Survey of Confidentiality and Privacy Preserving Technologies for Blockchains

Danny Yang, Jack Gavigan, Zooko Wilcox-O'Hearn, R3 Research

November 14, 2016

1 Introduction

Blockchain (or distributed ledger) technology has the potential to drive innovation across a range of industries. Like any emerging technology, however, it needs to mature before its full potential can be realised. As this technology continues to develop, the concepts of confidentiality and privacy have emerged as key areas of research.

In a recent survey by Greenwich Associates,[1] 56% of the 134 market participants surveyed cited transaction confidentiality as a major security concern. This concern is driven by several factors. In many cases, confidentiality and privacy are enforced by legislation (e.g. EU data protection legislation), regulation (client confidentiality) or contract (commercial confidentiality). For financial use cases, it is often necessary that users keep details of their transactions private. Individual end-users typically do not want the details of their income and spending to be exposed. Similarly, small businesses do not want to reveal details of their revenues and cash flows to competitors. For R3's user base, capital markets participants do not want their competitors to be able to see the details of their transactions. Maintaining the confidentiality of central bank money transfers and settlement activity is desirable to avoid the disclosure of information that could damage confidence, or, what the Federal Reserve refers to as "discount window stigma."[2]

In the early days of the Web, the Secure HyperText Transfer (SHTTP) and Secure Sockets Layer (SSL) protocols emerged as competing standards for adding privacy to the Web, with SSL eventually triumphing. We expect blockchain technology to follow a similar path, with multiple competing privacy standards emerging over the next few years.

In his review of the Ethereum platform[3], Vitalik Buterin provided a brief overview of some of the technologies that can be leveraged to add privacy to blockchains. In this report, we take a more detailed look at confidentiality and privacy, reviewing and examining the technologies and protocols that are currently in use or emerging as potential solutions for adding confidentiality to blockchains. We assess the advantages, disadvantages and implications in terms of adoption for financial use cases. We focus primarily on the underlying technologies, though we allude to specific implementations where appropriate.

Though we often use the terms "confidentiality" and "privacy" interchangeably in our daily lives, they are distinct terms from a legal standpoint. Throughout this report, we use the term *confidentiality* in the context of protecting *data* (e.g. transaction details, price, asset types, account and wallet balances, the business logic of smart contracts) from unauthorised third parties. We use the term *privacy* to refer to protection from intrusion into the identity of blockchain participants and parties to transactions.

To assess different confidentiality- and privacy-enabling technologies, we consider how they impact the key features that make blockchain technology attractive for financial use cases:

- Eliminates the need for a trusted third party (TTP) or central authority for record- keeping, which can reduce transaction costs
- Supports the issuance and transaction of digital bearer assets, which enables "on- chain" securities trading, novation, offsetting, etc.
- Creates a single shared view of the world (i.e., a single "source of truth"), which supports more efficient settlement and reconciliation by eliminating trade breaks, and makes it easier to share information with third parties such as regulators, trade reporting services, and central counterparties.

---

[1] https://www.greenwich.com/fixed-income-fx-cmds/securing-blockchain

[2] O Armantier, E Ghysels, A Sarkar, J Shrader (2011). "Discount Window Stigma during the 2007-2008 Financial Crisis."

[3] https://r3cev.com/blog/2016/6/2/ethereum-platform-review

This is a dynamic space and new technologies are constantly being developed, often privately within companies (e.g. R3's[4]). Therefore, this report represents a snapshot of the technologies that have been open sourced and made public at this point in time.

In this report, we focus on the key technologies that show the most promise and that are gaining the most traction in the space. We also outline how those technologies are being implemented in practise, with reference to the foremost implementation. We refer to Bitcoin and Ethereum frequently, for ease of analogy and because much of the public experimentation, testing, and innovation being carried out in this space are based on these two cryptocurrencies, though there are many other cryptocurrencies, some of which focus on privacy (e.g. Monero, Dash, Shadowcash, Anoncoin).

1.1 Confidentiality and Privacy versus Security

Information security practitioners often refer to the CIA triad: confidentiality, integrity and availability. While blockchain technology includes features that support both integrity and availability, achieving a similar level of support for confidentiality has proven to be more challenging.

Confidentiality & Privacy

In the context of blockchain technology, confidentiality and privacy means that both the data written to the blockchain and the identities of the parties involved are protected. For our purposes, it necessitates the following:

- The counterparties to a transaction cannot be identified by an unauthorised third party from the information that is written to the blockchain (including metadata), unless one of the counterparties has chosen to reveal that information
- Transaction details are not visible to unauthorised third parties and the world at large unless one of the counterparties has chosen to reveal that information
- Transaction details cannot be collated, analysed or matched with "off-blockchain" metadata to reveal any information about counterparties or transaction details. By this, our definition encompasses the use of graph analysis, pattern matching and machine learning to construct a profile of a counterparty based on the activities associated in the ledger

Integrity

Integrity in a blockchain system is about ensuring that the data that is written to the blockchain is correct and cannot be subsequently altered:

- Transactions cannot be created and added to the blockchain by unauthorised parties (this ensures that transactions cannot be faked and assets cannot be stolen)
- Transactions cannot be cancelled or reversed by unauthorised parties
- Transactions, once committed, cannot be later denied (non-repudiation)
- The record of transactions is immutable[5]
- Robust consensus or resistance to manipulation by bad actors

We deliberately allude to *unauthorised* third parties because some blockchain protocols may allow or mandate that counterparties' identities and/or transaction details be disclosed to specific third parties.

[4] http://r3cev.com/s/corda-introductory-whitepaper-final.pdf

[5] By this, we mean "extraordinarily difficult to tamper with or amend."

Availability

Availability of blockchain systems refers to their ability to withstand outages and attacks:

- Capacity to handle high loads and volumes while remaining functional and responsive
- Absence of a single point of failure (this is inherent in blockchain technologies' distributed nature)
- Ability to resist denial-of-service attacks

Security often requires tradeoffs. For example, requiring multiple signatures for a transaction is one way to improve security by making it harder for an unauthorised party to carry out a transaction. However, multiple signatures also necessarily involves multiple entities, resulting in a potential loss of privacy and confidentiality, and additional points of failure. There are third party services that specialize in adding security by being an extra signer to transactions, but the third party gains access to the transactions they are required to sign. Finally, enhanced privacy and confidentiality often necessitates greater difficulty and cost in terms of computational resource requirements.

1.2 Underlying Technologies

When assessing the benefits and drawbacks of various confidentiality- and privacy- preserving technologies, it helps to have a basic understanding of the underlying technologies and techniques. We therefore present a high-level overview of a selection of the relevant technologies. This overview is not intended to be exhaustively accurate, but rather to provide enough context for those who do not have a background in cryptography or computer science to understand the concepts we cover later in the report.

Public Key Cryptography

Public key cryptography (PKC) is the foundation of blockchain technology and forms the basis of many of the privacy techniques surveyed in this report.

Public key cryptography is primarily used for two things:

1. Encryption and decryption of sensitive data
2. Digital signatures to prove a message's authenticity

Encryption and Decryption

Cryptography is used to encrypt messages in order to prevent unauthorised parties from snooping on them. If Bob wants to send Alice a message, for example, he encrypts the plaintext message in plain English using a "key". Bob can then send the ciphertext to Alice, secure in the knowledge that, even if somebody else manages to intercept the ciphertext, they won't be able to decrypt it.

When Alice receives the ciphertext from Bob, she decrypts it. Before the invention of PKC, Alice would need to use the same key to decrypt the ciphertext that Bob had used to encrypt it. In other words, Alice and Bob would need to exchange a secret key between themselves before they could exchange encrypted messages. This type of cryptography is often referred to as *symmetric encryption* because the key used to encrypt messages and the key used to decrypt them are identical. If someone else managed to get hold of the secret key, they would be able to decrypt and read all of Alice's and Bob's messages to one another.

With PKC, or *asymmetric encryption*, there is no need to exchange a secret key. Instead, Alice generates a key pair, consisting of a public key and a private key. The public key is used to encrypt messages but it is not capable of decrypting messages; only the private key can decrypt a message encrypted with the corresponding public key.

Alice sends the public key to Bob and keeps the private key secret. Meanwhile, Bob generates his own key pair, and sends his public key to Alice. To send Alice a message, Bob encrypts it using Alice's

public key, and sends her the ciphertext. Alice then uses her private key to decrypt the message. Alice can then encrypt her reply using Bob's public key.

By eliminating the need to exchange a secret key, Alice and Bob have made an attacker's job far more difficult. To read all of Alice's and Bob's communication, the attacker must now steal *both* their private keys.

RSA

RSA was the first public key cryptosystem invented in 1977 by Ron Rivest, Adel Shamir and Leonard Adleman.[6] RSA, as it become known, is still widely used today although it is though it is gradually being supplanted by more efficient cryptosystems. For example, both Bitcoin and Ethereum use elliptic curve cryptography.

Despite its declining popularity, RSA is useful for explaining many of the concepts that underpin the privacy techniques that we survey in this report. Therefore, we present an overview of RSA for non-mathematicians in the Appendix.

Digital Signatures

In addition to encrypting messages, PKC can also be used to authenticate a message by creating a digital signature using the sender's private key. The recipient of the message can then verify the digital signature using the sender's public key.

Digital signatures help protect against fake or spoofed messages, as a third party with no access to the private key cannot forge a valid digital signature. It also ensures that the sender cannot later pretend that he did not send the message, because a valid digital signature can only be generated for a given message if the signer is using the correct private key (this is generally referred to as "non-repudiation"). See the Appendix for an example of how an RSA signature is created.

Digital signatures enable the transfer of value in blockchain-based cryptocurrencies such as Bitcoin. Control of a Bitcoin (i.e., the ability to prove possession and to send the bitcoin to another address) is associated with a public key. A transaction to transfer ownership of that bitcoin is only valid if the transaction is digitally signed with the correct private key. If the digital signature is valid, the transaction is accepted and recorded on the blockchain, and the bitcoin is now regarded as being "owned" by the recipient's public key, which is specified in the transaction.

Hash Functions

One way of creating a small digital signature is to use a hash function, a mathematical function that maps data of any size to a piece of data of a specific length, called the *hash value* or simply the *hash*. The hash acts as the digital fingerprint of the original data.

Cryptographic hash functions are one-way functions, meaning that they can't be reversed. In other words, it's impossible to find out what the original data is from a hash. A hash is also deterministic, which means the same input data will always produce the same hash. Blockchain technology leverages this quality to ensure the immutability of the transaction record.

An example of a popular hash function is SHA-256. SHA stands for "Secure Hashing Algorithm" and "256" indicates that the hashes it produces are 256 bits long. Below are examples of input data that are almost but not quite identical and their SHA-256 hashes, represented in hexadecimal.

---

[6] R.L. Rivest, A. Shamir, and L. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems."

Input data	SHA-256 hash
The quick brown fox jumps over the lazy dog	d7a8fbb307d7809469ca9abcb0082e4f8d5651e46d3cdb762d02d0bf37c9e592
The quick fox jumps over the lazy brown dog	109d51daea4988dbbcf10113bd7de272d5df5af1739844f4e3a0fb0f4b4567db
The quick fox jumps over the lazy brown dog.	90894b449198193133b3acd96561d61d677e48fe760071e0277ea70b900bf5c1
No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honour and reputation. Everyone has the right to the protection of the law against such interference or attacks.	57fda799521f01c9f1a2c320cd37dc1e2882790ba59729ee7357e5b236736871
No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honour or reputation. Everyone has the right to the protection of the law against such interference or attacks.	e4998f47c86fb13f4107729ae2a589b857c867f0b8093b562250316c8bef65d5

As you can see, any change (highlighted) in the input data results in a vastly different hash. You can also see that the length of the hash remains the same, no matter how big the input data is. You can hash images, hard drives, or entire data centres and the hashes will be the same length.

Digital signatures by themselves are proportional to the size of the data being signed. So in practice, it is the hash of the data or message that is digitally signed, rather than the data itself. This ensures that the size of the digital signature is not dependent on the size of the data being signed. The recipient can use the same hashing function to generate the hash of the message they have received and verify that the digital signature matches the hash they have calculated.

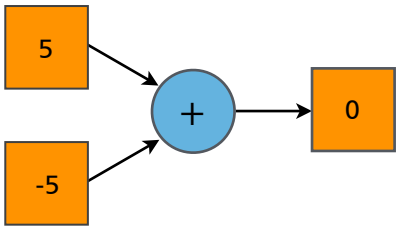
Although the limited size of the hash means that it is theoretically possible to find multiple data inputs that result in an identical hash (referred to as a collision), doing so is impractical. One would need to try many different inputs, with the chance of finding a matching hash being 1 in 2^{256} (or 1 in 115,792,089,237,316,195,423,570,985,008,687,907,853,269,984,665,640,564,039,457,584,007,913,129,639,936). Therefore, the message's recipient can be confident that the hash signed by the sender was generated from the original message.

Homomorphic Encryption

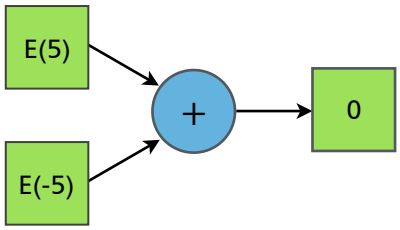
Homomorphic encryption allows arithmetic operations (e.g. addition, multiplication) to be carried out on encrypted values; when the result is decrypted, it yields the same result that would have been achieved had the same calculation been carried out on the unencrypted inputs.

> Plaintext1 encrypts to Ciphertext1
> Plaintext2 encrypts to Ciphertext2
> Ciphertext1 + Ciphertext2 = Ciphertext3
> Ciphertext3 decrypts to Plaintext3
> Plaintext1 + Plaintext2 = Plaintext3

Many cryptosystems are partially homomorphic. For example, the RSA cryptosystem is multiplicatively homomorphic. If you encrypt two numbers separately, using the same secret key, multiply the ciphertexts, then decrypt the result, you get the same result that you would get if you multiplied the two original numbers (see the appendix for a worked example.) While RSA does not support arithmetic functions other than multiplication (e.g. addition), other cryptosystems do. The development of fully-homomorphic encryption is the subject of ongoing academic research.



An arithmetic operation (addition)...



..with homomorphically encrypted data.

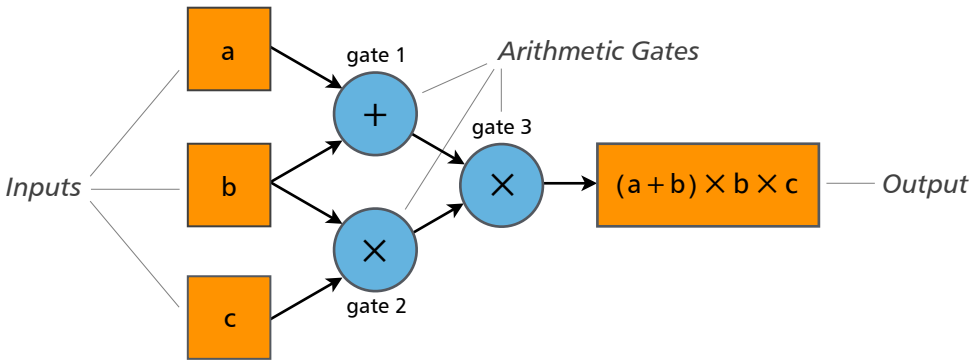
One practical application of homomorphic encryption is to allow untrusted third parties to carry out computation on encrypted data. For example, if one had two numbers and wanted somebody else to compute their product without revealing the two numbers, one could encrypt the two numbers using RSA encryption and provide the ciphertexts to the person doing the computation. She can multiply the two ciphertexts and provide us with the result, which we can then decrypt to get the product of the two original numbers.

In the appendix, we present an example which demonstrates the multiplicatively homomorphic qualities of the RSA cryptosystem. Other cryptosystems are additively homomorphic, which can be used to prove that a cryptocurrency transaction's outputs are equal to its inputs, without revealing either the inputs or the outputs (c.f. Pedersen commitments in section 2.8).

Arithmetic Circuits

When implemented in software, a single arithmetic operation is often referred to as an arithmetic gate. It typically takes two inputs and returns a single output.

Arithmetic circuits are constructed by connecting multiple arithmetic gates (each of which carries out a single arithmetic function, such as addition or multiplication) together, to carry out complex calculations.



A simple arithmetic circuit

When used in conjunction with homomorphic encryption, arithmetic circuits can be used to carry out complex computations on encrypted data.

1.3 Confidentiality and Privacy of Bitcoin and Ethereum Transactions

Both Bitcoin and Ethereum are public, open, transparent, pseudonymous blockchains. They are open in the sense that there are no restrictions on participation and transparent in the sense that all transaction details are visible on the blockchain. However, the counterparties are only identified by their public keys (commonly referred to as Bitcoin addresses or Ethereum accounts). There is no formal mechanism for identifying the person who controls a given public key.

	Bitcoin	**Ethereum**
Integrity	• Keys: ECDSA using the secp256k1 curve • Proof of Work: SHA-256 • Vulnerable to 51% attack	• Keys: ECDSA using the secp256k1 curve • Proof of Work: Ethash (based on Dagger-Hashimoto); move to proof of stake is planned • Vulnerable to 51% attack
Availability	• Both Bitcoin and Ethereum use a fully decentralized architecture, ensuring that there is no single point of failure.	
Privacy	• Pseudonymous • Can use Mixers/Tumblers to obfuscate transaction flows	
Confidentiality	• None - all transaction details are revealed	• None - all transaction details are revealed • Smart contract bytecode and execution is public

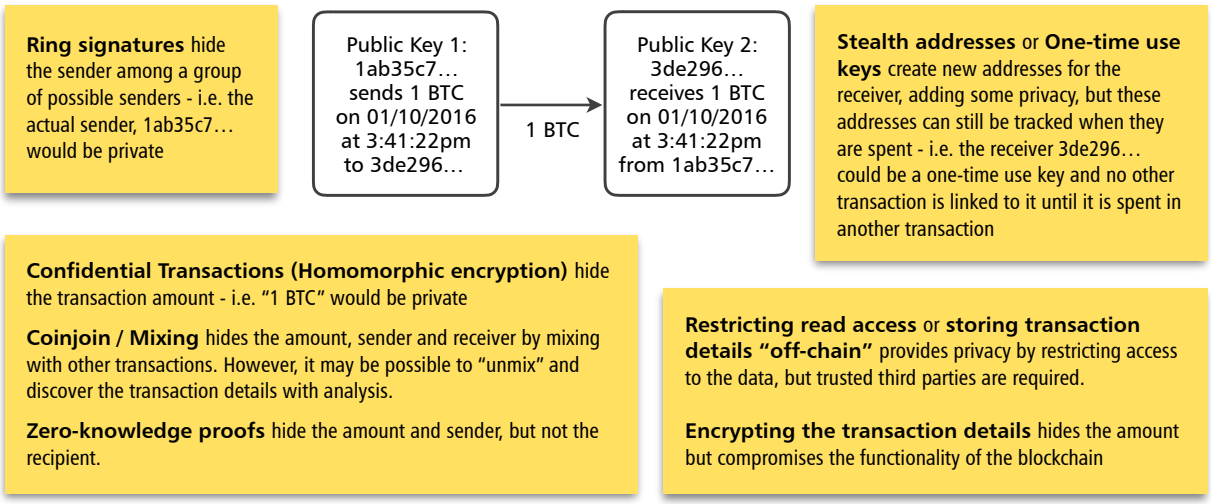
If a public key's owner (Alice) can be identified through information leaks, then it is trivially easy to monitor all associated transactions and all of Alice's funds by watching the blockchain for transactions to and from the public keys associated with Alice. The entities that Alice transacts with can then be further identified based on this association with Alice. Additionally, after a public key and associated transactions are identified, there is no way to "erase" this information as this is now public knowledge.

The flow of funds in a blockchain such as Bitcoin is public and the addresses of the senders and receivers can be linked across multiple transactions to try to deanonymize them and build profiles of their activity.

As previously noted, the privacy provided natively by Bitcoin and Ethereum is limited to pseudonymity, and transaction details are not confidential as the transaction amount and the assets being transferred, its metadata (including the time the transaction was executed), and its relationships to other transactions, are trivially available to anyone.

There has been a significant amount of research into how the Bitcoin blockchain can be analysed to identify the parties to a transaction, and what other transactions involve the same parties.[7] There has been less research on deanonymizing the transactions in Ethereum, but many of the same techniques can be used which include observing transactions in the network and correlating the source of the transactions based on their propagation across the network, or identifying patterns of senders and receivers.[8] Identifying these patterns is more straightforward in Ethereum, as public keys are more likely to be reused than in Bitcoin.[9]

The figure below illustrates how privacy can be added to the different aspects of a transaction in the blockchain using different techniques.



[7] a. E. Androulaki, G. Karame, M. Roeschlin, T. Scherer, S. Capkun, "Evaluating User Privacy in Bitcoin," in IACR: Cryptology ePrint Archive, 2012.

b. M. Fleder, M. Kester, S. Pillai, "Bitcoin Transaction Graph Analysis," in arXiv:1502.01657, 2015.

c. S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. Voelker, S. Savage, "A Fistful of Bitcoins: Characterizing Payments Among Men with No Names," In IMC '13: Proceedings of the 13th ACM SIGCOMM Conference on Internet Measurement, 2013.

d. M. Ober, S. Katzenbeisser, K. Hamacher, "Structure and Anonymity of the Bitcoin Transaction Graph," in Future Internet 5, 2013.

e. M. Ortega, "The Bitcoin Transaction Graph," Master's Thesis, Universitat Autonoma de Barcelona, 2013.

f. F. Reid, M. Harrigan, "An Analysis of Anonymity in the Bitcoin System," in IEEE International Conference on Privacy, Security, Risk, and Trust, 2011.

g. D. Ron, A. Shamir, "Quantitative Analysis of the Full Bitcoin Transaction Graph," in Proceedings of the 17th International Conference on Financial Cryptography and Data Security, 2013.

h. M. Spagnuolo, F. Maggi, S. Zanero, "BitIodine: Extracting Intelligence from the Bitcoin Network," in FC '14: Proceedings of the 18th International Conference on Financial Cryptography and Data Security, 2014.

[8] One early example of trying to identify users on the Ethereum blockchain is data analysis of The DAO hacker from Bok Consulting: https://www.bokconsulting.com.au/blog/the-dao-hackers-booty-is-on-the-move/

[9] There are open tools which can track and visualize Bitcoin address reuse: http://www.kristovatlas.com/rfc-tool-for-tracking-and-visualizing-bitcoin-address-reuse/

2 Comparison of key technologies and implementations

As the diagram above illustrates, different technologies have specific qualities in terms of the degree and nature of confidentiality and privacy they provide. The table below summarizes the most important techniques:

Technology	**Sender**	**Recipient**	**Amount**
Stealth Addresses	🔴	🟢	🔴
Pedersen Commitments	🔴	🔴	🟢
Ring Signatures	🟠	🔴	🔴
zk-SNARKs	🟢	🔴	🟢

Red: Provides no confidentiality/privacy
Amber: Provides limited confidentiality/privacy
Green: Provides strong confidentiality/privacy

In practise, some techniques can be used in concert to provide a greater degree of confidentiality and privacy than any single technique can provide.

Implementation	**Techniques used**
Confidential Transactions	Pedersen Commitments
CryptoNote	Ring Signatures Stealth Addresses
Monero with Ring CT	Ring Signatures Stealth Addresses Pedersen Commitments
Zcash	zk-SNARKs Stealth Addresses

The end result is illustrated below:

Implementation	**Sender**	**Recipient**	**Amount**
Confidential Transactions	🔴	🔴	🟢
CryptoNote	🟠	🟢	🔴
Monero with Ring CT	🟠	🟢	🟢
Zcash	🟢	🟢	🟢

Red: Provides no confidentiality/privacy
Amber: Provides limited confidentiality/privacy
Green: Provides strong confidentiality/privacy

2.1 Efficiency & Scalability Implications

Improved confidentiality and privacy comes at a cost, in the form of a larger transaction size (required to contain the cryptographic proofs required for the various techniques) or, in the case of zk-SNARKs, greater computational cost.

While transaction size varies considerably, depending on the number of inputs and outputs, the table below compares a typical transaction size for each of the key implementations. The median Bitcoin transaction size is included for comparison.

Technology	**Typical Transaction Size**
Bitcoin	300 bytes
Confidential Transactions	5,000 bytes[10]
Cryptonote	1,600 bytes
Monero with RingCT	13,000 bytes[11]
Zcash	2,000 bytes

Generating the zero-knowledge proof that must accompany a Zcash transaction involving shielded values is computationally intensive (currently benchmarked at 48 seconds on a typical desktop CPU). By contrast, the time required to create Confidential Transactions or RingCT transactions is measured in milliseconds, which may make them preferable for use cases that require low latency, and where transaction size isn't a constraining factor.

2.2 Restricting Read Access to the Blockchain

One way to make the blockchain private is to restrict who has read access to it. Instead of running a public blockchain in which there are no restrictions on participation, a private (or permissioned) blockchain is deployed.[12] In this instance, only authorised nodes are allowed to connect to the network and receive a copy of the blockchain. This allows both confidentiality and privacy to be maintained amongst the blockchain's participants. In other words, the information written to the blockchain is kept private from the world at large but the authorised participants can potentially see everything.

Restricting read access has several disadvantages. It is not suitable for use cases in which participants do not want other participants to be able to see details of their transactions. Furthermore, in order to manage access to the blockchain, one or more gatekeepers is required, which introduces a trusted third party (TTP): a source of both added cost and a potential single point of failure. Finally, any breach of the access control, or leak of the blockchain itself, will reveal all the transactions details stored in the blockchain. (The impact of this last risk can be mitigated by periodically resetting the blockchain, but this may not be suitable for all use cases.)

2.3 Storing Sensitive Information "Off-Chain"

An alternative to restricting read access to the blockchain is to store transaction details "off-chain" on another system with access control restrictions. The "off-chain" system is effectively a restricted read

[10] https://leastauthority.com/blog/zerocash_and_confidential_transactions.html

[11] http://monero.stackexchange.com/questions/471/how-much-larger-would-a-mixin-4-monero-transaction-be-compared-to-a-bitcoin-tran

[12] One approach in this environment that shows some merit as a technical solution is the use of software models for isolated, private execution to allocate private regions of memory, also known as enclaves, which are protected from other processes. An example would be Intel's SGX.

access system. The difference with the previous technique is that the blockchain itself can be public and nodes on the blockchain will not have any additional access. The transactions in the blockchain will contain a hash of the actual transaction details. No information about the actual transaction details can be gained from the hash, but the counterparties can confirm that their view of the transaction matches the other party's by verifying that the hash of the transaction details that they have recorded matches the hash that has been written to the blockchain. If the hashes do not match, this would indicate a mismatch in the records of the transaction details and the need to reconcile the records.

Storing information "off-chain" provides strong privacy of the transaction details. The off- chain system can be configured to restrict access to the transaction details to authorised parties only, supporting use cases where participants may wish to keep details of their bilateral transactions private from other blockchain participants.

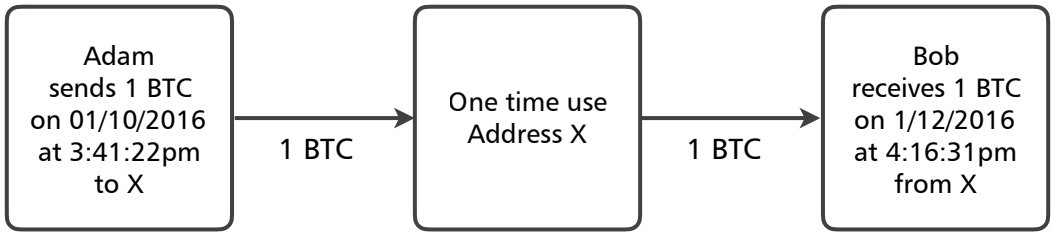
However, storing information "off-chain" negates many of the advantages of using a blockchain in the first place. Although the use of hashes can highlight breaks, without transaction details, the blockchain can no longer be a single, shared "source of truth." The issuance and trading of negotiable, fungible digital assets is no longer possible without reference to an off-chain position-keeping system.

Additionally, storing transaction information off-chain typically requires that both counterparties maintain their own record, or delegate that responsibility to a TTP, which brings with it the same costs and disadvantages as restricting read access to the blockchain.

2.4 One-time Use Payment Addresses

Re-using the same payment address makes it easy for an observer to watch the transactions being received and sent. An obvious solution is to avoid re-using addresses; this is an approach generally referred to as one-time use payment addresses. When using this approach, the sender generates a new address each time they wish to receive a transaction. Because the new address has no prior transaction history on the blockchain, it's more difficult for an observer to assemble a comprehensive view of transaction flows.

However, when the received funds are spent, information is leaked which can enable the use of graph analysis techniques to link addresses. A one-time use payment address is directly linked to both the deposit transaction and the withdrawal transaction. In each of these two transactions, additional addresses become associated with the one-time use payment address, and can be used to de-anonymize the one-time use payment address. The sender to the one-time use payment address and the receiver of funds paid from the one- time use payment address may not be anonymous. In the example below, address X is a one-time use payment address, but if we know that Adam sent to X and Bob received from X, then that can be used to infer that X is related to Adam and Bob. Similarly, assembling information on where the sender's funds originated, and where the recipient's funds were passed on to, can help de-anonymize the one-time use payment address. This technique is often called *transaction graph analysis.*[13]



[13] When a user later spends several one-time use payment addresses together ("merged") in the same transaction, that also leads to information leakage as those addresses can now be linked to the same person. Mike Hearn proposed "merge avoidance" as a way to lessen, but not totally avoid, these leaks: https://medium.com/@octskyward/merge-avoidance-7f95a386692f

Because information leakage regarding the identity of the receiver does not occur until they want to spend the received funds, one-time use payment addresses may be suitable for use cases where the counterparties only want to keep their involvement in a transaction private for a limited amount of time (i.e., transact using the one-time use payment address then consolidate into their main wallets after their identity is no longer sensitive information).

However, it should be noted that the privacy provided by one-time use payment addresses only extends to the counterparties to a transaction (principally the recipient). The transaction details are still published to the blockchain. Furthermore, the sender can tell when the recipient spends the funds, simply by watching the one-time use payment address.

2.5 Stealth Addresses

One-time use payment addresses can be unwieldy to manage. Each new address must be generated by the recipient and communicated to the sender. Imagine if you had to create a new email address each time you wanted to receive an email and let the sender know what address they should use.

Stealth addresses remove this requirement by allowing the sender to generate the new one- time use payment address.

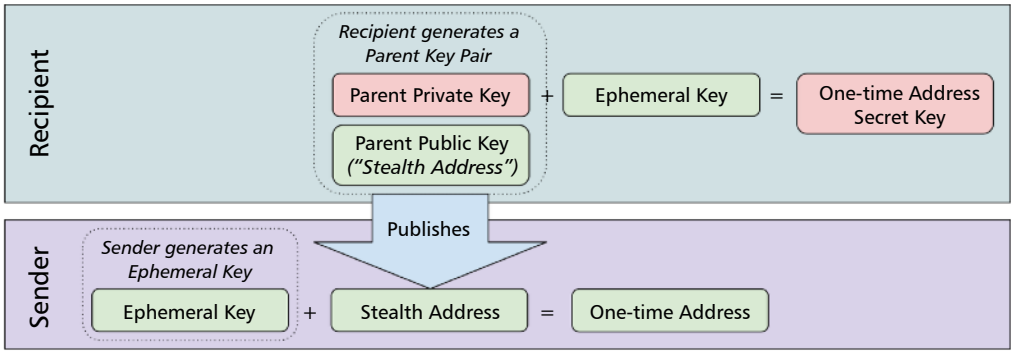
It works like this: the recipient generates a parent key pair and publishes the public key - this is the stealth address. Any sender can then use the stealth address to generate a new one-time use payment address. The recipient uses their parent *private* key to calculate the one-time use payment address's secret key, which is required in order to spend the funds.

There is no way for an observer to link a one-time use payment address to the stealth address used to generate it. More importantly, only the recipient can calculate the one-time use payment address's secret key. Despite the fact that it is the sender who generated the new one-time use payment address, the sender does not know the secret key and there is no way for he or she to calculate it.

How Stealth Addresses are Used to Generate One-time Use Payment Addresses

The one-time use payment address is generated using a variant of the Diffie-Hellman key exchange protocol[14]. The sender generates an ephemeral key and uses it in conjunction with the stealth address to create the one-time use payment address. The ephemeral key is then attached to the transaction.

The recipient can use the ephemeral key in conjunction with his or her parent private key to calculate the one-time use payment address' secret key.



Because the recipient does not know the one-time use payment address in advance, he or she must scan all transactions and attempt to generate the secret key for each transaction's destination address by using the ephemeral key attached to the transaction. This can be done automatically by a process

[14] W Diffie & M E Hellman: "New Directions in Cryptography" (1976)

that monitors the blockchain and checks each new transaction. Obviously, this is more computationally-intensive than simply watching out for transactions sent to a specific address, but the additional computational burden is not overly onerous.

The end result is an approach that provides the same (limited) degree of privacy for the recipient of a transaction as what is provided by manually-generated one-time use payment addresses. However, there is also the added benefit that the recipient need only manage a single parent key, instead of generating a new address and associated key pair for each transaction.

Stealth addresses are used in both the CryptoNote and Zcash protocols in conjunction with other techniques (ring signatures and zk-SNARKs, respectively) to provide stronger privacy for both the counterparties to transactions than is possible using one-time use payment addresses alone.

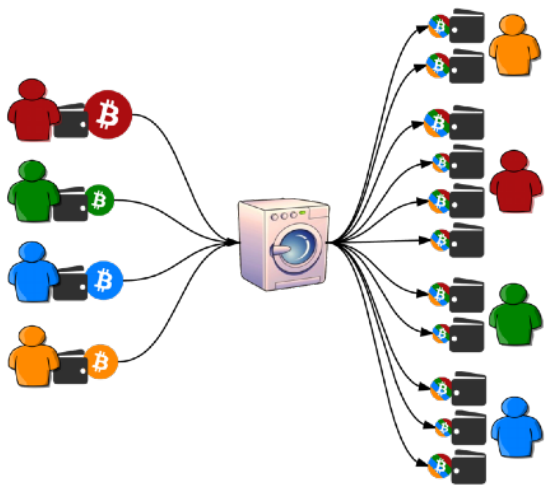
Stealth addresses have also been retrofitted into Bitcoin but require the use of specific wallet software (e.g., Dark Wallet[15]) and, as they are not native to the Bitcoin protocol, transactions that use stealth addresses can easily be identified. Specially, the presence of an ephemeral key on a Bitcoin transaction indicates that it is sent to a stealth address. Since there are few of them, these transactions stand out and result in some loss of privacy.

A number of people have contributed to the development of stealth addresses. Peter Todd published a formal description in January 2014[16], in which he credits a number of people as having contributed to their development.

2.6 Mixing

Transactions in a public blockchain like Bitcoin are linked and open to graph analysis. It is simple to follow the flow of funds from address to address. One-time use payment addresses do not solve this. The path of the funds from sender to receiver is clear. Mixing is a technique to confuse this path.

Bitcoin users can make it difficult to trace their transactions by taking advantage of Bitcoin mixing services (sometimes referred to as tumblers) like Bitcoin Fog and BitMixer, which receive bitcoins from many users, mix them together and send them back out to different addresses at different times and split into smaller amounts, making it more difficult to trace the flow of funds. These are third party services that take custody of the many users' coin for a period of time, mix them with each other, before sending them out. Note that any third party wallet or exchange service (such as Coinbase, Bitstamp, Bitfinex, etc.) can be an effective mixer if there is enough volume. Some Bitcoin companies like Btc-e and Localbitcoins are often used to mix coin in exactly this way because these companies either do not do proper Know-Your- Customer (KYC) or do not respond to law enforcement requests.



The mixer services generally charge a percentage of the funds mixed and are often accessed through Tor to conceal users' IP addresses. (IP addresses are one example of metadata that can be used to extract transaction details.) Mixers are often used by customers of dark markets, hackers, fraudsters

[15] https://www.wired.com/2014/04/dark-wallet/

[16] https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2014-January/004020.html

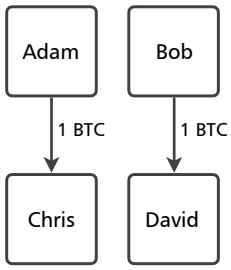
who steal bitcoins and criminals who use ransomware and threaten Distributed Denial of Service (DDoS) attacks[17] to extort payment in Bitcoin.

However, they are also used by individuals who are not engaged in illicit or criminal activities but want to preserve the privacy of their Bitcoin transactions. Therefore, use of such a service does not necessarily imply that the user has engaged in criminal or illicit activity. Users of mixers will mix their transactions with transactions that are engaged with illicit activity, thus tainting their transactions as well.[18]

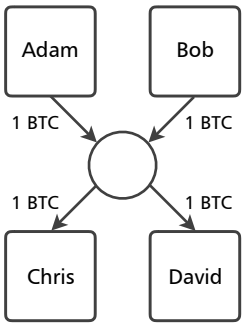
For this type of mixing to be effective, the mixer needs many users with many Bitcoins to mix. The longer the mixer holds the coin and mixes them, the better the mixing. The risk here, is that the mixer can steal the coin themselves, can shut down or be hacked in this period. The mixer also knows where the coin actually is sent to; so, if the mixer keeps logs, that record could compromise the mixing. The mixer could even leak the data about how to parse the coin on purpose or by accident. The mixers are almost always anonymous themselves, so the user has absolutely no recourse if the mixer does anything that compromises the mixing.

Coinjoin

The weak point in the mixing described above is the third party mixer who needs to be trusted. Coinjoin is a mixing technique that tries to remove the third party requirement. Coinjoin uses the property that a single transaction can have multiple inputs and multiple outputs. If you have two parties A and B who want to send one Bitcoin each to C and D respectively, then they can combine both sends in a single transaction. This transaction would have two inputs, A and B, and two outputs, C and D. The order of inputs and outputs is randomized, so it is unclear who C and D receive coins from. One interpretation is that C receives one Bitcoin from either A or B. Another interpretation is that both A and B send half a Bitcoin to C. if you have more people participating in a single transaction, i.e. joining their transaction into one coinjoin transaction, than the uncertainty of the source and destination of the coins is even greater.



Two transactions,
no mixing, no coinjoin



One coinjoin transaction

A trusted third party is not needed to hold the coins as required by mixing services, but all the parties involved in a coinjoin transaction must communicate with each other to sign the coinjoin transaction. This coordination is difficult for several reasons. First, at any given time, there may not be enough people who want to participate in a coinjoin transaction. Second, the participants in a coinjoin transaction may know who is participating in their coinjoin transaction and know how to unmix that transaction. A third party service can be used to coordinate coinjoin transactions, but then that third party will know how to unmix the transactions.

---

[17] DDoS attacks flood the bandwidth of the victim's system with network traffic and effectively shut down the victim's system. http://www.zdnet.com/article/how-bitcoin-helped-fuel-an-explosion-in-ransomware-attacks/

[18] George Fogg. "The UCC and Bitcoins: Solution to Existing Fatal Flaw." (Bloomberg, April 2015)

CoinShuffle is a technique proposed to allow for decentralized coinjoin transactions where all parties who participate in a coinjoin can create the transaction without a third party and no party has any knowledge of how to unmix the coinjoin besides knowing their own input and output addresses. Some weaknesses of CoinShuffle are, like coinjoin, there may not be enough users participating in it for the mixing to be effective, and there is an increased communication overhead.

CoinShuffle example with three parties, A, B, C who want to initiate one coinjoin transaction. A will use C's public key to encrypt A's destination, and send this message to B. B will then use C's public key to encrypt B's destination, and send both messages to C. C can now decrypt both messages and know all three destinations. C can set up the transaction and sign it, then pass it to both A and B to sign. All parties will only sign if the transaction contains the correct destinations and no party can unmix the transactions of the other parties.

2.7 Sidechains & State Channels

Sidechains and state channels (also known as payment channels) can be used to enhance privacy. Both are techniques for executing transactions "off-chain" by holding funds in escrow on the primary blockchain and redistributing the funds at a later point to reflect the net outcome of the transactions that have taken place in the state channel or on the sidechain.

For the purpose of this report, we define a sidechain as a parallel blockchain, that sits alongside the primary blockchain, serving multiple users and likely persisting permanently. We define a state channel as a temporary payment channel that is established between a specific (and, typically, limited) set of users.

When a state channel is opened, at least one of the parties commits funds to the channel by depositing them into a smart contract, and both parties exchange a cryptographic commitment which specifies how the funds held in the contract are to be divided between the parties when the channel is closed.[19] Each time a payment is made within the channel, the parties create an updated commitment and revoke the old one to reflect the new balances.

At any point, either party can unilaterally trigger the closure of the channel using the most up-to-date commitment. The smart contract, which has effectively been acting as an escrow account, will then disburse the funds it holds to the parties, splitting them as indicated by the commitment.

State channels provide no privacy for the parties to the channel but they do provide confidentiality for the transactions that take place within the channel (with the caveat that the amount deposited and withdrawn by each participant is visible on the primary blockchain).

With sidechains, a user effectively deposits funds "into" the sidechain, and later he (or, more likely, other parties to whom he has sent those funds on the sidechain) can "withdraw" funds from the sidechain, back onto the primary blockchain. Like state channels, sidechains provide no enhancement to privacy (beyond that provided by the primary blockchain) when depositing or withdrawing funds. The degree of confidentiality and privacy provided for transactions that take place on the sidechain depends on what technologies the sidechain implements (e.g. Blockstream's Elements sidechain technology uses Pedersen Commitments to conceal transaction details). However, even without additional confidentiality- and privacy-preserving technologies, sidechains can act in a manner similar to mixers, helping obfuscate the destination of deposited funds and the source of withdrawn funds.

[19] For more on how Ethereum defines a "smart contract" in the context of state channels see: Ethereum: Platform Review by Vitalik Buterin - https://r3cev.com/s/Ethereum_Paper-97k4.pdf

BOLT (Blind Off-chain Lightweight Transactions)

With existing state channel implementations, both the parties to a channel have full visibility of what transactions take place. A recently-published protocol[20] called BOLT (for "Blind Off-chain Lightweight Transactions") would add an extra layer of privacy within the state channel by concealing individual payments within the set of all payments made to that receiver across all the BOLT channels he is a party to.

The total amount committed to a specific BOLT channel is still visible to the recipient when the channel is opened, as are the closing balances at the point the channel is closed but individual payments are anonymous to the recipient. She will know that she has received a payment (and how much the payment is for) but if she has multiple BOLT channels open, she will not be able to tell which of the channels the payment came from.

Tumblebit

Tumblebit is a payment mixing technique that allows for an untrusted 3rd party mixer to facilitate mixing of payments.[21] It works by creating two payment channels, one between the payer A and the mixer M, and one between the mixer M and the payee B. Payment amounts must be the same amount (i.e., one Bitcoin) for payments by all participants for the mixing to work, and the mixing is only as good as the number of participants at the time of the mixing. The mixer does not know the direct matches between payers and payees, only the list of all payers and payees at a given time and how much each paid and received. This could be used to unmix some of these transactions via data analysis.

One advantage of using payment channels in Tumblebit is that, besides the opening and closing of the payment channels which must be on the blockchain, the payment phase where A and B communicate is done off-blockchain. The disadvantage of this is that there are a number of steps that need to be completed for the payment to work. After A pays the mixer, for the payee B to be paid, B must get a secret from A and then use that secret to receive payment. If B does not get the secret or fails to use it before a cutoff time, then B does not get paid, even though A has already paid the mixer, who can then choose to keep the payment.

The Tumblebit white paper was published in September 2016. An implementation is currently under development.[22]

2.8 Pedersen Commitments with Range Proofs

Commitments are a cryptographic mechanism which allow one to keep a piece of data secret but "commit" to it by publishing a hash of the data. A blinding factor[23] can be added when the data size is short (e.g. a number) to minimise the risk of the data being unmasked by brute force search.

Having "committed" to the piece of data by publishing the hash, the publisher can later reveal both the blinding factor and the data, allowing others to verify that the hash of the blinding factor and data matches the hash that they published.

---

[20] M Green & I Miers. "Bolt: Anonymous Payment Channels for Decentralized Currencies."

[21] E Hellman et al. "TumbleBit: An Untrusted Bitcoin-Compatible Anonymous Payment Hub."

[22] https://github.com/BUSEC/TumbleBit

[23] A blind signature scheme (by David Chaum) allows a message to be signed by someone without revealing the message to the signer. This is done by first "blinding" the message (m) with a blinding factor (b), blind(m, b). Then the signer signs with his private key (k), sign(blind(m, b), k). Finally, the party that blinded the message can unblind the result to get the signed message, sign(m, k).

A Pedersen commitment works like the above but with an additional property: commitments can be added, and the sum of a set of commitments is the same as a commitment to the sum of the data (with a blinding key set as the sum of the blinding keys):

C(BF1, data1) + C(BF2, data2) == C(BF1 + BF2, data1 + data2)

C(BF1, data1) - C(BF1, data1) == 0

In other words, the commitment preserves addition and the commutative property applies.
If data_n = {1,1,2} and BF_n = {5,10,15} then:

C(BF1, data1) + C(BF2, data2) - C(BF3, data3) == 0

and so on.

Quoted directly from https://elementsproject.org/elements/confidential-transactions/

Confidential Transactions

The use of Pedersen commitments as a means of concealing the amount being transferred in a Bitcoin transaction was first proposed by Adam Back in October 2013.[24] It was subsequently formalized by Greg Maxwell[25] under the name "Confidential Transactions" and was implemented in Blockstream's Elements sidechain solution.

Confidential Transactions leverages the additively homomorphic qualities of elliptic curve cryptography public keys, to prove that the inputs and the outputs of a confidential transaction sum to zero. Range proofs are a cryptographic mechanism used to prove that a value lies within a certain range, without revealing the value. In confidential transactions, they are used to prove that none of the outputs are negative, which would otherwise allow the sender to create money out of thin air.[26]

The sender discloses the value of the transaction and the blinding factor they used to the receiver, who can then verify the value of the transaction. The receiver then has the option of using the confidential value he has just received in a new confidential transaction. The Pedersen commitment allows the concealed value to be spent without revealing it.

Confidential Transactions have been implemented as part of Blockstream's Elements sidechain platform, and will shortly be implemented for the Monero cryptocurrency.

Because Confidential Transactions only conceal the amount being transferred, information leakage can theoretically occur if knowledge that a specific transaction has taken place (e.g. receiving a competitive RFQ on an MTF; observing the disappearance/filling of an order on an exchange) can be correlated with the creation of a transaction on a blockchain.[27]

Mimblewimble

Mimblewimble is an outline proposal released in July 2016[28] under the pseudonym "Tom Elvis Jedusor" (which, like the name "mimblewimble," is a Harry Potter reference), for adding privacy to Bitcoin. Mimblewimble leverages the techniques that underpin Confidential Transactions and CoinJoin to combine all the transactions in each block into a single large transaction with many inputs and outputs. This is similar to combining all transactions in a block into a single CoinJoin transaction. Every transaction would be mixed with all the other transactions in the same block, making graph analysis extremely difficult, if not impossible.

---

[24] https://bitcointalk.org/index.php?topic=305791.0

[25] https://people.xiph.org/~greg/confidential_values.txt

[26] This proof of non-negativity contributes significantly to the large size of Confidential Transactions (typically 5,000 bytes).

[27] https://leastauthority.com/blog/zerocash_and_confidential_transactions.html

[28] https://download.wpsoftware.net/bitcoin/wizardry/mimblewimble.txt

Mimblewimble also has potential advantages for scaling and opens up the possibility of replacing payment addresses with blinding factors. However, this would come at the cost of relinquishing the use of Bitcoin's scripting language.

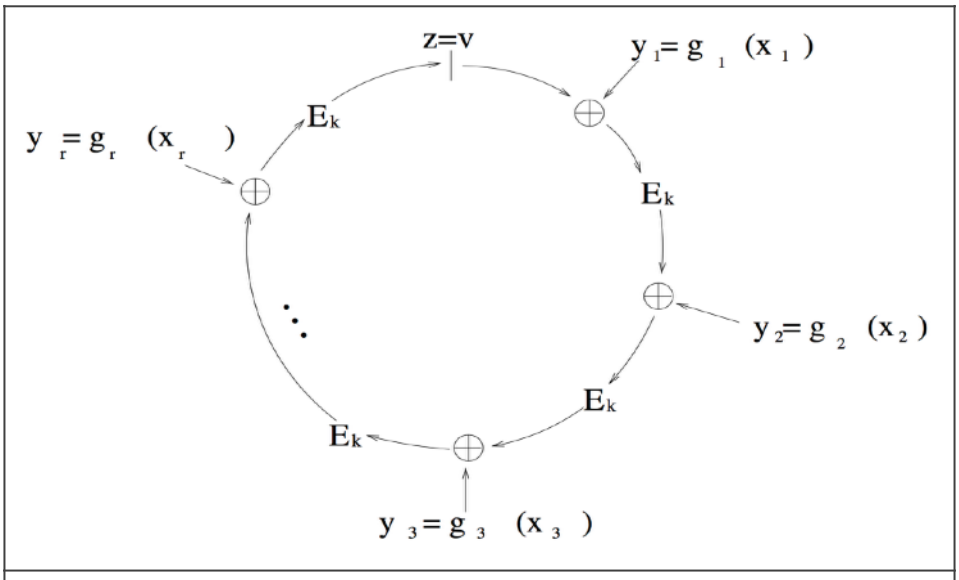
There are, as yet, no implementations of the Mimblewimble technique. It is unlikely to be incorporated into Bitcoin itself, because of the lack of support for Bitcoin script, but may eventually find use as a sidechain.

2.9 Ring Signatures

In order to verify a standard digital signature, the verifier must know which public key was used to create the signature. This is what allows observers to trace the flow of funds from address to address. Ring signatures were invented by Rivest, Shamir and Tauman in 2001,[29] as a means of creating a signature with a group of possible signers, without revealing which member of the group actually created the signature.

A ring signature is created using group of keys, which includes the signer's own key (for which he possesses the secret key) and a number of other public keys chosen by the signer (no consent or participation by the other public keys' owners is required). A third party can verify that the resultant signature was created using one of keys in the group, but it is not possible to identify which key in the group belongs to the signer.

This is done by constructing a ring equation using the public keys, where the output of one calculation becomes the input of the next. Someone who knows one of the secret keys can ensure that the final output of the ring equation (z) is equal to the initial input (v).



Source: Rivest R.L., Shamir A., Tauman Y. (2001) How to Leak a Secret[29]

In the case of a blockchain, ring signatures can be used to create transactions where the sender cannot be identified (although an observer will know that the sender is one of a specific group that corresponds to the public keys used in the ring signature).

[29] http://link.springer.com/chapter/10.1007%2F3-540-45682-1_32

CryptoNote

The use of ring signatures to conceal the origin of a blockchain transaction was first described as part of the CryptoNote protocol, first released in December 2012[30] and updated in October 2013,[31] by Nicolas van Saberhagen. Like Satoshi Nakamoto, this name appears to be a pseudonym, as do the other names listed as authors of the CryptoNote Standards.[32]

With CryptoNote, the sender constructs the ring signature using the public keys that correspond to addresses that hold cryptocurrency balances identical to the sender's own address. An observer can be confident that the creator of the resultant transaction possesses a secret key that corresponds to an address holding the amount of cryptocurrency being sent in the transaction, but they cannot tell which address the transaction is being sent from

However, by themselves, ring signatures would allow double-spending. This is because the ability to obfuscate which address the coins are coming from means that it is not possible to determine which of the coins owned by the addresses that are used in the ring signature should be marked as sent.

CryptoNote's solution to this problem is called one-time use ring signatures. It leverages traceable ring signatures, which were originally conceived as a means of supporting anonymous voting, while preventing voters from voting twice.

> A traceable ring scheme is a ring signature except it can restrict "excessive" anonymity. The traceable ring signature has a tag that consists of a list of ring members and an issue that refers to, for instance, a social affair or an election. A ring member can make any signed but anonymous opinion regarding the issue, but only once (per tag). If the member submits another signed opinion, possibly pretending to be another person who supports the first opinion, the identity of the member is immediately revealed. If the member submits the same opinion, for instance, voting "yes" regarding the same issue twice, everyone can see that these two are linked.
>
> - E Fujisaki & K Suzuki, "Traceable Ring Signature" [33]

With CryptoNote, a "key image" (effectively a hash of the private key the signer used) is added to the transaction. Any attempt to double-spend will result in the same key image being used. Nodes maintain a list of all the key images ever used and reject any new transactions where the key image has been used previously. A side-effect of this approach is that addresses must never be reused. If they are, only one of the transactions sent to the that address would be spendable.

With CryptoNote, the source of a transaction cannot be provably identified, providing "plausible deniability." In other words, an observer can tell that the transaction originated with a member of the "ring" of addresses, but not which one. Neither the recipient address nor the amount being transferred are concealed. With that said, CryptoNote's use of one-time use addresses can make it difficult to link recipient addresses.

The images below illustrate how the use of ring signatures can conceal which input (UTXO) is being spent in a transaction.

---

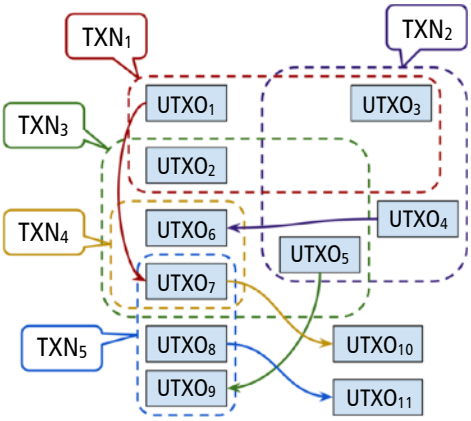
[30] https://cryptonote.org/whitepaper_v1.pdf

[31] https://cryptonote.org/whitepaper.pdf

[32] https://cryptonote.org/standards/

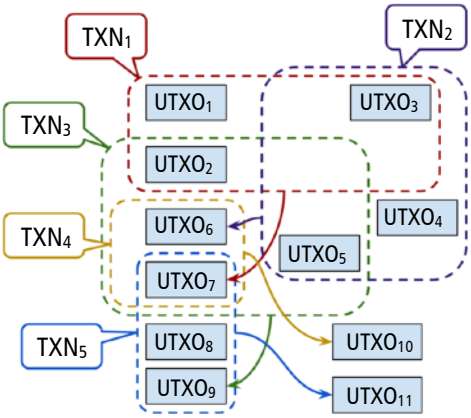
[33] https://eprint.iacr.org/2006/389.pdf

What is occurring:



Transaction	Input UTXO	Mixing (other UTXOs in the ring signature)	Output UTXO
TXN_1	$UTXO_1$	$UTXO_2$, $UTXO_3$	$UTXO_7$
TXN_2	$UTXO_4$	$UTXO_3$, $UTXO_5$	$UTXO_6$
TXN_3	$UTXO_5$	$UTXO_2$, $UTXO_6$, $UTXO_7$	$UTXO_9$
TXN_4	$UTXO_7$	$UTXO_6$	$UTXO_{10}$
TXN_5	$UTXO_8$	$UTXO_7$, $UTXO_9$	$UTXO_{11}$

What a third party observer sees:



Transaction	Possible Inputs	Output UTXO
TXN_1	$UTXO_1$, $UTXO_2$, $UTXO_3$	$UTXO_7$
TXN_2	$UTXO_3$, $UTXO_4$, $UTXO_5$	$UTXO_6$
TXN_3	$UTXO_2$, $UTXO_5$, $UTXO_6$, $UTXO_7$	$UTXO_9$
TXN_4	$UTXO_6$, $UTXO_7$	$UTXO_{10}$
TXN_5	$UTXO_7$, $UTXO_8$, $UTXO_9$	$UTXO_{11}$

Note that if the owner of $UTXO_6$ were to create a transaction with no mixins, that would reveal that $UTXO_7$ was the input for TXN_4, linking $UTXO_7$ to $UTXO_{10}$.

Monero

There are several cryptocurrencies that implement the CryptoNote protocol. The most notable is Monero. Work is underway[34] to implement Ring Confidential Transactions[35] in Monero, which will also conceal the amount being transferred.

To be effective in concealing the source of a transaction, ring signatures rely on the ability of the sender to find other keys holding the same amount of funds in order to create the group of keys required to generate the ring signature. The larger the group, the better; small groups can facilitate transaction graph analysis.

The figure below gives an indication of how large these groups are for the Monero blockchain. The "mixin" figure indicates how many keys *other* than the sender's were used when creating the transaction. As you can see, the majority of transactions used just one or two other keys.

[34] https://github.com/monero-project/bitmonero/pull/961

[35] S Noether. "Ring Confidential Transactions."

mixins used in transactions (%)

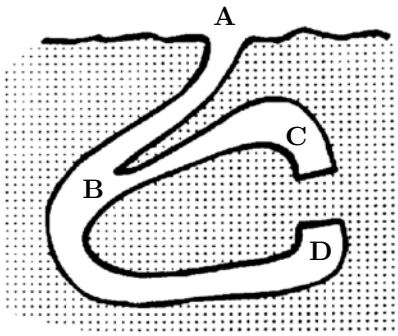
mixin:	none :(	1 - 2	3 - 9	9 - 99	> 99	highest	lowest	total tx
last day	0.13	64.42	33.27	2.18	0.00	20	0	1560
last week	0.06	66.25	32.04	1.65	0.00	23	0	13043
last month	0.19	66.35	31.14	2.28	0.04	200	0	50195

Source: http://moneroblocks.info/stats

2.10 Zero-Knowledge Proofs

Zero-knowledge proofs are cryptographic protocols that allow one party to prove to another party that a statement is true without revealing any information apart from the fact that the statement is true.

In *How to Explain Zero-Knowledge Protocols to Your Children,*[36] Jean-Jacques Quisquater and Louis Gillous use the analogy of a cave to explain zero-knowledge proofs.



The cave that is in question has two branches that are connected by a secret passage that can only be opened by uttering a magic passphrase. Mick has discovered the cave's secret and wants credit for his discovery, so he seeks out a journalist to write a story about it, but the journalist is skeptical and will only publish a story crediting Mick with the discovery if Mick can prove that he knows how to access the secret passage.

However, Mick doesn't want to reveal the magic passphrase to anyone else, so he and the journalist come to an agreement. Having inspected both passageways and having confirmed that they both appear to be dead-ends, the journalist will return to the entrance to the cave (A) while Mick goes inside, walks down one of the passages and waits at the end (at either C or D).

The journalist (who doesn't know which passage Mick went down because of the bend in the cave entrance) then goes inside the cave, as far as the point where the passages diverge (B), and flips a coin. If it comes up heads, she will call out "Right!" (D); if it comes up tails, she will call out "Left!" (C).

Mick listens to what the journalist calls out and returns to point B along the corresponding passage. If, when he entered the cave, he walked down the correct passage, he simply retraces his footsteps and the journalist will see him coming out of the correct passage. If, however, Mick went down the other passage, he whispers the magic words, walks through the secret passageway, and walks back up the correct passage, then the journalist will see him coming out of the correct passage.

Performing this ritual once doesn't really prove anything. After all, Mick could have gotten lucky by choosing to go down the same passageway that the coin specified (a 50% chance). However, if they perform the ritual multiple times, the probability that Mick is lucky enough to pick the correct passage every time halves each time they repeat the ritual. If they repeat it five times, the chance that Mick was able to guess which tunnel the report would choose is 1 in 2^5 (32) or 3.125%. If they do it ten times, the probability that Mick is fooling the journalist drops to 1 in 2^{10} (1024) or 0.098%.

Eventually, if the journalist sees Mick return via the correct passage enough times, she will be convinced that Mick is telling the truth and Mick will have achieved his objective; he would have

[36] http://pages.cs.wisc.edu/~mkowalcz/628.pdf

proven to the journalist that he knows the cave's secret, without revealing the secret.[37] Zero-knowledge proofs operate on a similar principle; they don't prove an assertion beyond all doubt. There is always a small probability, known as the soundness error, that the prover has been lucky but a well-constructed zero-knowledge proof allows for the soundness error to be reduced to a vanishingly small probability.

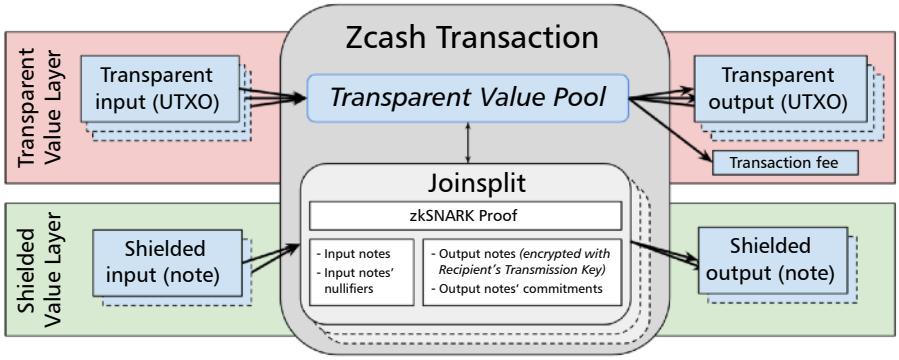
In the example given, the prover and verifier must interact. However, there exists a class of zero-knowledge proofs that are non-interactive, where a cryptographic system replaces the interaction, and complex arithmetic circuits are used to validate the proofs.

Zcash

In 2014, a group of computer scientists and cryptographers released a paper outlining the Zerocash protocol,[38] which describes how zero-knowledge proofs can be used to support blockchain transactions in which the sender, recipient and assets being transferred are kept private. A team led by Zooko Wilcox has implemented the Zerocash protocol using a fork of the Bitcoin codebase, to create a new cryptocurrency called Zcash, which launched on 28th October 2016.[39]

Zcash leverages what it describes as zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARKs) to support the "shielding" of coins by allowing them to be added (or "committed")[40] to a list of private coins (referred to as "notes").

A Zcash transaction can convert coins (representing transparent value) into notes (shielded value), convert notes back into coins and transfer coins and notes to new recipients.



Transactions that involve notes (either as inputs and/or as outputs) are constructed using a cryptographic operation called a joinsplit, which creates a zk-SNARK proof to prove that:

a. The note(s) being transacted are valid (meaning they are "on the list")
b. The note(s) being transacted had not previously been used in another transaction (i.e. proof that double-spending is not occurring)
c. The sum of the transaction's outputs is equal to the sum of its inputs

The use of zero-knowledge proofs means that these three facts can be proven without providing any information about the source or the value of the transaction. In other words, there is no need to reveal which notes are being used, or what value is being transferred.

By using stealth addresses, the destination of transactions is also concealed. The end result is when notes are sent using a transaction with no transparent inputs or outputs (other than the transaction

---

[37] In reality, Mick could prove that he knows the cave's secret by simply walking down one passageway and back up the other, while the journalist waited at point B but that wouldn't make for a very good analogy.

[38] E Ben-Sasson et al. "Zerocash: Decentralized Anonymous Payments from Bitcoin."

[39] https://z.cash/

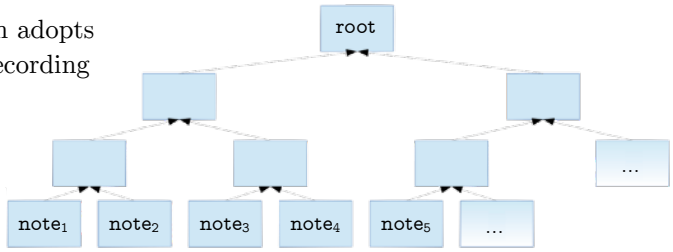
[40] See the description of commitments in section 2.8

fee), the transaction benefits from a level of privacy similar to that produced by ring signatures. However, there is a key difference; with Zcash, the equivalent of the ring (often referred to as the anonymity set) encompasses *all* coins that have ever been shielded, or committed to the list of notes, instead of a limited subset. This means that transaction graph analysis is entirely ineffective.

Another key advantage over ring signatures is that there is no way for the sender to ascertain when the recipient spends the funds. This is referred to as "perfect forward privacy." This combination of techniques provides for a particularly strong form of privacy. No information about the sender, the recipient or the transaction is revealed. Third-parties can only ascertain that a transaction took place. This prevents the type of information leakage made possible by correlating on-chain transactions with off-chain events, which is possible with Confidential Transactions.

How Zcash Works

In order to provide this level of privacy, Zcash adopts a new mechanism to keep track of notes by recording them in a binary tree data structure. This facilitates the creation of the zk-SNARK proofs required to maintain the integrity of the blockchain, while supporting the creation of private transactions.



Every time a new note is created, it uses up a slot in the tree; slots that have been used, cannot be reused. The depth of the binary tree determines the maximum number of notes that can be committed to that tree. For a tree of depth d, the maximum number of joinsplits that can be carried out using that blockchain is $2^{(d-1)}$, so a depth of 20 would support a maximum of 524,288 joinsplits, while a depth of 29 would support 268,435,456.

When the tree is exhausted, a new set of parameters (see next section) must be generated and adopted by the blockchain's participants. Obviously, the deeper the binary tree, the better. However, increasing the depth of the binary tree has a performance downside. Joinsplits are computationally-intensive operations, and their complexity increases as the depth of the binary tree increases. Zcash launched with a tree depth of 29, and has benchmarked the joinsplit operation at 48 seconds on a typical desktop CPU (although this metric is expected to improve as the code is optimized).

Zcash's Trusted Setup

Generating zk-SNARK proofs require a set of parameters that are shared between the prover and the verifier. These parameter must be generated in advance. The process for doing so creates a toxic byproduct in the form of a set of data (referred to as "toxic waste") that could be used to subvert the blockchain by creating counterfeit coins that are indistinguishable from real coins. The security of the blockchain relies on this data being destroyed. Zcash is, therefore, often referred to as requiring a "trusted setup," meaning users must trust that the "toxic waste" has been destroyed. To minimize this risk, some of the Zerocash authors designed a multi-party computation protocol to allow generation of the shared parameters to be shared amongst a group. Each participant generates one piece of the parameters. As long as one member of the group destroys their portion of the toxic waste, it cannot be reassembled and the blockchain can be considered secure.[41]

The Zcash team plan to extend the protocol to support multiple assets, and they plan to offer a zk-SNARKs-based "security layer" that can add support for private transactions to other blockchain solutions. (In fact, Zcash itself is effectively a Bitcoin-style blockchain with the zero-knowledge security layer bolted on.)

---

[41] E Ben-Sasson, A Chiesa, M Green, E Tromer, M Virza. "Secure Sampling of Public Parameters for Succinct Zero Knowledge Proofs."

3 Confidentiality and Privacy of Smart Contracts

Smart contracts are of particular interest to those in the financial sector.[42] For the purposes of this paper, smart contracts are effectively computer programs that embody a self-executing and -enforcing contract to which users may become party, by interacting with it electronically (subject to whatever limitations or requirements embodied in the contract). The concept is not a new one, but the advent of blockchain technology spurred interest in smart contracts because the blockchain eliminates the need to rely on a trusted third party to "execute" the contract, and enables to use of cryptocurrency as "programmable money."

An example of a simple smart contract is Bitcoin's "multi-sig" functionality, which provides support for escrow of bitcoins by creating transactions that require 2-of-2 or 2-of-3 parties to sign the transaction before funds are released.

Bitcoin's support for smart contracts is extremely limited. Ethereum, on the other hand, incorporates full support for smart contracts, with a "Turing-complete" smart contract language that supports complex logic, with few limitations other than the cost of executing the smart contract's code. Ethereum smart contracts are written to the Ethereum blockchain and executed by the blockchain's nodes, in much the same way that a transaction is executed. Users interact with smart contracts by creating special transactions that call functions within the smart contract. Smart contracts can themselves hold and control ether, which opens up potential in terms of creating decentralised autonomous organisations (DAOs) with no managers: just stakeholders and a set of business logic that controls how the DAO functions and behaves.

However, because Ethereum's smart contracts are stored on a public blockchain, their underlying code (and the business logic that code represents), along with their state (i.e. what information is stored within the contract) and all inputs to and outputs from the contract, can be read and analyzed by anyone.

3.1 Hawk

Hawk is a framework for creating private smart contracts. It was designed by a group of cryptographic researchers from Cornell University and the University of Maryland and released as a whitepaper in late 2015.[43] It was originally designed to complement the Zerocash protocol and, in general, relies on being used with a cryptocurrency that supports private transactions.

Whereas existing smart contracts execute entirely on a public blockchain, Hawk splits the smart contract into two parts:

- a public contract, which runs on the public blockchain, and
- a private contract, which is executed "off-chain".

The public contract collects inputs and funds from the contract's participants (e.g. bids in an auction). The private contract takes the form of a cryptographic protocol which decides how the funds are collected and held by the public contract are to be distributed. The public contract will only distribute funds in response to a valid instruction from the private contract. The authors imagine this contract will also include a cryptographic verification that the private contract executed correctly. Because the private contract must be executed "off-chain", Hawk requires the involvement of a "manager" to execute the private contract. While the manager can see the transactions that take place in a private contract (and, hence, must be trusted by the participants to protect their privacy), she

[42] Note: as of this writing there is no standard, industry-wide consensus as to the definition of a smart contract. See for example, Christopher D. Clack, Vikram A. Bakshi, Lee Braine. "Smart Contract Templates."

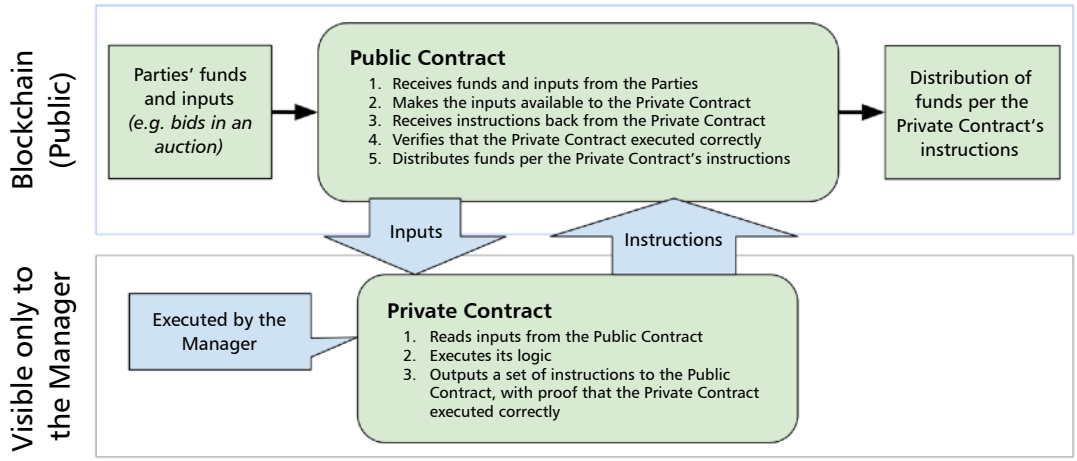
[43] A Kosba, A Miller, E Shi, Z Wen, C Papamantho. "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts."

cannot affect the outcome of the contract. In essence, the worst thing a manager can do is not execute (or abort) the private contract.

To protect against this, the public contract can be programmed to refund the parties' funds if the private contract isn't executed correctly in a timely fashion. The manager can also be required to make a security deposit into the public contract which they lose if they fail to execute the private contract.

The logic contained within the private contract can be kept private from everyone except the manager.[44] However, in practice, nobody is likely to use a private contract which they haven't had sight of. By inspecting the public and private contracts, the Parties can verify their behaviour, specifically to ensure that:

- the public contract will only re-distribute funds in response to a valid instruction from the private contract,
- the private contract is written in such a way that the manager cannot influence how it operates, and
- the logic implemented in the private contract is as advertised.[45]



If the public blockchain supports private transactions (e.g., Zcash), then the identity of the participants can be kept secret from everyone (including the manager). The inputs and funds the parties send to the public contract will only be visible to the manager.

3.2 Enigma

Enigma is a high-level description of a decentralised computation platform for processing encrypted data. It was designed by a team at MIT and the whitepaper was released in June 2015.[46]

Enigma is intended to complement an existing blockchain, with on-chain payments that has been used to incentivize off-chain processing of data, using distributed secure multiparty computation (MPC). Secure MPC was conceived by Andrew Yao in 1982.[47] It allows participants to contribute sensitive data as inputs to a calculation without revealing the information to anyone else. Yao uses the example of the Millionaires' Problem, where two millionaires want to find out which one of them is the

[44] This assertion is based on the assumption that the verification key provided by the Private Contract to the Public Contract doesn't reveal the structure of the Private Contract, which is not proven.

[45] It should be noted that Hawk cannot protect participants from implementation bugs such as the one that affected TheDAO.

[46] G Zyskind, O Nathan, A Pentland. "Enigma: Decentralized Computation Platform with Guaranteed Privacy."

[47] A Yao. "Protocols for Secure Computations."

wealthiest, without revealing how wealthy they are. It is possible to construct a protocol, using secure MPC, which makes it possible to determine whether one value is larger than another, without the need for either party to reveal the value.

The first significant application of this type of protocol took place in 2008, when secure MPC was used to conduct a private auction amongst Danish farmers for the rights to supply sugar beets to Danisco, the sole sugar beets process in Denmark.[48] The farmers encrypted their bids and offers before submitting them, and a secure MPC protocol was used to calculate the clearing price.

With Enigma, the MPC protocol (i.e., the computational logic) is compiled as an arithmetic circuit, with the individual arithmetic operations distributed across a number of untrusted nodes.[49] The users' data is encrypted, split up, and distributed across the nodes. Each node carries out a single step in the computation and passes the (encrypted) result on to the next node. The Enigma protocol provides the framework for coordinating the computation. Eventually, the user receives the encrypted result, which only they can decrypt.

While the computational logic cannot be considered to be confidential (as it is disclosed to the computation nodes), the data being processed is encrypted, so the inputs and outputs remain confidential.

Enigma is designed to complement an existing blockchain, with a cryptocurrency that can be used to pay fees. The computation nodes are also required to make security deposits to ensure that they execute their code

In its current form, Enigma is primarily a protocol for enabling distributed processing of confidential data, as opposed to a smart contract protocol. No implementation has been released as of yet, and It remains to be seen whether Enigma could be adapted to provide support for smart contracts in the same way that Hawk does.

4 Conclusion

Blockchain technology has had proven success with providing integrity and trust, while other properties like scalability, confidentiality and privacy are less mature and active areas of research and development. This survey focused on confidentiality and privacy, covering the different technologies that are being used and developed today to make blockchains more private.

The "low-tech" solutions of restricting read access, storing data off-chain and one-time use addresses are relatively simple and low risk. However, they each have their own particular limitations.

Privacy is often achieved via some form of mixing (e.g. CoinJoin, ring signatures), which may turn out to be legally problematic, from the perspective of asset provenance.

Ring signatures and stealth addresses are based on mature cryptography, and have been deployed for some time. The recent and more private crypto-currencies use these techniques.

Pedersen commitments, sidechains, state channels and the Enigma protocol are all based on mature cryptography, however they have undergone little real-world testing.

The newest privacy techniques, zk-SNARKs, HAWK and BOLT, rely on new cryptographic techniques. The first widespread deployment of zk-SNARKs has only just taken place, and it remains to be seen whether it will survive being battle tested. Neither HAWK nor BOLT have been implemented as yet.

---

[48] P Bodetoft et al. "Secure Multiparty Computation Goes Live."

[49] See the description of arithmetic circuits on page 7

5 Appendix

5.1 Quantum Computing

Quantum computing is a nascent form of computing that may have important consequences in the next few decades. Potentially, it could have the ability to decrypt many encryption systems that we rely on today to keep data secure and private. Quantum computing takes advantage of quantum mechanical properties to carry out computation. The two properties are superposition and entanglement. Data in a quantum computer is measured in quantum bits or qubits.

A bit in classical computing is a single binary value, one of two possible values, true or false (1 or 0). A qubit in quantum computing is a mixture of two possible values, or superposition of the two values. A bit is either true or false, but a qubit is a linear combination of both true and false. This is quantum superposition and it allows a qubit to encode a lot more data than a bit. For example, n bits are described by n binary numbers, while n qubits are described by 2^n-1 complex numbers. Quantum entanglement is the other property that is unique to quantum computers. Entanglement is the property that the values (or states) of groups of qubits can be correlated, or entangled.

Superposition and entanglement in qubits is what gives quantum computers their advantages over classical computers. Quantum computers are able to solve certain problems much faster than possible today with classical computers. One quantum algorithm that is well known is Shor's algorithm, formulated in 1994, that can factorize an integer N in polynomial log(N) time, as opposed to sub-exponential time for a classical algorithm. Many of today's encryption systems (such as RSA) rely on the computational difficulty of factoring, but if a powerful enough quantum computer is built, those encryption systems will easily be decrypted and defeated. Currently, the most powerful quantum computers that are known to exist are still small and can only factor small numbers, but given the continuous advancement of technology, it may be only a matter of time before we will need to move to quantum hard algorithms for encryption.

The public key cryptography used in Bitcoin can be defeated by quantum computers. Luckily, there are cryptosystems that are thought to be immune to attack by quantum computers, such as the McEliece and New Hope cryptosystems. Switching to a different cryptosystem is not too difficult, but does entail some additional computation and increased signature size and block size.

5.2 Introduction to RSA

Quotes outlined in blue are taken directly from the RSA paper.

> A message is encrypted by representing it as a number M, raising M to a publicly specified power e, and then taking the remainder when the result is divided by the publicly specified product, n, of two large secret prime numbers p and q. Decryption is similar; only a different, secret, power d is used, where $e \cdot d \equiv 1 \pmod{(p-1)\cdot(q-1)}$. The security of the system rests in part on the difficulty of factoring the published divisor, n.

mod refers to the modulo operation, which finds the remainder that is left over when one number is divided by another. For example, 23 modulo 4 is 3, because 3 is the remainder that is left over when you divide 23 by 4.

The symbol ≡ signifies a congruence relationship:

For a positive integer *n*, two integers *a* and *b* are said to be **congruent modulo** *n*, written:

$a \equiv b \pmod{n}$,

if their difference $a - b$ is an integer multiple of n (or n divides $a - b$). The number n is called the *modulus* of the congruence.

For example,

$38 \equiv 14 \pmod{12}$

because 38 − 14 = 24, which is a multiple of 12.

Source: Wikipedia

Note that if $a \bmod n = b$ then $a \equiv b \pmod{n}$. For example, because $23 \bmod 4 = 3$, it follows that $23 \equiv 3 \pmod{4}$.

The first step in creating an RSA key pair is to pick two prime numbers: p and q.

You first compute n as the product of two primes p and q:

$$n = p \cdot q \ .$$

For our example, we choose 47 and 59 as the primes, which means that n is 2773.

You then pick the integer d to be a large, random integer which is relatively prime to $(p-1)\cdot(q-1)$. That is, check that d satisfies:

$$\gcd(d, (p-1)\cdot(q-1)) = 1$$

("gcd" means "greatest common divisor").

"Relatively prime" means that the only positive integer that divides both numbers (i.e. their greatest common divisor) is 1.

In our example, $(p-1)\cdot(q-1)$ equals 2668, so we need to find a number that is relatively prime to 2668, such as 157.

The integer e is finally computed from p, q, and d to be the "multiplicative inverse" of d, modulo $(p-1)\cdot(q-1)$. Thus we have:

$$e \cdot d \equiv \pmod{(p-1)\cdot(q-1)}.$$

In modular arithmetic, the **modular multiplicative inverse** of an integer a, modulo m is an integer x *such that*

$a\,x \equiv 1 \pmod{m}$

Source: Wikipedia

Computing e is quite complex to do manually but fairly straightforward using a computer.

There isn't always a valid value for e, in which case you have to start over from scratch with a new pair of primes.

In our example, 17 is a valid value for e.

We now have a public key, consisting of e and n (17, 2773) and a private key, d (157). The private key must be kept secret (as must the original two prime numbers, p and q).

To encrypt a message M with our method, using a public encryption key (e, n), proceed as follows. (Here e and n are a pair of positive integers.)

First, represent the message as an integer between 0 and $n-1$.

Then, encrypt the message by raising it to the eth power modulo n. That is, the result (the cipher text C) is the remainder when M^e is divided by n.

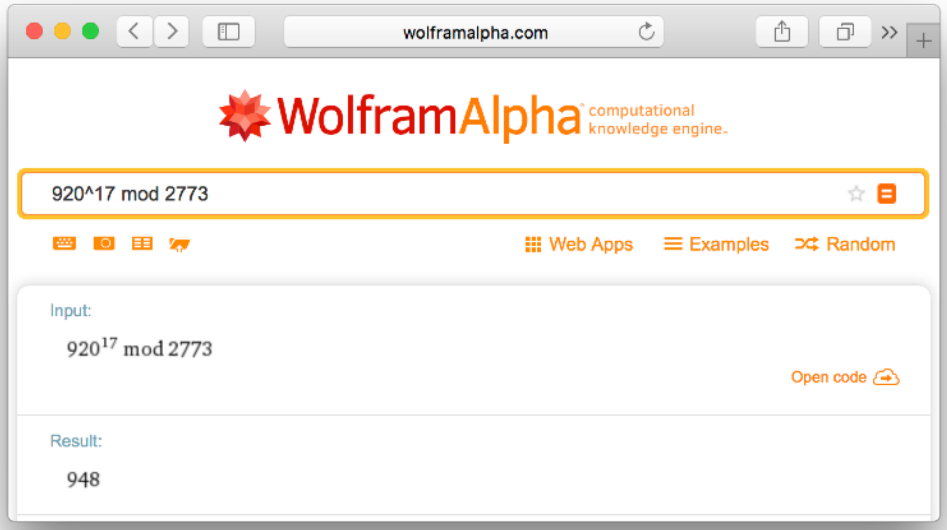
To decrypt the cipher text, raise it to another power d, again modulo n. The encryption and decryption algorithms E and D are this:

$$C \equiv E(M) \equiv M^e \pmod{n}, \text{ for a message } M\,.$$

$$D(C) \equiv C^d \pmod{n}, \text{ for a ciphertext } C\,.$$

If our message, M, is expressed as 920, we encrypt by calculating 920^{17} mod 2773 which results in a ciphertext of 948.

The website Wolfram Alpha is useful for performing the calculations described above. However, note that when inputting these calculations into Wolfram Alpha, you must express a^b as $a\hat{}\,b$.



To decrypt the ciphertext, you calculate 948^{158} mod 2773 which results in 920. This example is summarized in the table on the next page.

The strength of RSA encryption relies on the difficulty of figuring out what p and q are. The only way to do this is by factoring n (i.e., figuring out which two numbers, when multiplied together, result in n). Factoring a relatively small number like 2773 is not particularly difficult but, in practice, far larger numbers are used.

In cryptography, the size of a number is often defined by the number of bits required to express it in binary form. 2773 requires 12 bits (101011010101), so it would be referred to as a 12-bit number. Today, most RSA signatures are least 2048 bits long, and some are 4096 bits, which means that writing them down in decimal format could require up to 200 digits. Factoring a number that large is a task that even the most powerful supercomputers would take many years to complete.

Pick two primes: p, q	$p = 47,\ q = 59$
$n = p \cdot q$	$n = 47 \cdot 59$ $= 2773$
$\phi(n) = (p-1)\cdot(q-1)$	$\phi(n) = (47-1)\cdot(59-1)$ $= 46 \cdot 58$ $= 2668$
Choose d such that d and $\phi(n)$ are relatively prime	$d = 157$
Calculate e such that $e \cdot d \equiv 1 \pmod{\phi(n)}$	$e = 17$ $e \cdot d \bmod \phi(n) = 17 \cdot 157 \bmod 2668$ $= 2669 \bmod 2668$ $= 1$ ✓
The public key is (e, n)	$(17, 2773)$
The private key is (d)	(157)
Express your message as an integer M, where $0 < M < n$:	$M = 920$
Encryption: Ciphertext, $C = M^e \bmod n$	$C = 920^{17} \bmod 2773$ $= 948$
Decryption: Plaintext, $M = C^d \bmod n$	$M = 948^{157} \bmod 2773$ $= 920$

5.3 Digital Signature Example

Below is an example of an RSA digital signature generated and verified using the same RSA key pair that we created previously:

The public key is (e, n)	$(17, 2773)$
The private key is (d)	(157)
Express your message as an integer M, where $0 < M < n$:	$M = 920$
Signing: Signature, $S = M^d \bmod n$	$S = 920^{157} \bmod 2773$ $= 192$
Verification: Confirm that $S^e \bmod n = M$	$192^{17} \bmod 2773 = 920$ ✓

5.4 Demonstrating RSA's Multiplicative Homomorphism

The example below demonstrates the multiplicatively homomorphic qualities of the RSA cryptosystem. The plaintexts 41 and 24 are encrypted to produce the ciphertexts 1244 and 2179. When the product of the latter two numbers (2710676) is treated as a ciphertext and decrypted, the resulting plaintext (984) is the product of the original two plaintexts.

The public key is (e, n)	$(17, 2773)$
The private key is (d)	(157)

	Encrypt M_1	*Encrypt* M_2	*Decrypt* $(C_1 \cdot C_2)$
Express your message as an integer M, where $0 < M < n$:	**41**	**24**	
Encryption: Ciphertext, $C = M^e \bmod n$	1244	2179	
Multiply the two ciphertexts:	$1244 \cdot 2179 = \mathbf{2710676}$		**2710676**
Decryption: Plaintext, $M = C^d \bmod n$			**984**
			$\mathbf{41 \cdot 24 = 984}$