Lesson 18

Zero Knowledge Proofs

Context

"Human dignity demands that personal information, like medical and forensic data, be hidden from the public. But veils of secrecy designed to preserve privacy may also be abused to cover up lies and deceit by institutions entrusted with Data, unjustly harming citizens and eroding trust in central institutions." - Starkware

"ZK gives out similar vibe as ML. More and more people just mention ZK as a magic solution that fixes everything with no context of its current limitation." - 0xMisaka

Introductory Maths

Numbers

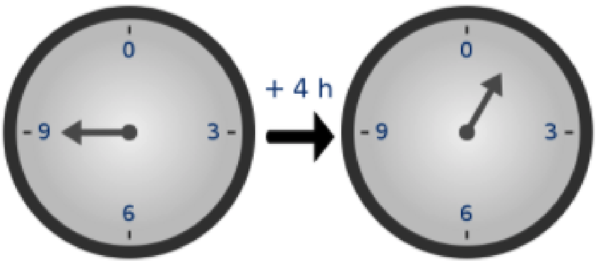
The set of Integers is denoted by $\mathbb{Z}$ e.g. {⋯,−4,−3,−2,−1,0,1,2,3,4,⋯}
The set of Rational Numbers is denoted by $\mathbb{Q}$ e.g. $\{...1, \frac{3}{2}, 2, \frac{22}{7} ...\}$
The set of Real Numbers is denoted by $\mathbb{R}$ e.g. $\{2, -4, 613, \pi, \sqrt{2}, ...\}$

Fields are denoted by $\mathbb{F}$, if they are a finite field or $\mathbb{K}$ for a field of real or complex numbers we also use $\mathbb{Z}_p^*$ to represent a finite field of integers mod prime p with multiplicative inverses.

We use finite fields for cryptography, because elements have “short”, exact representations and useful properties.

Modular Arithmetic

See this introduction



Because of how the numbers "wrap around", modular arithmetic is sometimes called "clock math"

When we write n mod k we mean simply the remainder when n is divided by k. Thus
25 mod 3 = 1
15 mod 4 = 3
The remainder should be positive.

Fields

A field is a set of say Integers together with two operations called addition and multiplication.
One example of a field is the Real Numbers under addition and multiplication, another is a set of Integers mod a prime number with addition and multiplication.

Intuitive grasp of Zero Knowledge Proofs

Introduction

It is difficult to find zero knowledge resources that avoid the extremes of either over simplifying the subject, or presenting so much mathematical detail that the reader gets bogged down and loses interest.

We start with some examples to show how zero knowledge proofs can proceed, and the situations where they could be used.

What is a zero knowledge proof

A loose definition
It is a proof that there exists or that we know something, plus a zero knowledge aspect, that is the person verifying the proof only gains one piece of information - that the proof is valid or invalid.

Actors in a Zero Knowledge Proof System

- Creator - optional, maybe combined with the prover
- Prover - I will call her Peggy
- Verifier - I will call him Victor

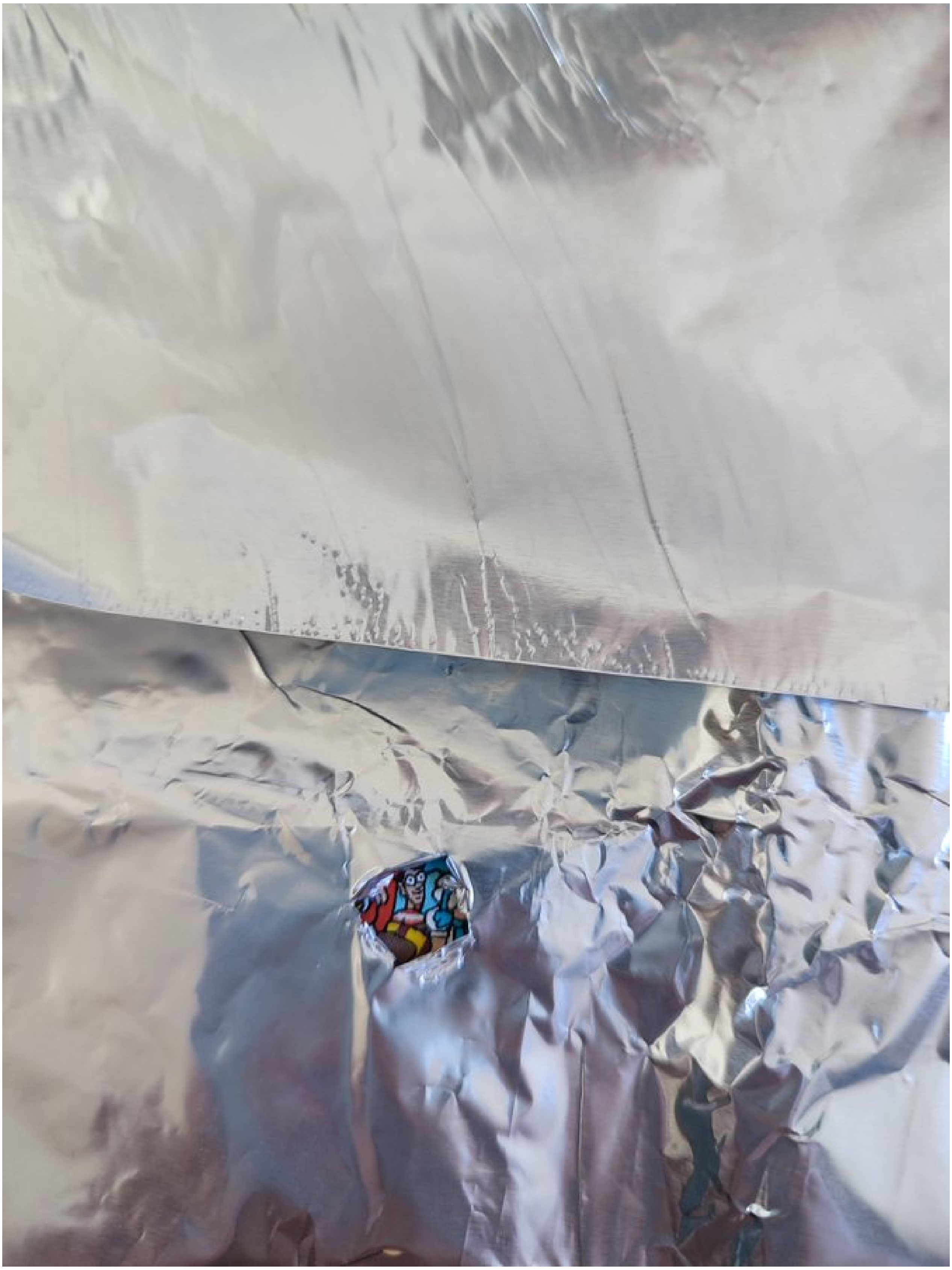
Examples to give an Intuitive grasp of zero-knowledge proofs

1. Colour blind verifier
   This is an interactive proof showing that the prover can distinguish between a red and a green billiard ball, whereas the verifier cannot distinguish them.

   ```
   • The prover wants to show the verifier that they have
   different colours but does not want him to learn which is
   red and which is green.

   • Step 1: The verifier takes the balls, each one in each
   hand, holds them in front of the prover and then hides them
   behind his back. Then, with probability 1/2 either swaps
   them (at most once) or keeps them as they are. Finally, he
   brings them out in front.

   • Step 2: The prover has to say the verifier switched them
   or not.

   • Step 3: Since they have different colours, the prover can
   always say whether they were switched or not.
   But, if they were identical (the verifier is inclined to
   believe that), the prover would be wrong with probability
   1/2.

   • Finally, to convince the verifier with very high
   probability, the prover could repeat Step 1 to Step 3 k
   times to reduce the probability of the prover being
   successful by chance to a extremely small amount.
   ```

2. Wheres Wally
   Based on the pictures of crowds where Wally is distinctively dressed, the aim being to find him within a sea of similar people.
   The proof proceeds as follows :

> Imagine the Peggy has found Wally in the picture and wants to prove this fact to Victor, however if she just shows him, Victor is liable to cheat and claim he also found Wally.
> In order to prove to Victor that she has indeed found Wally, without giving away his location in the picture
>
> ```
> 1. Peggy cuts a hole in a (very) large sheet of paper, the hole should be the exact shape of Wally in the underlying picture.
> 2. Peggy places the paper sheet over the original picture, so that the location of the picture beneath the paper is obscured.
> 3. Victor can then see throught he hole that Wally has indeed been found, but since the alignment with the underlying picture cannot be seen, he doesn't gain any information about the location of Wally.
> ```



> Quote from Vitalik Buterin
> "You can make a proof for the statement "I know a secret number such that if you take the word ‘cow', add the number to the end, and SHA256 hash it 100 million times, the output starts with `0x57d00485aa`". The verifier can verify the proof far more quickly than it would take for them to

> run 100 million hashes themselves, and the proof would also not reveal what the secret number is."

Zero Knowledge Proof Timeline

Changes have occurred because of

- Improvements to the cryptographic primitives (improved curves or hash functions for example)
- A fundamental change to the approach to zero knowledge
  See the excellent blog post from Starkware :
  The Cambrian Explosion

1984 : Goldwasser, Micali and Rackoff - Probabilistic Encryption.
1989 : Goldwasser, Micali and Rackoff - The Knowledge Complexity of Interactive Proof Systems
1991 O Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing but their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. Preliminary version in 1986. (Graph colouring problem)

....

2006 Groth, Ostrovsky and Sahai introduced pairing-based NIZK proofs, yielding the first linear size proofs based on standard assumptions.

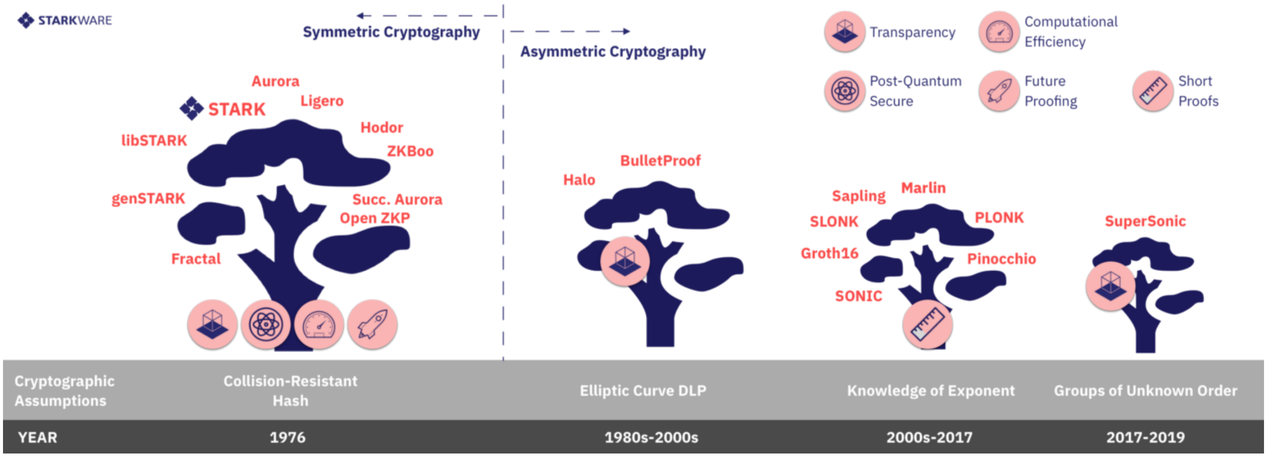
2010 Groth combined these techniques with ideas from interactive zero-knowledge arguments to give the first constant size NIZK arguments.

2016 : Jens Groth - On the Size of Pairing-based Non-interactive Arguments

From Matthew Green

Prior to Goldwasser et al., most work in this area focused the soundness of the proof system. That is, it considered the case where a malicious Prover attempts to 'trick' a Verifier into believing a false statement. What Goldwasser, Micali and Rackoff did was to turn this problem on its head. Instead of worrying only about the Prover, they asked: what happens if you don't trust the Verifier?

ZKP Ecosystem

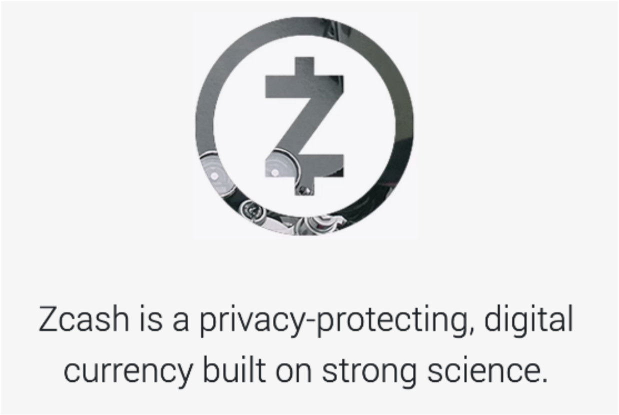


from

[The Cambrian Explosion]()

ZKP Use Cases

Privacy preserving cryptocurrencies



Zcash is a privacy-protecting, digital currency built on strong science.

Also Nightfall , ZKDai

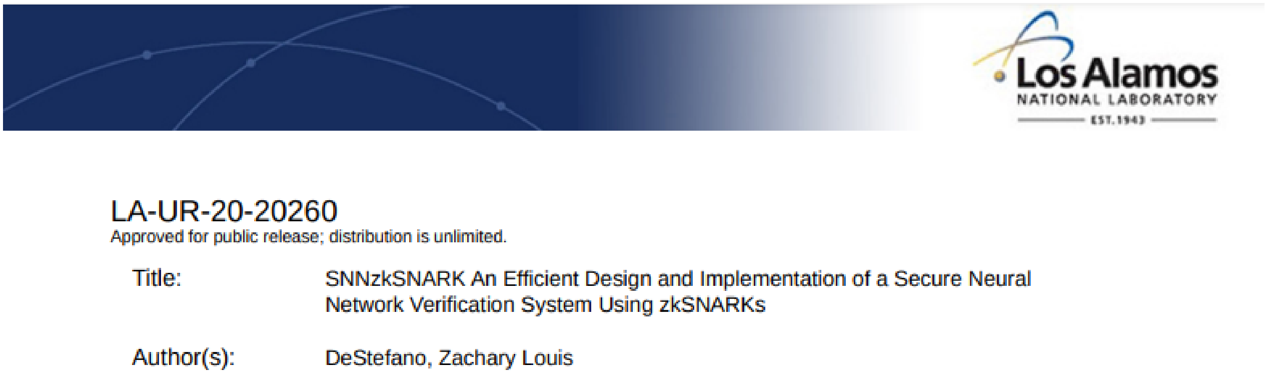
Blockchain Scalability

For example
Rollups on Ethereum

"The scalability of ZK rollup will increase by up to 4x, pushing theoretical max TPS of such systems well over 1000." - Vitalik

ZkSync
ZK Sync is designed to bring a VISA-scale throughput of thousands of transactions per second to Ethereum.

Nuclear Treaty Verification

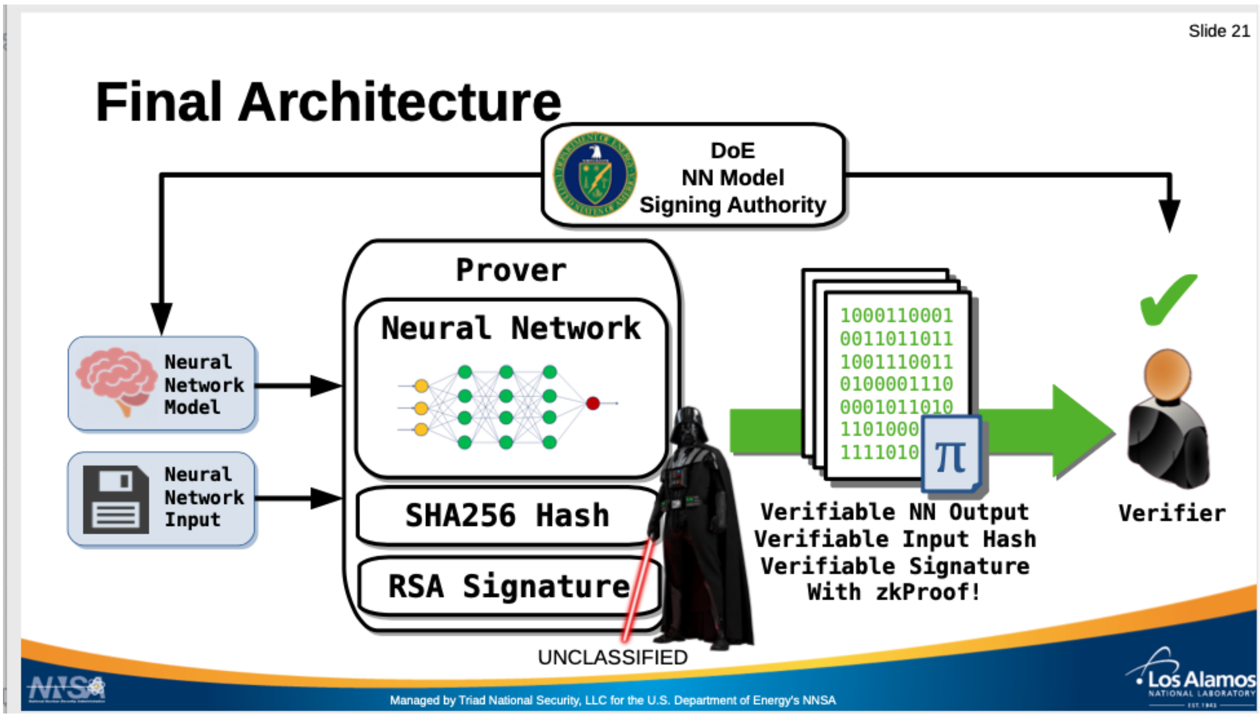


Los Alamos NATIONAL LABORATORY EST.1943

LA-UR-20-20260
Approved for public release; distribution is unlimited.

Title: SNNzkSNARK An Efficient Design and Implementation of a Secure Neural Network Verification System Using zkSNARKs

Author(s): DeStefano, Zachary Louis



Slide 21

Final Architecture

DoE
NN Model
Signing Authority

Neural
Network
Model

Neural
Network
Input

Prover

Neural Network

SHA256 Hash

RSA Signature

1000110001
0011011011
1001110011
0100001110
0001011010
1101000
1111010

π

✓

Verifier

Verifiable NN Output
Verifiable Input Hash
Verifiable Signature
With zkProof!

UNCLASSIFIED

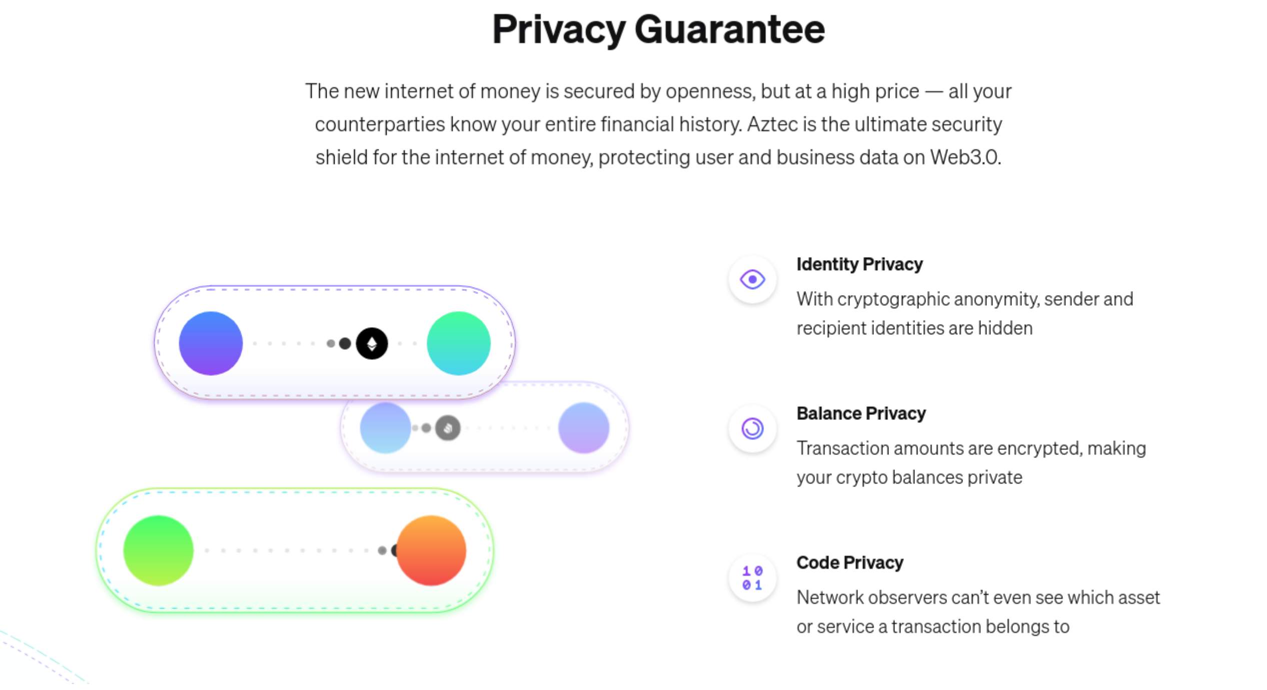
NNSA

Managed by Triad National Security, LLC for the U.S. Department of Energy's NNSA

Los Alamos
NATIONAL LABORATORY

Privacy Preserving Financial Systems

Aztec



Privacy Guarantee

The new internet of money is secured by openness, but at a high price — all your counterparties know your entire financial history. Aztec is the ultimate security shield for the internet of money, protecting user and business data on Web3.0.

Identity Privacy

With cryptographic anonymity, sender and recipient identities are hidden

Balance Privacy

Transaction amounts are encrypted, making your crypto balances private

Code Privacy

Network observers can't even see which asset or service a transaction belongs to

ZK Proofs in more detail - zkSNARKS

The process of creating and using a zk-SNARK can be summarised as

The Creator takes a secret parameter lambda and a program C, and generates two publicly available keys:

- a proving key pk
- a verification key vk

These keys are public parameters that only need to be generated once for a given program C. They are also known as the Common Reference String.

The prover Peggy takes a proving key pk, a public input x and a private witness w.
Peggy generates a proof $pr = P(pk, x, w)$ that claims that Peggy knows a witness w and that the witness satisfies the program C.

The verifier Victor computes $V(vk, x, pr)$ which returns true if the proof is correct, and false otherwise.
Thus this function returns true if Peggy knows a witness w satisfying

$C(x, w) = true$

Trusted Setups and Toxic Waste

Note the secret parameter lambda in the setup, this parameter sometimes makes it tricky to use zk-SNARK in real-world applications.
The reason for this is that anyone who knows this parameter can generate fake proofs.
Specifically, given any program C and public input x a person who knows lambda can generate a proof $pr2$ such that $V(vk, x, pr2)$ evaluates to true **without** knowledge of the secret w.

Interactive v Non Interactive Proofs

Non-interactivity is only useful if we want to allow multiple independent verifiers to verify a given proof without each one having to individually query the prover.

In contrast, in non-interactive zero knowledge protocols there is no repeated communication between the prover and the verifier. Instead, there is only a single "round", which can be carried out asynchronously. Using publicly available data, Peggy generates a proof, which she publishes in a place accessible to Victor (e.g. on a distributed ledger). Following this, Victor can verify the proof at any point in time to complete the “round”. Note that even though Peggy produces only a single proof, as opposed to multiple ones in the interactive version, the verifier can still be certain that except for negligible probability, she does indeed know the secret she is claiming.

Succinct v Non Succinct

Succinctness is necessary only if the medium used for storing the proofs is very expensive and/or if we need very short verification times.

Proof v Proof of Knowledge

A proof of knowledge is stronger and more useful than just proving the statement is true. For instance, it allows me to prove that I know a secret key, rather than just that it exists.

Argument v Proof

In a proof, the soundness holds against a computationally unbounded prover and in an argument, the soundness only holds against a polynomially bounded prover.
Arguments are thus often called "computationally sound proofs".

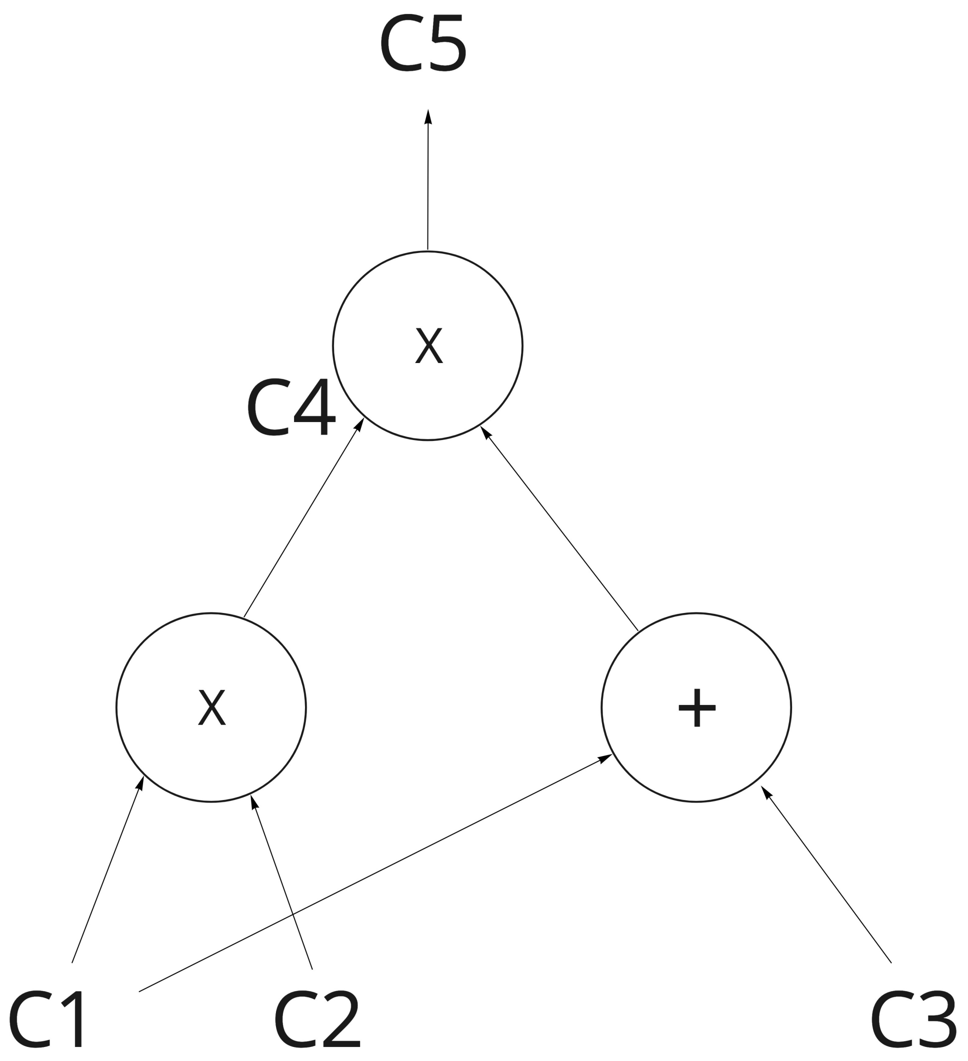
The Prover and the Verifier have to agree on what they’re proving. This means that both know the statement that is to be proven and what the inputs to this statement represent.

zkSNARK stands for **z**ero **k**nowledge **S**uccinct **N**on interactive **A**rgument of **K**nowledge. The features of succinctness (they are small in size and the amount of computation required) and Non Interaction (a single step is sufficient to complete the proof) has helped their adoption in applications.

Simplified Overview of the process needed to create a zkSNARK

1. Trusted Setup
   ZKSNarks require a one off set up step to produce prover and verifier keys. This step is generally seen as a drawback to zkSNARKS, it requires an amount of trust, if details of the setup are later leaked it would be possible to create false proofs.

2. A High Level description is turned into an arithmetic circuit
   The creator of the zkSNARK uses a high level language to spcify the algorithm that constitutes and tests the proof.
   This high level specification is compiled into an arithmetic circuit.
   An arithmetic circuit can be thought of as similar to a physical electrical circuit consisting of logical gates and wires. This circuit contrains the allowed inputs that will lead to a correct proof.



3. Further Mathematical refinement
   The circuit is then turned into a series of formulae called a Quadratic Arithmetic Program (QAP).
   The QAP is then further refined to ensure the privacy aspect of the process.
   The end result is a proof in the form of series of bytes that is given to the verifier. The verifier can pass this proof through a verifier function to receive a true or false result.
   There is no information in the proof that the verifier can use to learn any further information about the prover or their witness.

Real Life ZKP choices

- We don't always need snarks,

Problems

There maybe a problem trusting the witness , so how do we know that the witness value is true in a real world sense, we may need a combination of other sites and oracles

Other technology

- Decentralised Identifiers
- Homomorphic Encryption
- Verifiable Random Functions
- MACI
- Threshold Cryptography / Secret Sharing / MPC

Decentralised Identifiers

W3 standard :
DIDs are URLs that associate a DID subject with a DID document allowing trustable interactions associated with that subject. Each DID document can express cryptographic material, verification methods, or service endpoints, which provide a set of mechanisms enabling a DID controller to prove control of the DID.

Service endpoints enable trusted interactions associated with the DID subject. A DID document might contain semantics about the subject that it identifies.

A DID document might contain the DID subject itself (e.g. a data model).**

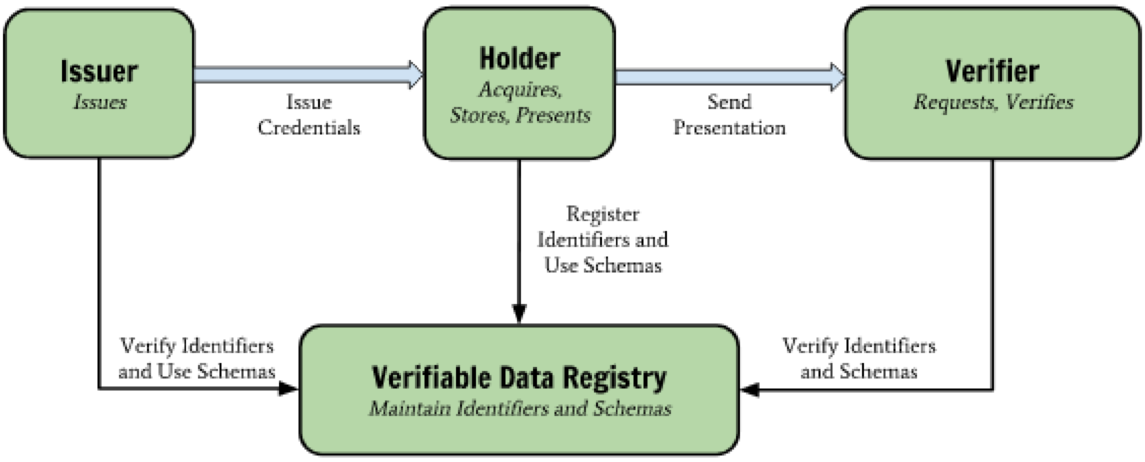

Issuer
Issues
Issue Credentials
Holder
Acquires, Stores, Presents
Send Presentation
Verifier
Requests, Verifies
Register Identifiers and Use Schemas
Verify Identifiers and Use Schemas
Verifiable Data Registry
Maintain Identifiers and Schemas
Verify Identifiers and Schemas

Figure 1 The roles and information flows forming the basis for this specification.

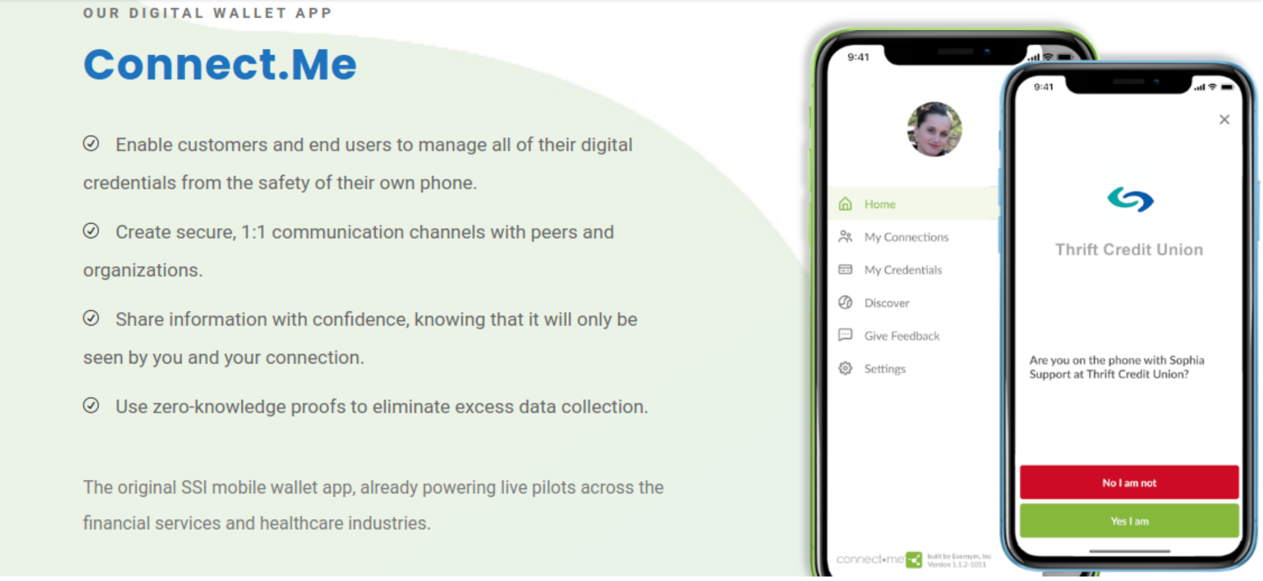
SSI Wallet and Verifiable Credentials

The Wallet contains verifiable credentials that can cryptographically prove to any verifier:

1. Who (or what) is the issuer;
2. To whom (or what) it was issued;
3. Whether it has been altered since it was issued;
4. Whether it has been revoked by the issuer.

Major Companies in this space

- Spherity (MPC from Unbound)
- Microsoft (https://didproject.azurewebsites.net/docs/overview.html)
- uPort
- Civic
- Evernym (Sovrin blockchain)



OUR DIGITAL WALLET APP

Connect.Me

✓ Enable customers and end users to manage all of their digital credentials from the safety of their own phone.

✓ Create secure, 1:1 communication channels with peers and organizations.

✓ Share information with confidence, knowing that it will only be seen by you and your connection.

✓ Use zero-knowledge proofs to eliminate excess data collection.

The original SSI mobile wallet app, already powering live pilots across the financial services and healthcare industries.

Are you on the phone with Sophia Support at Thrift Credit Union?

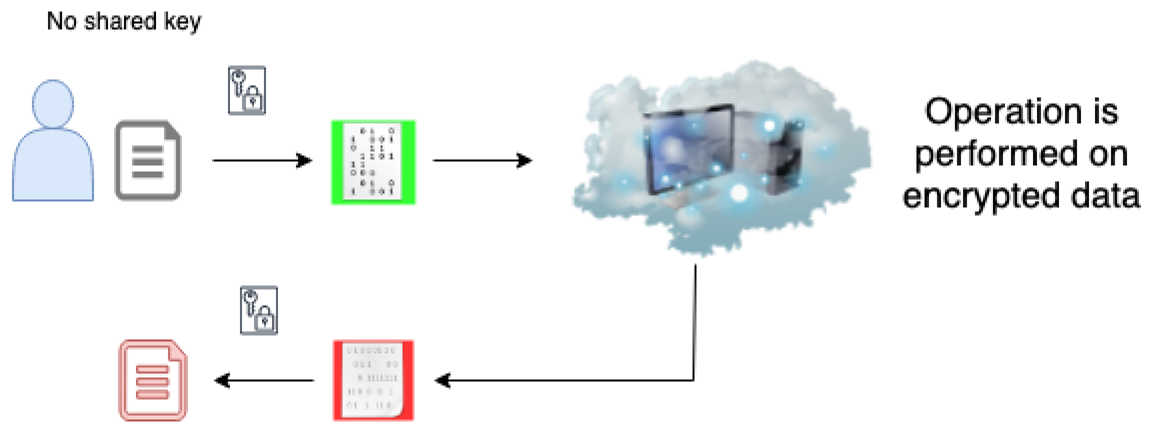
KYC providers

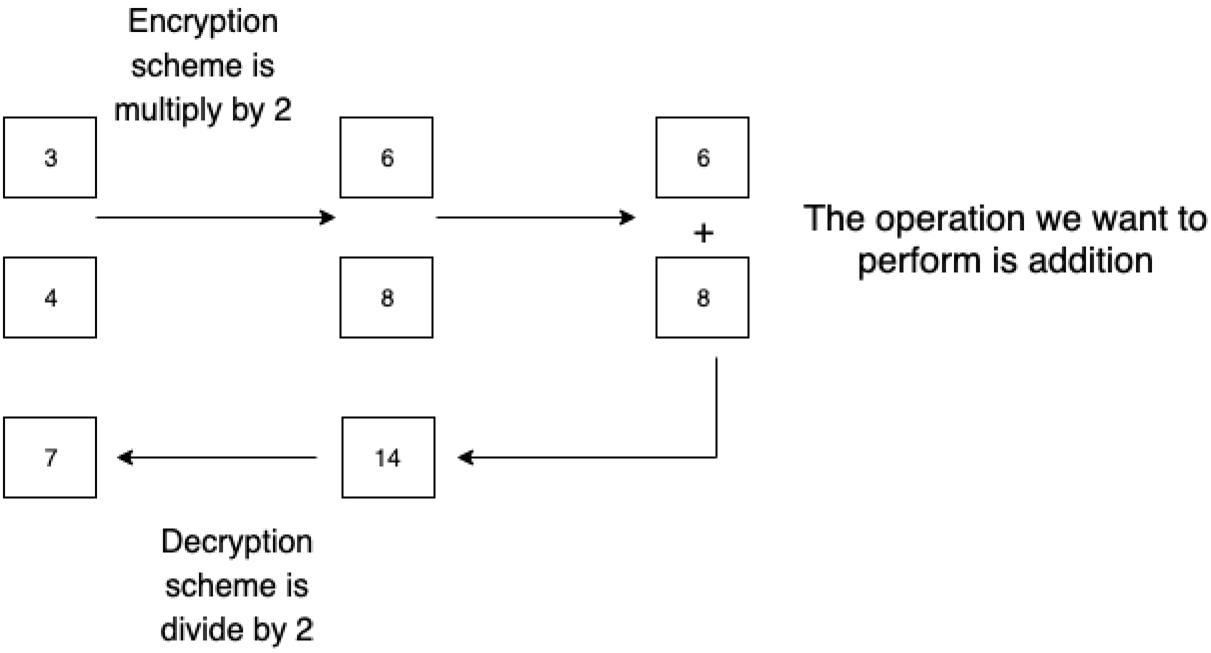
- SecureKey (https://securekey.com/partner-directory/)
  Notable partners: Hyperledger, Intel
- Opus (www.opus.com)
  Notable partners: Experian, Financial Times
- FICO (https://www.fico.com)
  Notable partners: Honeywell, Thames Water, Santander
- Onfido (https://onfido.com/)
- Yoti (https://www.yoti.com/)

(Fully) Homomorphic Encryption

Fully Homomorphic Encryption , the ‘holy grail’ of cryptography, is a form of encryption that allows arbitrary computations on encrypted data.

Homomorphic encryption is a form of encryption with an additional evaluation capability for computing over encrypted data without access to the secret key. The result of such a computation remains encrypted. Homomorphic encryption can be viewed as an extension of either symmetric-key or public-key cryptography. Homomorphic refers to homomorphism in algebra: the encryption and decryption functions can be thought as homomorphisms between plaintext and ciphertext spaces.




Encryption scheme is multiply by 2
3
4
6
8
6
+
8
The operation we want to perform is addition
7
14
Decryption scheme is divide by 2

Alice, the data owner, encrypts data with her key and sends it to an outsourced machine for storage and processing.

> The outsourced machine performs arbitrary computations on the encrypted data without learning anything about it.
> Alice decrypts the results of those computations using her original key, retaining full confidentiality, ownership, and control.
>
> Example :
>
> Medical Data using FHE

Verifiable Random Functions

From Algorand VRFs :

A Verifiable Random Function (VRF) is a cryptographic primitive that maps inputs to verifiable pseudorandom outputs. VRFs were Introduced by Micali, Rabin, and Vadhan in '99.

Given an input value x, the knowledge of the secret key SK allows one to compute $y = F_{SK}(x)$ together with the proof of correctness π_x. This proof convinces every verifier that the value $y = F_{SK}(x)$ is indeed correct with respect to the public key of the VRF. We can view VRFs as a commitment to a number of random-looking bits

The owner of a secret key can compute the function value as well as an associated proof for any input value. Everyone else, using the proof and the associated public key or verification key can check that this value was indeed calculated correctly, yet this information cannot be used to find the secret key.

Algorand have released it as an extension to the libsodium library

Such functions are ideal to find block producers in a blockchain in a trustless verifiable way.

Voting Systems

A Problem with rewarding users in voting systems

Imagine a system where users can vote with tokens (which they retain) and are rewarded for the number of votes they receive in a period.

Suppose that some wealthy user acquires some quantity N of tokens, and as a result each of the user's k votes gives the recipient a reward of $N \cdot q$ (q here probably being a very small number, eg. think $q = 0.000001$).
The user simply upvotes their own sockpuppet accounts, giving themselves the reward of $N \cdot k \cdot q$. Then each user has an "interest rate" of $k \cdot q$ per period.

See collusion article and Governance by Vitalik

Bribery in voting Systems

Suppose Alice can vote for a project to receive a grant.
If Charlie has a candidate project he may want to bribe Alice to vote for his project, he could do this via a side channel, and it would be unknown to the voting system.

A partial way around this is to encrypt the votes, so that if Alice's vote is seen in the system, we cannot tell which project she voted for.
To do this Alice could use some key, however if the encrypted vote is public, Alice could send the details of how she voted to Charlie, who could verify it , and Alice could claim her bribe.
A further refinement is then to allow Alice to vote multiple times, revoking the previous key she used, effectively invalidating the previous vote.
In this case Charlie loses the confidence that he has in the information that Alice sends him, as he knows she could have accepted his bribe, then later voted for someone else (and maybe get a bribe from them etc.)
MACI uses this approach, plus ZKPs to create an infrastructure that mitigates the effect of collusion.

Minimal Anti-Collusion Infrastructure

See Repo
Discussion
Zero Knowledge Podcast

The process of implementing this in a smart contract
From Introduction

Whitelisted voters named Alice, Bob, and Charlie register to vote by sending their public key to a smart contract. Additionally, there is a central coordinator Dave, whose public key is known to all.

When Alice casts her vote, she signs her vote with her private key, encrypts her signature with Dave's public key, and submits the result to the smart contract.

Each voter may change her keypair at any time. To do this, she creates and signs a key-change command, encrypts it, and sends it to the smart contract. This makes it impossible for a briber to ever be sure that their bribe has any effect on the bribee's vote.

If Bob, for instance, bribes Alice to vote a certain way, she can simply use the first public key she had registered — which is now void — to cast a vote. Since said vote is encrypted, as was the key-changing message which Alice had previously sent to Dave, Bob has no way to tell if Alice had indeed voted the way he wanted her to.

Even if Alice reveals the cleartext of her vote to Bob, she just needs to not show him the updated key command that she previously used to invalidate that key. In short, as long as she had submitted a single encrypted command before her vote, there is no way to tell if said vote is valid or not.

Threshold Cryptosystems / Secret Sharing / Multiparty Computation

The goal is to divide secret S into n pieces of data $S_i..S_n$ in such a way that:

Knowledge of any k or more S_i pieces makes S easy to compute. That is, the complete secret S can be reconstructed from any combination of k pieces of data.
Knowledge of any $k-1$ or fewer S_i pieces leaves S completely undetermined, in the sense that the possible values for S seem as likely as with knowledge of 0 pieces.

A naive splitting of a key would just make a brute force attack easier.

Secret Sharing

Shamir Secret Sharing

Properties of Shamir's (k, n) threshold scheme are:

- Secure: Information theoretic security.
- Minimal: The size of each piece does not exceed the size of the original data.
- Extensible: When k is kept fixed, D_i pieces can be dynamically added or deleted without affecting the other pieces.
- Dynamic: Security can be easily enhanced without changing the secret, but by changing the polynomial occasionally (keeping the same free term) and constructing new shares to the participants.
- Flexible: In organizations where hierarchy is important, we can supply each participant different number of pieces according to their importance inside the organization. For instance, the president can unlock the safe alone, whereas 3 subordinates are required together to unlock it.

Multiparty computation overview

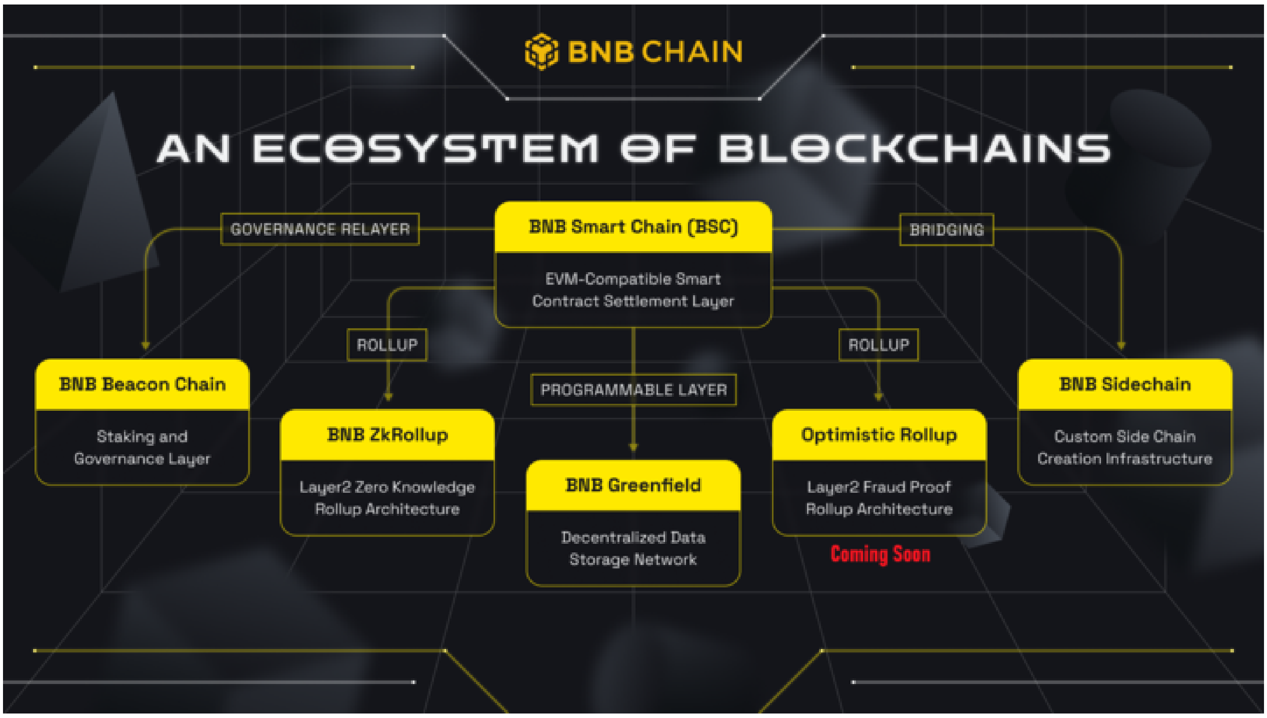
A key point to understand is that MPC is not a single protocol but rather a growing class of solutions that differ with respect to properties and performance. However, common for most MPC systems are the three basic roles:

- The Input Parties delivering sensitive data to the confidential computation.
- The Result Parties receiving results or partial results from the confidential computation.
- The Computing Parties jointly computing the confidential computation

Demonstration of Zokrates

Using ZKPs

BNB Ecosystem


BNB CHAIN
AN ECOSYSTEM OF BLOCKCHAINS
GOVERNANCE RELAYER
BNB Smart Chain (BSC)
EVM-Compatible Smart Contract Settlement Layer
BRIDGING
ROLLUP
ROLLUP
BNB Beacon Chain
Staking and Governance Layer
BNB ZkRollup
Layer2 Zero Knowledge Rollup Architecture
PROGRAMMABLE LAYER
BNB Greenfield
Decentralized Data Storage Network
Optimistic Rollup
Layer2 Fraud Proof Rollup Architecture
Coming Soon
BNB Sidechain
Custom Side Chain Creation Infrastructure

zkBNB

Introduction

ZkBNB is founded on the Zero Knowledge (ZK) Rollup framework, which is a Layer-2 solution that enables computations and state alterations to occur off-chain, or on a sidechain. In this structure, a summary of the modifications and accompanying cryptographic proofs for validating these adjustments are submitted to the Mainnet.

ZkBNB, similar to zkRollups, has the ability to consolidate hundreds of transactions into a single off-chain group (Rollup Block) and produce a cryptographic verification. These verifications can take the form of SNARKs (succinct non-interactive argument of knowledge), which can authenticate the legitimacy of each transaction within the Rollup Block. This approach guarantees that all funds remain on the BSC while computation and storage take place on BNB Sidechains, resulting in reduced costs and increased speed. Moreover, due to the implementation of zk-SNARK proofs, ZkBNB maintains the same level of security as the BNB Smart Chain.

zkBNB Key Features

1. Equivalent L1 Security: ZkBNB maintains the same level of security as BSC, ensuring cryptographic security through the use of zkSNARK proofs. Users can rely on this without having to monitor Rollup blocks or trust third parties to prevent fraud.
2. Smooth L1-L2 Interaction: BNB and BEP20/BEP721/BEP1155 tokens created on BSC or ZkBNB can move effortlessly between BSC and ZkBNB.
3. Integrated AMM (Automated Market Maker) swap: ZkBNB supports permissionless and automatic trading of digital assets using built-in liquidity pools.
4. Native NFT marketplace: Developers can easily create marketplaces for crypto collectibles and non-fungible tokens (NFTs) on ZkBNB.
5. Rapid transaction speed and quicker finality: BNB Smart Chain prioritises performance, and ZkBNB delivers impressive results by supporting 100 million addresses and processing up to 10,000 transactions per second (TPS) – unparalleled in the blockchain industry.
6. Gas Tokens: ZkBNB allows the use of either BEP20 or BNB as gas tokens, with fees up to 10x lower.
7. "Full exit" on BSC: If a user believes their transactions are being censored by ZkBNB, they can request a "full exit" at any time to withdraw their funds, ensuring continuous access to their assets.

Automated Proof of Reserves

In November 2022, Binance introduced its Proof of Reserves system, which employs Merkle tree cryptography, enabling users to confirm their holdings.

This approach had 2 possible drawbacks

1. To protect user privacy, leaf nodes in Merkle proof represented the hash of users’ holdings – thus, the Merkle root couldn’t reflect the sum of its leaf nodes’ balance information.
2. The user whose reserves were being verified could potentially add a negative balance under a fake account somewhere in the tree to make the total required reserves appear smaller.

Binance has since enhanced this system by incorporating zk-SNARKs.

This upgrade now allows users to privately and securely verify that every account has a non-negative total net balance and that all user assets contribute to Binance's declared overall net balance of user assets.

Details of the upgrade

A bug was found, see Hacken Discovery
A missing range check would allow a very large value to be used to create a 'false' proof. This has since (late Feb 2023) been fixed.