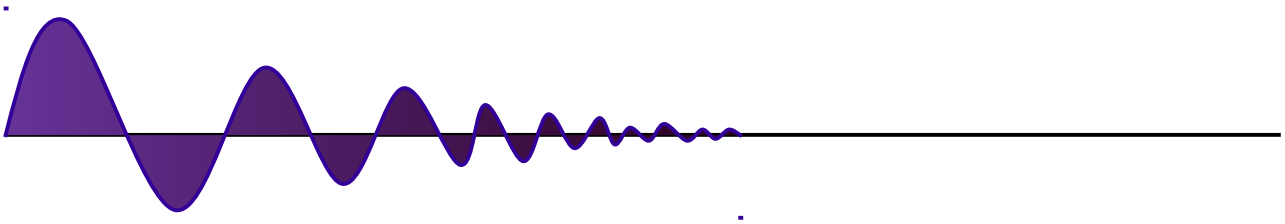


Permissioned Ledgers on Open Blockchains

John P. Conley[1]

Vanderbilt University

May 2017

Abstract

The blockchain space appears to be complicated and disorganized with new and innovative uses being proposed all the time. We will argue these use cases can be broken down along a small number of structural fundamentals. Viewing blockchains from this standpoint suggests that many important applications are limited by conflict between the openness needed for credibility, and the privacy that many types of records require. The main purpose of this paper is to propose a set of solutions to this conflict. We outline a combination of smart contracts and record formats that implement granular access control on an open, trustless, distributed blockchain. We also show how these contracts can be extended to implement access control on demand and to notify record owners when other authorized agents access their data. Finally, we suggest an architecture that allows searches of encrypted data held in blockchains without compromising access and data control

1 j.p.conley@vanderbilt.edu. The author takes sole responsibility for the content of this paper.

John P. Conley
Department of Economics
Vanderbilt University
Nashville, TN, 37235

jpc@jpconley.com
http://jpconley.com

Introduction

Blockchains are distributed, append only ledgers, usually validated using some type of consensus mechanism. They can hold time-stamped, digitally signed, records of any kind, and make them immutable. Blockchains can be configured to preserve the anonymity of its users, and if copies of the ledger are widely distributed, censuring the record is difficult or impossible Neither access to, nor validation of, records in a blockchain requires trust in the good behavior of any central authority.

The blockchain space appears to be complicated and disorganized, with new and innovative uses being proposed all the time. We will argue these use cases can be broken down along a small number of structural fundamentals. Viewing blockchains from this standpoint suggests that many important applications are limited by conflict between the openness needed for credibility, and the privacy that many types of records require.

The main purpose of this paper is to propose a set of solutions to this conflict. We outline a combination of smart contracts and record formats that implement granular access control on an open, trustless, distributed blockchain. We also show how these contracts can be extended to implement access control on demand and to notify record owners when other authorized agents access their data. Finally, we suggest an architecture that allows searches of encrypted data held in blockchains without compromising access and data control.

Blockchain and Record Types

Despite the variety of uses that are being proposed, there are really only two fundamental types of blockchains: object transfer and document certification.

Object transfer: The distinguishing feature of object transfer blockchains is that the records are interdependent in the sense that they must collectively satisfy accounting rules or business logic to be valid. As a result, object transfer records have strictly defined structures and must include enough cleartext information for the miners or other validators to confirm that the rules are followed. Transfer objects may be homogeneous or heterogeneous, and this leads to two slightly different transaction rules for the underlying blockchains.

> **Balance transfers:** Homogeneous transfer objects have no meaningful existence as individual items. Examples include bitcoins, gold, commodities, stocks, and bank and credit card dollar balances. For a block to be valid, the basic accounting rule that input must equal output in the appropriate sense must be satisfied. The blockchain itself contains a compete record of transactions for each account, agent, or PPK (Public/Private Key) address. Adding these up gives the net balance credited to each address when the last block was written. The accounting rule requires that sum of outgoing transfer requests from each address to be written to the current block cannot exceed this balance. Double spending is not allowed, and each incoming transfer much have a corresponding outgoing transfer. Of course, objects may enter or leave the system, and be created or destroyed in some cases. As long as the rules that govern these processes are followed as well, blocks can be validated.

Ownership or title transfer: Heterogeneous transfer objects are different from one another and so cannot be grouped as balances. Each object must be allocated to an address as an individual. In some cases, this is due to the nature of the object. No house or piece of land is exactly like any other. Used cars, livestock, art, and most objects that have a physical existence fall into this category. In other cases, the individuality of the transfer object is artificial. Units of cryptocurrency, stock certificates, and other financial assets may be assigned arbitrary serial numbers to give them unique identities even though they are otherwise indistinguishable from one another. It may also be desirable to take a similar approach to real world items that are essentially homogeneous for accountability and inventory control reasons. For example, new cars, guns, ammunition, manufacturers' lots of chemicals or controlled substances, and shipments of produce, may be indistinguishable to a user, but placing a serial number or other identifier on each unit allows a chain of custody, an audit trail, and a mapping of specific units to specific owners to be created. For a chain to be valid, each transfer object must reside at one and only one address. In turn, a block is valid if the all transfer requests move an object from the address that owns it to another valid address, and no object has more than one transfer request to be written to the current block.

Document certification: All other blockchains have no native logic that must be satisfied for a block to valid. There may be rules about who is allowed write records to the chain, what the format of records must be, and so on, but miners or other validators are able to decide whether a new record should be included in the current block without looking at anything besides the proposed record itself. In other words, records proposed for the current block have no bearing on one another's validity. Of course, it may be that data contained in different records conflict somehow, but checking for this should not be done at the level of block validation. For example, a medical record for a patient might say that a certain test was positive and another written at the same time might say that the test was negative. The signatures of the machine that produced the results, the technician who ran the test, or the doctor who interpreted its meaning will be part of both records. If checking these details fails resolve the conflict, then the audit trail to the party who may have entered data incorrectly is clear. There are many ways that conflicts between various aspects of records, failures to follow required business logic, omissions, redundancies, or simple error might lead to conflicts. The blockchain and its validators are not well-equipped to understand what might constitute an inconsistency, and placing the burden of checking on validators risks making errors worse. Thus, document certification chains simply hold immutable, nonreputable, signed, time-stamped records. The correctness, truth, meaning, and value of these records is for users to determine.

Access to Records

Bitcoin transactions are written on an open blockchain. Copies are widely distributed, maintained on public servers, and can be viewed by anyone who cares to. The distributed and open nature of bitcoin's blockchain contributes to its credibility as a ledger and makes it essentially impossible for any central authority to censor. Bitcoin transaction records are not encrypted and so can be read, understood, and audited by anyone with access to the blockchain. The identities of the parties involved in transactions are also unencrypted, but consist of PPK addresses that anonymize the real-world identities of bitcoin owners.

Blockchains can also be closed in the sense that they are maintained on private servers with access restricted to a group of stakeholders. In almost all cases, however, records in such chains includes data that stakeholders do not wish to share with one another. For example, Ripple provides bank and foreign exchange settlement services to banks. Ripple's ledger is closed, but access to individual transaction records is limited to the parties involved. Making a blockchain private is not enough to protect the confidentiality of the data it contains. Finer access controls at the level of individual records are usually needed.

Note that both open and closed blockchains may anonymize the identities of agents mentioned in records. Anonymizing agents is simply a matter of choosing what sort of cleartext data to include in a record. Both open and closed blockchains may include data in cleartext, ciphertext, or a combination of both. Thus, the only security difference between closed and open blockchains is that the set of stakeholders or the public at large, respectively, has full access to whatever the blockchain contains. There is seldom anything to gain by keeping a blockchain closed, and if it contains privileged data, protecting it depends on the unverifiable good behavior of the stakeholders.

Access control comes down the question of the permissioning structure used when writing records to the chain, not openness, closeness, anonymity of users, or generalized encryption.

Implementing Granular Permissioning in Open Blockchain

Traditionally, data is stored in private databases and access to specific records is controlled through passwords or other types of authentication. In the case of blockchains, however, if an agent can see any record, he can see them all. Even closed blockchains are completely visible to stakeholders.

What is needed is a way to restrict access on a record by record bases to a specific set of agents on an open, distributed and trustless blockchain. Permissioned agents should be able to read, verify, and prove the veracity of records to others without relying on any other agent, trusted or not. Below, we propose a solution that accomplishes this simply, and at low cost.

We start with a contract, a bill of lading, a public record, a bitcoin trade, or any other document or data to a be added to a blockchain. We refer to agents who signify agreement with, or attest to the truth of, the contents of the document with digital signatures as **parties** to the record. For example, an agency putting documents in a public blockchain, the set of agents signing a contract, and the pair of agents on each side of a transaction are parties to the resulting record. We refer to agents who are not parties but who are granted access to the record as **permissioned readers**. For example, the IRS or other federal agency, the final receiver of a package in a logistics or shipping chain, a company's auditors, or the manager of an employee, might be permissioned readers of certain records that they themselves have not signed. Smart contracts may also be involved in creating or accessing records. We discuss this later sections.

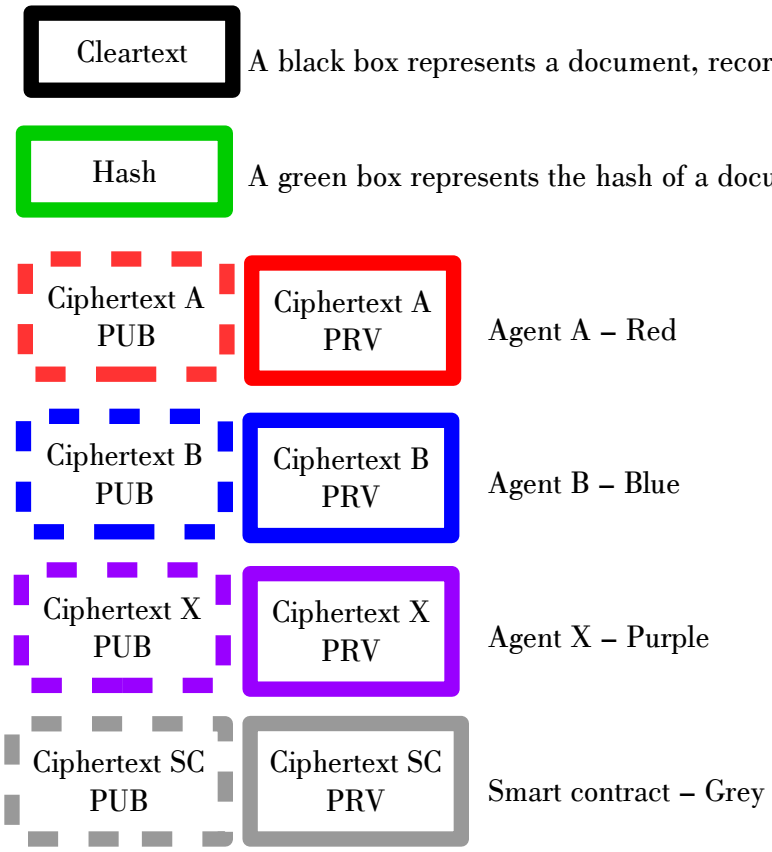
To be slightly more formal:

Agents A and B: Parties to the document. There may be one, two, or more such parties, and all digitally sign the document

Agent X: Permissioned readers. There can be any number of such agents, and none sign the document in the record

Smart contract: Autonomous code that may be involved in creating or controlling access to a record. Smart contracts may be given PPK pairs and SSL certificates so that they can sign or access records and documents if required.

Data or documents may be in clear text, hashed, or encrypted with either the public or private key of these agents:

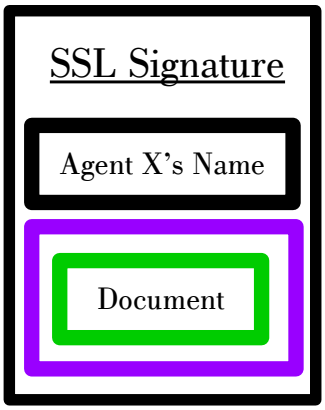


Encryption of data using a public key signified by a dashed boxed.
Encryption of data using a private key is signified by a solid box.

Recall that if agent X wishes to sign a document, he takes its hash and encrypts it with his private key. By finding the SSL certificate of associated with agent X's name in the PKI[2] (Public Key

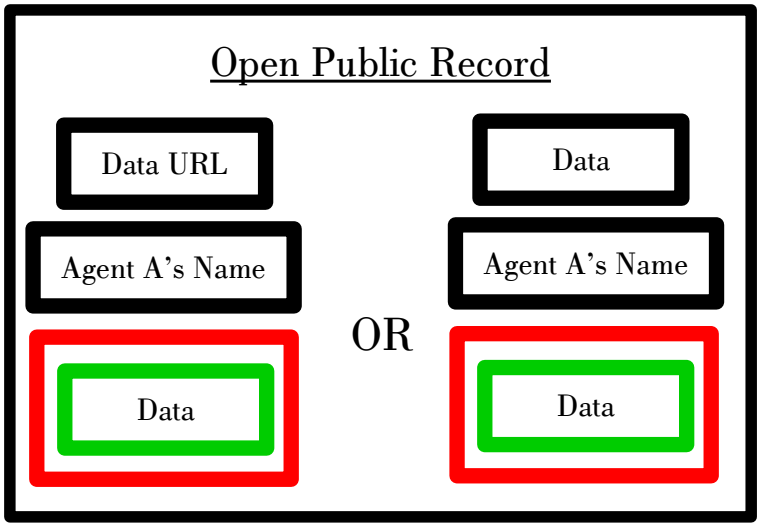
2 An SSL/TLS (Secure Socket Layer/Transport Layer Security) certificate associates a name or other identity with a public key. In addition to a name, public key, and some technical information, the certificate file contains a digital signature of the Certificate Authority (CA) that issued the certificate. Collectively, this information says that a specific CA (Microsoft or VeriSign, for example) attests that a certain individual is in control of the private key associated with

Infrastructure), anyone can use the public key it contains to decrypt the hash and compare it to his own hash of the cleartext document. If the hashes match, he knows that (a) the cleartext document he has is the same as the one that was signed, and (b) agent X encrypted the hash and so attests that document is genuine/true/binding.



Nonpermissioned records

The most straightforward uses of digital signatures is to create an open public record. A county clerk, for example, could uses his department's public key to sign a cleartext copy of deed or marriage certificate. The document and signature together could then be written into an an open blockchain. Users could then access the record in the blockchain and know that the document they see is the exact one that the country clerk attested was correct when the record was created. The data or documents could be placed in the chain itself or stored offline. This is an example of a nonpermissioned record and would look like the following:



the public key in the certificate. The signature includes the public key of the CA encrypted with its private key. If we can find the CAs SSL certificate, we can verify that the attestation is genuine. But how do we know that the SSL certificate we find really belongs to the CA? Perhaps another agent decided to publish a fake a certificate associating "VeriSign" with his own public key. Fake or not, this SSL certificate is also singed by another CA, whose certificate is singed by other CAs, and so on. If we can find one SSL certificate on this backward set of links that we think is genuine, this validates the chain from that point forward. The PKI, in turn, is a set of agents, policies, protocols, and equipment that creates, revokes, stores, distributes and validates SSL certificates.

Permissioned records

Medical records, transcripts, diplomas, and payment records, contain private information. Only specific agents should be permissioned to read them. Below is a simple approach to creating a granular permissioning structure on an open blockchain.

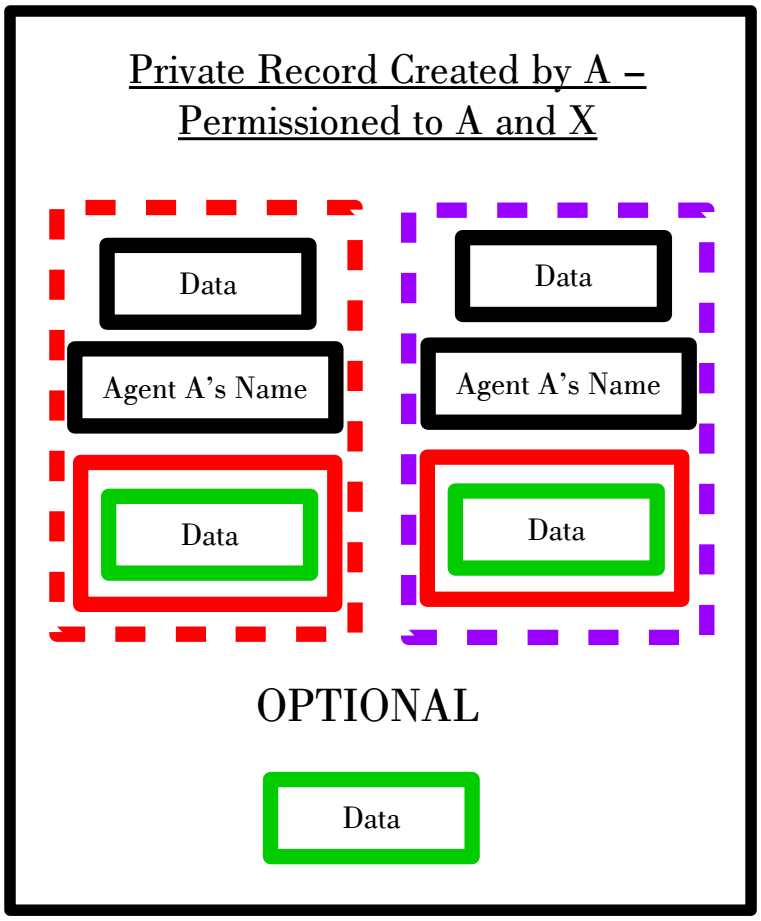
1. Agent A creates the data, takes its hash, encrypts it using his private key, and adds his identity in cleartext so that his public key can be found on the PKI. In other words, he signs the data he creates to attest to its truth.

2. Agent A encrypts the cleartext data and his signature using his public key. The the encrypted cleartext data and signature pair can only be read by the holder of A's private key. Thus, publishing this encrypted pair in an open blockchain does not make it public. Access is restricted to agent A.

3. To make the data available to permissioned readers, agent A finds their public keys using the PKI. He then uses these keys to encrypt the cleartext data and signature pair. Of course, a URL pointing to the data encrypted with these public keys could be placed in the blockchain instead of the data itself.

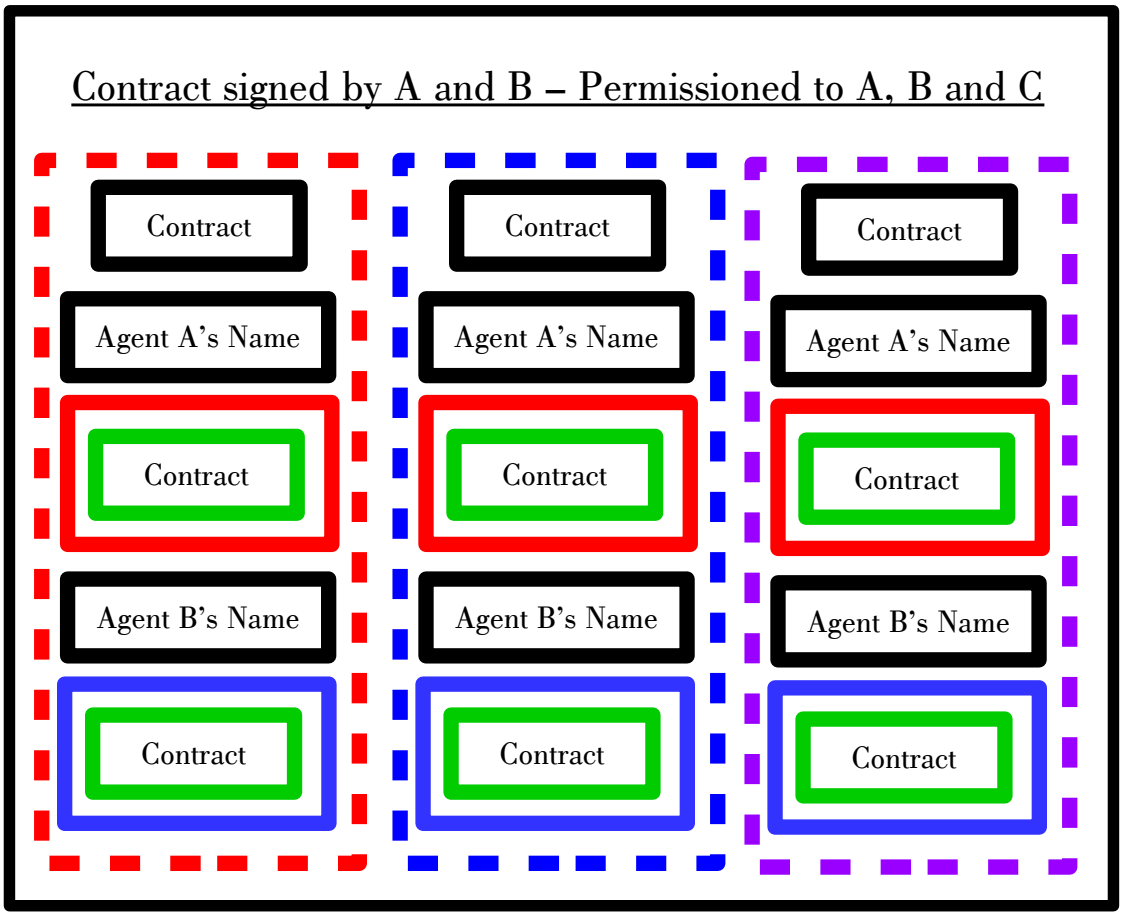
4. It may or may not matter if the data in these different encrypted pairs are the same. Each permissioned agent has access to cleartext data that he can verify was signed by agent A. However, by adding a cleartext hash of the data to the record, two things become possible. First, each privileged agent can verify that the record pertains to exactly the data to which he has encrypted access. Second, each agent can provide cleartext of the data to any outside agent, a court, for example, and the outside agent would be able to verify that the document is the one referred to in the record.

   The resulting record would look like this:



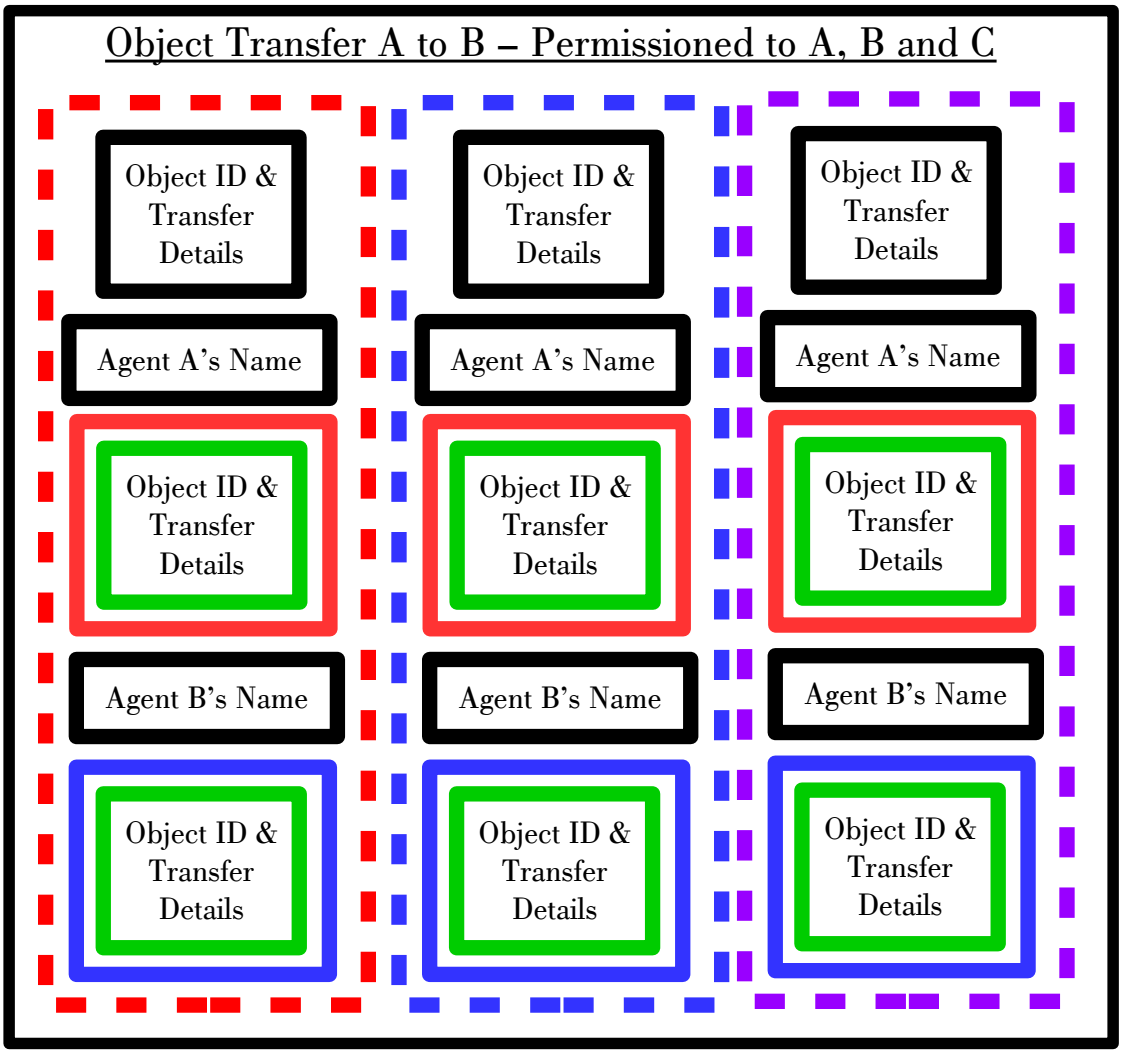
The same essential idea can be used to add contracts or agreements to an open blockchain while maintaining granular permissioning. The difference is that each of the parties to the document (agents A and B) sign it, and then encrypt the cleartext document along with all the signatures of all the parries using their public keys. Permissioned readers are added by encrypting this same information group with their public keys.

Such a record would look like this:



Permissioned object transfer records operate in the same way. The only difference is that the format of the record is strictly controlled. For example, a given object transfer blockchain might require that a record include a legal description of a property, the VIN of a car, bill of lading, or the ID number assigned when the transfer object was legally tokenized. Other details might include the names or public key addresses of the sender and receiver, the time and place of the hand-off, and whether taxes and customs have been paid. The parties are the agents on each side of the transfer, and both sign the document.

Such a record would look something like this:



Metadata, States, Search, and Privacy

An object transfer blockchain provides a list of transactions that can be integrated to find out the set of objects or the balance each address has to its credit. Although availability of the complete record is important because of the audit trail, the blockchain's **state** and is all that is needed to most purposes.

Document certification blockchains do not have states in a strict sense, but the records they contain may have certain structural relationships. For example, a document might contain inspection reports on equipment or the current flow rate measured by IoT sensors in a pipeline. Only the most recent document, or perhaps some sort of integration of related documents, is of interest in such cases. Medical, educational, maintenance, employment and other records form sets that evolve over time also have kind of state. A complete medical record for a patient, the set of prescriptions a writ-

ten by a specific doctor over the last year, all maintenance records for a given piece of equipment, and so on, need to be collected and used as a whole to provide their full value.

How do we generate states and collections from the raw blockchain? When object transfer blockchains are nonpermissioned like bitcoin, the current state can be found by inspecting the historical record and continuously updating as new blocks are written. In general, nonpermissioned chains have enough data in the records to allow the useful collections and integrations to be made. To the extent they do not, additional metadata can be added.

For permissioned blockchains, direct collection and integration is difficult. Putting together a patient's medical record is only possible for someone who is permissioned for access, and who knows the exact location of all the relevant records in the chain. Losing a record's location is equivalent to losing the record.

The obvious solution is to add metadata to the records to facilitate search and retrieval. Metadata attached to a record of a medical checkup, for example, might include information regarding the patient, doctor, facility, insurance payer, diagnostic tests preformed, date of the visit, and so on. A logistic record might need metadata that identifies the shipment, all agents involved in delivering the shipment to its final destination, date, mode of transportation, and type of cargo. How much metadata is needed depends on the types of reports, collections, and abstractions that are required for the specific use case.

The obvious problem with adding cleartext metadata to permissioned records is that it creates a significant potential for a breach of confidentiality or privacy. The more metadata is included, the greater the risk becomes.[3] A different way of enabling search while protecting privacy must be found.

Smart Contracts

Smart contracts consist of open, verified code that operates within a blockchain. Since the code is visible, agents can be sure that it does exactly what it claims it does. Smart contracts can solve the search/privacy problem outlined above as follows.

The creator of a record generates a metadata file complete enough to allow all the searches needed over the records in the blockchain. The creator then encrypts the metadata with the public key attached to a smart contract. The record and the encrypted metadata are then sent to the smart contract instead of being directly submitted the blockchain's validator network. The smart contract, in turn, decrypts and reads the metadata attached to each record and generates a cleartext record number. The record and its number is then passed on the validator network for inclusion in the next block.

3 Any agent who is permissioned for a record can immediately correlate the metadata identifiers with their meaning. For example, a patient would know the real name of the doctor he just saw and his identifier will be written in the cleartext metadata on the otherwise permissioned record of the visit. This in turn can be used identify other records involving the same doctor. By observing the time that the record was written and seeing who was in the waiting room at the time, other patients can be identified as well.

The smart contract could store the metadata and associated record numbers it accumulates over time in an encrypted database for use as needed. Alternatively, the blockchain could be setup to permit a specific set of searches, or to generate and update a specific set of record groups. The smart contract would keep a list of record numbers that satisfy each one of the permitted searches or groupings, updated as each new block is added to the chain. The metadata would then be discarded. Security could be enhanced if the smart contract uses a new PPK pair to encrypt the metadata it receives for each block and then discards the private key after the these lists are updated. The metadata would then become permanently unreadable, and only the record number lists would remain.

Whatever approach is taken, users could request a list of record numbers satisfying a given search criterion from the smart contract. The requester would prove his identity with his own public key and the smart contract would deliver him a list of record numbers that satisfied his request *and for which he is permissioned*. Even though requesters would not able to decrypt records for which they are not permissioned, revealing the existence of such records might allow nonpermissioned agents to infer private information. In any event, having gotten the list of record numbers the requester would retrieve the records from the blockchain, decrypt them, and use them as he wishes.

More formally, the smart contract would work as follows:

Secure search smart contract

Registering metadata for a record:

1. Creator Record and ciphertext of metadata → Smart Contract

2. Smart contract Decrypts metadata, creates a record number, and either adds this to an encrypted database for future searches or runs a set of programmed searches that update a table containing lists of record numbers that meet the permitted search criteria.

3. Smart contract Adds the record number to the record received from the creator and passes the pair along to the validation network for inclusion in the next block.

Running a secure search:

1. Requester request for records satisfying a specified search criterion and requester's public key → Smart contract

2. Smart Contract file with list of record numbers satisfying the search criterion and which have the provided public key as part of the records metadata → Requester

Patients or customers might have a legal right to control who has access certain private records that pertain to them. Giving consumers fuller control over their data might be good public policy or

good for a business who wished to gain trust. For example, sovereign identity schema require that agents be able to allow or prevent others from seeing personal records such as passports, driver's licenses, educational credentials and even medical and commercial data. At the same time, agents need to be able to prove the authenticity of such records to those to which he grants access. Businesses could make "Opt-in" privacy policies more granular by allowing agents to authorize access to specific information by specific agents at a specific time instead of asking agents to give blanket access to their data in advance. Customers might be willing to share more of their data as a result

In some cases, broad opt-in privacy policies make sense. For example, insurers have a legitimate need to see medical records for treatments they are asked to pay for, and ticketing agents, reservation systems, flight attendants, homeland security, etc. all need on-demand access to customer data to get a passenger to his destination.

Even if broad opt-in data access is desirable, it might still be good policy or good business to require that agents be informed when their data is accessed, and by whom. This would discourage abuse of private data and also allow agents to verify that privacy policies are being followed.

Assume for the moment all permissioned agents agree to access the records in the blockchain exclusively through a smart contract. Government agencies might be constrained to do so by regulation or policy, and companies might be bound by their privacy agreements with customers, for example. The contract could be configured to require consent from some, all, or none of the permissioned agents to allow record access. Similarly, the contract could notify some, all, or none of the agents when granting access.

Doing this while preserving privacy depends on the availability of a secure communications layer. One approach is to have a smart contract post requests for access or records of access granted to a webpage for which the controlling party has the password. This kind of passive notification is well suited to notifying agents that their records have been accessed, but unless agents check the page regularly, long delays for access authorization would occur.

Actively pushing notices would require that agents register a connection between a public key and an email account or mobile device with a smart contract. Note that this public key need not be part of an SSL certificate that identifies the party. The smart contract would then send an email or text with a link encrypted using this public key, The user would decrypt it with his private key, click the link, and complete his registration. Changing phone numbers, getting new anonymous email addresses, or changing passwords would work the same way.

This smart contact would then operate as a generalized communications intermediary. Other smart contacts or real world users would feed the communications smart contract a public keys/ message pair. If the key is registered, the contract would the recipient an encrypted link to a webpage where the encrypted message could be accessed. Routing communications through a single hub also has the advantage of obscuring their origins in the event that messages are somehow intercepted.

Communications hub smart contract:

Registration:

1. Blockchain party public key, email address or phone number → Smart contract

2. Smart contract one-time use URL for registration confirmation encrypted with the provided public key → email address or phone number provided.

3. Blockchain party decrypts the link, navigates to the webpage, and completes registration.

Secure communication:

1. Requester public key of party, message text (cleartext or ciphertext) → Smart contract

2. Smart Contract one-time use URL where message is posted encrypted with the provided public key → email or phone number associated with the public key.

3. Notified party decrypts the link, navigates to the webpage, reads it, and clicks yes or no to allow access to the record in question if this is required.

The communications hub makes it possible request permission to access or share records. Can a smart contract go further and actually mediate access to permissioned records in a verifiable way? The answer seems to be: to some extent. Such an access control/notification contract might look like the following:

All records would be written using the access control/notification contract. Agent A, agent B and any other parties to the record would all send each other signed versions of the document. Each would verify the hashes to confirm that the same contact was signed by all the parties. Each party would then create a data object consisting of the cleartext of the document, and the signatures of all the parties to the document including his own. Each party would create a hash of this data object and a copy the data object encrypted with his own public key. These would all be sent to to the access/notification contact. Once the hashed and encrypted data objects from all parties arrive, the smart contact would verify that the hashes of the data objects sent by all the parties were the same. The smart contract would then encrypt each of the already encrypted data objects with its own public key and also encrypt a list of the public keys of all the parties to the document. Finally, the record containing each of the doubly encrypted data objects, the hash of the data object, and the encrypted public keys of the parties would be sent to the validator network for inclusion in the chain.

To access a record, a permissioned party would send a request to the smart contract containing his public key. The smart contract would verify that this key belonged to a permissioned agent by decrypting the set the of public keys included in the record. If the key was found, the smart contract would do one of two things:

1. If the contract imposes access control, a request would be sent the communications hub contract to obtain permission from which ever parties were necessary. If the correct parties respond to the messages signifying agreement, the access control contract decrypts the permissioned agent's version of the data object and sends it to the agent. The data object that goes out is still encrypted with the requester's public key, but the smart contact's encryption is removed.

2. If the contract imposes notification rules, a request would be sent the communications hub contract to notify the required agents. The access control contract would then decrypt the permissioned agent's version of the data object and send it to the agent.

Adding permissioned readers could be done according to a number of rules. The parties could create a list of the public keys of agents they wished to make permissioned readers. The smart contact writing the record might take the union or the intersection of these lists, choose the set of permissioned readers requested by a majority of the parties, and so on. Mandatory permissioned readers such as the IRS could be coded directly into the smart contacts. The smart contact would ask some (or one, or all) of the parties to send copies of the data object encrypted with the public keys of the set of permissioned readers. These encrypted data objects would then be encrypted with the smart contract's public key and added to the record before it is sent to the validator network.

Note the following: First, the smart contract never sees a cleartext of the document, and a cleartext is never transmitted over the network. Second, even permissioned agents cannot access the information contained in the open blockchain record without going through the smart contract. Third, The blockchain still allows independent verification that the record is authentic. Once a permissioned agent receives and decrypts the data object, he can take a hash and compare it to the time-stamped hash in the blockchain. He can also provide the cleartext object to any other agent who can make a similar verification.

Unfortunately, this access/notification control contract only works if parties and permissioned agents can credibly bind themselves to accessing records through the contract alone. All the parties to contract could have retained copies of the original cleartext of the document and signatures of the other parties. Thus, any party could provide the data object to third parties directly who could then verify it by checking its hash against the one in the blockchain. There is no way to detect of prevent this. Once a document is seen by any party, it cannot be unseen.

Given this, a less rigorous version of the contract might be preferred. The access/notification smart contract could use its public key to encrypt only the copies of the data object that were intended for permissioned readers' public keys. The copies encrypted with the parties' public keys would be written into the record directly. While the any of the parties could still send readers or anyone else copies of the document, readers themselves could not access the document independently without going through the smart contract. Of course, once a permissioned reader is granted access for the first time, has his own unencrypted copy of the data object and is in the same position as the parties to contract.

Access/notification smart contract:

Record creation:

1. Document parties signature of the document → All other document parties

2. Document parties data object encrypted with their public keys, hash of data object, public keys of all parties, public keys of the desired permissioned readers → Smart contract

3. Smart contract request to encrypt data object with public keys of approved permissioned readers → All document parties

4. Document parties data object encrypted with public keys of approved permissioned readers → Smart contract

5. Smart contract validates the record creation request (e.g. Do all the hashes of the data objects agree, do all the data objects encrypted for the permission agree).

6. Smart contact creates a record by encrypting some or all of the encrypted data objects with its own public key, encrypting a list of the permissioned agents' public keys, and putting this together with a hash of the data object.

7. Smart contact record → Validator network

Record access:

1. Permissioned agent public key, request for access to a specific record → Smart contract

2. Smart contact checks that the public key in in the list of permissioned agents that is encrypted in the record.

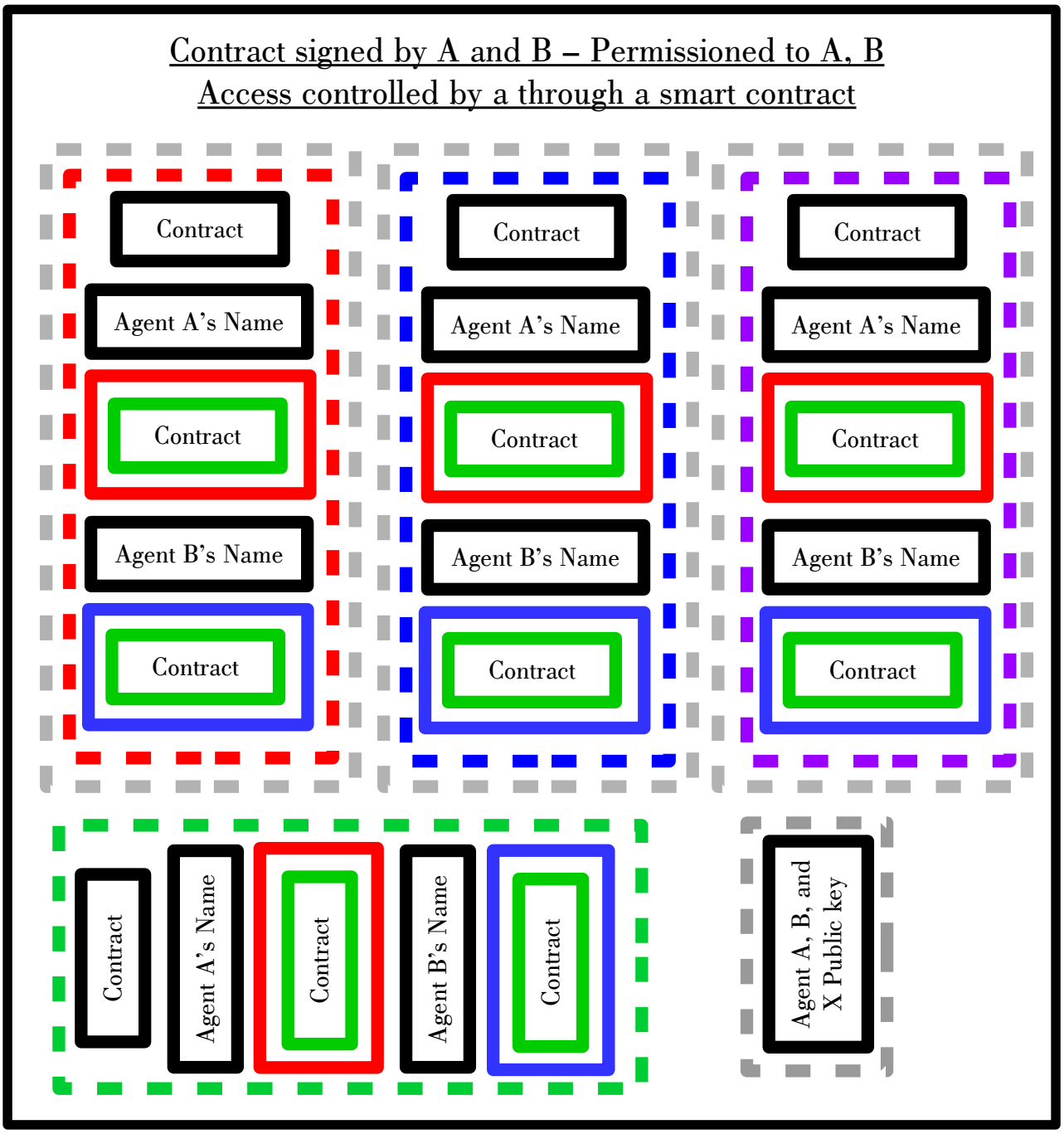
3. Smart contract messages to communications hub contract → Communications hub

   a. If permission from some or all parties is required, the messages requesting permission go to the parties that are needed. If all signal their agreement, by clicking a link that sends a message back to the access/notification contract, proceed to step 4.

   b. If notifications to some or all parties are required, then the messages go the needed agents.

4. Smart Contract decrypted copy of data object (still encrypted with the requesting party's public key) → Requesting permissioned agent

The record itself would look like this:



In general, smart contacts are where any business logic needed for a blockchain should reside. Business logic contracts could be combined with the types of contracts outlined above to gain greater security.

Compliance and Legal Foundations

Provided the courts accept the authenticity of the digital signatures, a contract stored in a blockchain is just as valid as one written on paper. The legal foundation to enforce contracts exists and directly transfers to contracts in blockchains.

The legal meaning of other types of records kept in a blockchain is less clear. Records documenting chains of custody and logistic hand-offs might be viewed as equivalent to receipts and inherit their existing legal standing, but this is not yet well-established in all cases. For example, there are reporting structures already in place for registering the transfer of controlled pharmaceuticals. Acknowledging receipt in a blockchain does not satisfy these requirements at present.

This becomes especially problematic when transfers require the tokenization of digital or real assets. As the law stands, deeds, titles and land transfers must be registered with government agencies. A bill of sale or a receipt is not enough to legally prove transfer of ownership. Of course, a county or state could decide to move all of its land ownership records to a blockchain and declare that blockchain validated transfers are legally binding. Until this happens, tokenization is of limited value

In the financial sector, on the other hand, tokenization is a long established practice. A stock certificate represents a claim over some faction of distributed profits and certain voting rights in a company. It is not a title certifying the ownership of any real asset or good. Bonds, options, futures, and so on, are similar. Physical stock certificates are seldom issued today. Instead, companies keep a digital register of stockholders and the number of shares they own. Brokerage houses maintain accounts for clients, but send notices of purchases and sales of stock to the originating company. Even though a share of stock is already a digital token that is ideally suited for exchange on a blockchain, two things are required before this can happen. First, publicly traded companies must agree that the blockchain is the definitive record of ownership. Second, the Securities and Exchange Commission (SEC) would have to approve this change and create standards that blockchains would have to meet to constitute a legally valid record of ownership.

Fiat currency is mostly represented by electronic balances in bank accounts and other financial institutes today. Nevertheless, tokenizing dollars and trading them on blockchains is especially difficult. In addition the lack of an enabling legal framework, it is actually illegal to do so unless strict regulations are followed. The United States and other governments are justly concerned about terrorism, financial crime, and tax evasion. US banks are required to comply with Know Your Customer (KYC) and Anti-Money Laundering (AML) laws largely as a result of the USA Patriot Act of 2001. Similar requirements have been since enacted in the majority of developed countries. Failing to comply can result in large fines.[4]

Of course, cryptocurrencies such as bitcoin make enforcement of these types of regulations difficult. Given the decentralized nature of the blockchain, there is no agent who can be forced verify the identities of users.[5] As long as transactions stay on the blockchain, everything is fine. However, moving money into, and especially out of, of bitcoin is more problematic. The easiest and lowest

4 For example, the US fined BNP Paribas $8.9B in 2014 for violations of AML regulations.

cost ways of converting bitcoin to fiat currency involve banks or other real world financial institutions that must comply with KYC and AML rules.[6] Thus, to the degree that agents decide that cryptocurrencies satisfy financial obligations, there is no need to deal with fiat currency, whether tokenized or off-chain. At present, cryptocurrencies have no legal standing and so security depends on trust in the platform or confidence in its smart contracts.

Key Security

The biggest attack surface for blockchains built on the architecture outlined above is private key security. As long a user has exclusive control over his private keys, all is well. If a user forgets or loses his ability to access a private key, any record that was encrypted with the corresponding public key becomes unavailable. Medical and financial records would be lost just as if paper records had been burned.

Smart contracts on blockchains could help ameliorate this to some degree. Most encrypted records are permissioned to at least two parties. If an agent could establish his identity to a recovery smart contract (or perhaps a trusted human who could credibly attest to the agent's identity), the contract could agree to that a new public key represented the user. It could then request that other agents who are privileged for each record decrypt them and then create new records using the new public key. These recovered records would retain the original time-stamp as well as the time-stamp of the new block into which they are written. In effect, this is like requesting copies of lost records from other agents who have them filed away.

If a key is stolen, on the other hands, things are much worse. First, until the PPK is revoked, the thief can sign agreements, affect transfers of tokenized assets, and act in all ways with the full authority of the key's owners. This is like a super-charged identity theft, and given that the actions of the thief are written in immutable, irrevocable, blockchain records, recovery is very difficult. Second, the key thief would have access to all the records encrypted with the corresponding public key and would be able to prove their authenticity. Revoking the key would prevent new records from being exposed to the thief, but records already written in the blockchain would be permanently compromised.

This is a general problem for many new technologies. PPKs, especially when validated by an SSL certificate, are the gold standard for digital identification. Biometrics, smart cards, and more complicated multi-factor authentication schema can help, especially with access control when a user is physically present. Unfortunately, they cannot be used to create independently verifiable digital signatures when the user is not present.

---

5 It should be mentioned that the structure of bitcoin's protocols limit privacy and anonymity. The nature of it public blockchain makes it possible to infer about users by looking at record of transactions. The Ethereum protocol does a much better job of protecting user's privacy and anonymity.

6 There are more elaborate ways such as selling on eBay, meeting someone in person using Craig's list, going to local bitcoin group meetings to buy and sell, having an online merchant who takes bitcoin send you merchandise that you can resell, etc. All of these have high transactions costs and some are risky.

Until there are better approaches to key security, enterprise and other sophisticated agents who are capable of key control may be the main beneficiaries of blockchain. The security risk for individuals and other less sophisticated users may simply be too great to justify putting sensitive information like medical and personal financial records in privileged open blockchains

Validation and Immutability

Blockchains are append-only ledgers validated through distributed consensus. Ideally they should be set up so they cannot be censured, do not depend on trusted agents, and allow all participants to be anonymous. The credibility of a blockchain as a record depends on confidence that new blocks appended to the chain contain only accurate, verified information, and that the data already recorded in the blockchain is immutable.

If we let a single agent (maybe a bank or company like Microsoft) create and append each block, we would have to trust in the honesty of that agent. What stops the agent from filling the block with transactions transferring money to its own account instead of following users' instructions? What prevents it from rewriting records to erase any evidence that certain uses ever had money in our account in the first place? Even if the agent is trustworthy, how do we know that some government will not force it to reveal users' identities, or tax all transactions at 10%?

The solution is to establish a credible distributed consensus over the contents of the each block before it is appended to the chain. These consensus builders must have some reason to behave honestly. The two major approaches to solving this problem are called **Proof of Work** and **Proof of Stake**, and both have limitations.[7]

Proof of work is resource intensive, but is trustless, anonymous, and open. Public blockchains built in this way can be widely distributed and are essentially impossible for governments or any other actor to censor, influence, or alter.

Proof of stake is cheap, but requires trust, is not anonymous, and is closed. Private blockchains built this way are generally only for the use of the stakeholders and are not publicly exposed or distributed outside this circle. However, since the stakeholders are known, they can be subject to pressure and influence from governments and others.

Establishing the credibility of a blockchain is fundamental to its value. If it is possible to alter records in a blockchain, then they are no more trustworthy than records stored in an ordinary database. Unless the blockchain is immutable, it does not provide a credible audit trail, a guarantee that a document was created at a specific time, proof that a record or document has not been altered since it was written, or a clear foundation to settle disputed transactions, agreements, or ownership. This means that even blockchains that created by a single agent to publish open records need validation. The same is true for blockchains intended for use within a single firms or by small groups of mutually trusting agents.

7 However, see John Conley (2017) "Dominant Strategy Coalition-proof Blockchain Validation without Proof of Work or Stake," manuscript, for a third approach that combines the open and trustless nature of proof of work with the low cost of proof of stake.

The immutability of blockchains also brings with it some responsibilities. If a blockchain was set up to publish unencrypted information at users' request without restriction, for example, child pornography, libelous statements, classified information, trade secrets, or copyrighted material might find its way in. The only way to remove undesirable records would be to destroy the entire blockchain and start over. Even this might not succeed if the copies of the blockchain are widely distributed.

This means that what gets into a blockchain must be carefully policed. Object transfer blockchains make this easy because the format of records is fixed and does not permit the inclusion of undesirable material. Blockchains recording public records coming from a single source such as an agency or university are safe unless the submission process is compromised or abused. The consequences of a disgruntled employee slipping a naughty picture or a racist rant into an institutional research depository or vehicle registration blockchain would be dire. Measures to make sure this does not happen are extremely important. Permissioned blockchains are not usually not intended to publish any unencrypted content, and as long as it is verified that no record has uncontrolled cleartext, this concern is minimal

Conclusion

This paper has argued that all implementations of blockchain can be classified using a small number of fundamentals. Specifically:

- Object transfer/document storage
- Open chain/closed chain
- Anonymous/nonanonymous
- Permissioned/nonpermissioned
- Proof of work/proof of stake (or other).

The difference between object transfer and document storage is whether the mining or validation network is required to verify that the new records to be written to a block collectively satisfy some type of business logic, or if business logic is handled exclusively by users and smart contracts. The choice between the two is driven by the use case.

There is almost never a good reason to keep records on a closed blockchain. The point of blockchain is to provide verifiable security without trust. The integrity and behavior of an active directory serving a gateway to open data cannot be independently verified.

The anonymity of parties and other agents participating in a blockchain is a simply a mater of choosing what goes into records.

Permissioning records in a blockchain as opposed to making their contents publicly readable is a non-trivial distinction. Permissioning can be implemented in many ways and requires the use of smart contracts and alterations is record formats.

At one level, how records in a blockchain are validated is not an important decision. Validation is validation, and if the method is trustworthy, it does not matter to the user how it was accomplished. At another level, this choice has a significant impact on the cost of running a blockchain, and different approaches bring their own set of weaknesses and attach surfaces with them.

Putting this together, the open-anonymous-permissioned type of blockchain is best adapted to the great majority of use cases. This means that a permissioning architecture that can be implemented on a write-only, open ledger is essential. This point of this paper is to suggest how this might be done with a minimum of overhead and in a way that does not require the use of any trusted agent.

Appendix: Examples of Blockchain Types and Access Rules

Open – Anonymous – Permissioned
Digital dead–drops
Tokenized object transfers
Stock, bond, and security allocation
Identity
Educational and professional certification and licensing

Open – Anonymous – Nonpermissioned
Bitcoin and other cryptocurrencies
Comments complaints, polls, and voting
Review and opinion sites
Public comment on permits, legislation, and regulation

Open – Nonanonymous – Nonpermissioned
Court records
FOIA eligible records
Required compliance fillings
Copyrights
Land title ownership
Vehicle registration
Food provenance for safety
Government contracts, bidding
Public hearing notices
Vehicle registration

Open – Nonanonymous – Permissioned
?

Open – Mixed Anonymity and Permission
Real time auction and ad placements
Betting and prediction markets
P2P selling, Craigslist, Ebay
B2B markets
E–commence

Closed – Anonymous – Permissioned
Transactions:
Bank settlements
Consumer bank and credit accounts
Merchant credit card settlement
Prepaid debit cards
Micropayments
Virtual wallets
Device to device payments
Government payments to individuals

Tokenized object transfer:
Prescriptions
Multiparty logistic and shipping chains
Chain of custody
Digital asset records

Multi-party business process:
Claims processing
Real estate sales process

Permissioned access by many parties:
Military service records
Medical records
Student records

Sales of digital or tokenized objects:
Tickets and reservations
Car leasing and sales
Licensing of content and software

Other:
Royally tracking and payments
Digital rewards
Audit trails for financial compliance
KYC/AML
IoT Device directories
Grid monitoring Operations (e.g. water flow)

Closed – Anonymous – Nonpermissioned
Whistle blower
suggestion box

Closed – Nonanonymous – Permissioned
?

Closed – Nonanonymous – Nonpermissioned
Internal Inventory control
Shared internal data and processes

Several things standout here.

1. It seems natural to keep bank accounts, medical records, and audit trails for private companies on private servers. Most such records, however, are anonymous and permissioned. Verifiable access control must be implemented in any event. Keeping such records on a closed blockchain does nothing to add to this security. Thus, using an open blockchain instead provides the increased trust that comes with public validation, saves the costs of maintaining and updating active directories on private servers, is is just as good from a security standpoint.

2. Only if records were nonpermissioned or nonanonymous would it ever make sense to keep them on closed blockchains. The potential benefit of using a closed chain is that it might allow all the agents in a trusted group easier access to the data the chain contains. Use cases, however, are fairly limited. Even the for internal business processes and inventory control, for example it is unlikely that a company would want or need to allow every employee to have access to all the data in the chain.

3. This means open-anonymous-permissioned blockchains are what the vast majority of use cases require.

4. Open-anonymous-nonpermissioned blockchains do have one killer app: bitcoin and certain other object transfers where confidence in the process requires that the public be able to check to see if the accounting and transfer rules are being followed.

5. Open-mixed anonymity-mixed permissioned blockchains also have one killer app: decentralized markets and commerce. Anonymity is mixed because sellers may wish to establish real-world reputations or may be required by tax regulations or other law to publicly identify themselves. Buyers are more likely to wish to be anonymous. After all, there is no reason for the public to know what I buy on eBay or Amazon. Permissioning is mixed because seller data (prices and descriptions) regarding items offered for sale, bids and asks, the current odds and outcomes in prediction markets, etc., need to be nonpermissioned and publicly visible for the markets to work. On the other hand, information on completed sales or final prices may be permissioned and private.

6. Open, nonanonymous, nonpermissioned blockchains also have one important use case: creating public records.

Summary

Public Records	Open-nonanonymous-nonpermissioned
Markets	Open-mixed anonymity-mixed permissioned
Bitcoin and open object transfer	Open-anonymous-nonpermissioned
Everything else (almost)	Open-anonymous-permissioned