ACOLYTE SIGNING KEY ASSIGNMENT
**Datestamp**: 20160902
**Signature**: signature of signing key (entry contents signed by key1)
9BB3ED85E5AEEFE59E905A5463C702F112ACF2671D0BED0DA4B05B8ACC17FBD90F701
89BA12194B31494EBDBBDEC00283DEC458DB453C0B1C985A1FF4ECD2503

**Contents of Entry**: new public signing key as Entry Credit Address
EC3gyAd5yiGUaUDsEWe9MW1ocgsH2tR9Sbp58vUsM1oH3WkUyray

**Entry Created**:
5C051FBC8F10D4F6996E03470E7CCAD1A3E219CE29A1455DC689EB4962D34801

# Register Identity Chain

The Identity needs to be registered in chain
da7aaeaae20d77e03e3ea57e90d9b46efc8fa187cb79836e046375033b7680aa. This is the Acolyte Identity Registration Chain. This entry has 4 External IDs. The content is optional.

The External IDs are:
**Version**: 0
**Type**: ACOLYTE IDENTITY REGISTRATION
**Identity Chain:**
DA7A34162343565E246D48007D620AF01A47AA92271C2C4EA954B619757713E4
**Signature**: Identity Chain signed by Identities Signing Key.
2443A49076987E68CF8D5D976BBCAAFEA45CBC3B587C999F9BCD1C88D18D4AE60E3CF
C655BB6A3E77C8B1282FE869C436E9AB28A5FC093A8D9F4666B75FC1707
Factom-cli put -e 0 -e “ACOLYTE IDENTITY REGISTRATION” -e
da7a1c0dd6b0982f03df164b3101f6c9e884dd6ecf7fe23dbb1704221fd003b6 -e

**Content**:
DA7A34162343565E246D48007D620AF01A47AA92271C2C4EA954B619757713E4

# Identity Management Chain

The Identity Management chain hold information about the Identity that does not change the Identity itself. These can be application keep alives, reputation entries ao any other information that is needed to use the Identity. This is a Chain creation event, so a nonce is used to make the Chain ID start with ‘DA7A’.

The chain creation of the Identity Management chain has 4 External IDs. These are:
**Version**: 0
**Type**: ACOLYTE IDENTITY MANAGEMENT CHAIN
I**dentity Chain ID**:
DA7A34162343565E246D48007D620AF01A47AA92271C2C4EA954B619757713E4
**Nonce**: 24448991

**Content of Entry:**
This is the Identity Management Chain for Aculyte Identity Chain
DA7A34162343565E246D48007D620AF01A47AA92271C2C4EA954B619757713E4.
(content data is up to user)

**Chain Created:**
DA7A19EAE2C995F2A30A86199BF7C6280F092EC18D278731BDC9F23250D67B3E

# Identity Management Chain Registration

This is make in the Identity chain so the Identity Chain can find it.

**Version**: 0
**Type**: ACOLYTE IDENTITY MANAGEMENT CHAIN REGISTRATION
**Identity Managment Chain ID:**
DA7A19EAE2C995F2A30A86199BF7C6280F092EC18D278731BDC9F23250D67B3E

51D4B483473C0A83E45F968E732BF128750D1D37514A0854EC9F7C759B03A14F5D0BDB1
E58B13EC15BAA49F2C7B270CC260178593E3DA632C53DE040B3AE9A0E

**CONTENT**:
DA7A19EAE2C995F2A30A86199BF7C6280F092EC18D278731BDC9F23250D67B3E
CHAIN USED:
DA7A34162343565E246D48007D620AF01A47AA92271C2C4EA954B619757713E4

**Entry Created**:
550627FA23D87C91681FD5572C3117EB78B8BE569A0F32196D79185B4CA83F63

# Oracle Chain

Oracle Creation is very similar to Identity Creation.  The primary difference is that Oracles do not have 4 private key pairs.  Instead of those values, only a reference to the Identity Chain is used.
The Oracle chain can be unique to an Identity or Identities can have more than one Oracle. This is up to the Oracle creator.

# Acolyte Oracle Chain Creation

An Acolyte Oracle is a distinct chain that keeps track of the Oracle itself.  These are data definitions and a description of the data or process that the Oracle was created for.  This is a chain creation so it uses a nonce.  There are three defined external IDs (first second and last). The rest aer user defined.

**External IDs:**
**Version**: 0
**Identity Chain ID:**
DA7A34162343565E246D48007D620AF01A47AA92271C2C4EA954B619757713E4
**User Defined:**  test
**User Defined:** oracle
**Nonce**: 15682

**CONTENT**:
hi  (describe your oracle in this)

**CHAIN CREATED:**
DA7ADDD98BEAA25D5478E870648BFCA6652D720EA07FE45BAB85CF3345AB8742

# Oracle Management Chain

Oracle Chain creation has four Exterior IDs.
**Version**: 0
**Type**: ACOLYTE ORACLE MANAGEMENT CHAIN
**Identity Chain ID:**
DA7ADDD98BEAA25D5478E870648BFCA6652D720EA07FE45BAB85CF3345AB8742
**Nonce**: 69662

**CONTENT:**
This is the Oracle Management Chain for Acolyte Oracle Chain DA7ADDD98BEAA25D5478E870648BFCA6652D720EA07FE45BAB85CF3345AB8742. It holds informational entries concerning the Oracle Chain that do not alter the Oracle itself.

**CHAIN CREATED:**

DA7AF4D61647EF097DB6D3E801025E7B291E9CE07478D5B479C4F29A35120FC2

# Oracle Management Registration

Now the management chain needs to be registered with the Oracle it is managing.
There are 4 External IDs:
**Version**: 0
**Type**: ACOLYTE ORACLE MANAGEMENT CHAIN REGISTRATION
**Management Chain ID**:
DA7AF4D61647EF097DB6D3E801025E7B291E9CE07478D5B479C4F29A35120FC2
**Signature of entry content**:
A3E6F8D67C983EB2D9BE224C7BE3944AECB4AFBF65C328C98D7D9ABDA1D37081EFF10
08568719E5ED88B2C2FB268F5A9DD9BF228E54CBD92C0F5DB6B9B07A001

**CONTENT**:
DA7AF4D61647EF097DB6D3E801025E7B291E9CE07478D5B479C4F29A35120FC2

# Acolyte Oracle Registration

Now we need to register this into the Acolyte Oracle Registration Chain.  This is the chain that Acolyte uses to discover Oracles.
**Version**: 0
**Type**: ACOLYTE ORACLE REGISTRATION
**Oracle Chain ID:**
DA7ADDD98BEAA25D5478E870648BFCA6652D720EA07FE45BAB85CF3345AB8742
**Signature of Content:**
704B5B068DE3AAFE91A28BDF834E0C21A56A8052E3ED82424D6BBE4FF2A114D026B3AE
0F90110AAA1C5569529D29C3293B757139A8DEC419B63BC1C4102E4407

**CONTENT**:
DA7ADDD98BEAA25D5478E870648BFCA6652D720EA07FE45BAB85CF3345AB8742