Backscatter Network

Andrew Klitzke
https://backscatter.network/

November 10th, 2023

Abstract

This paper addresses the challenge of enabling businesses that handle sensitive data to operate as decentralized autonomous organizations (DAOs) without reliance on trusted third parties or permissioned networks. Blockchain, homomorphic encryption, and zero-knowledge proofs provide part of the solution, but together still require some mechanism for decryption. This work proposes a solution to the problem by sharing a single secret across a network of parties that collectively store, analyze, and reveal sensitive data. Data is encrypted for this network secret, and can only be analyzed or decrypted if the participants collectively agree to do so. The secret is created using distributed key generation to prevent single points of failure. Parties cannot individually view the data they are deciding on, and the conditions under which data can be read or analyzed is programmed with smart contracts. Economic mechanisms are used to keep parties honest. Lastly, parties can leave and join the network as needed to provide robustness and security.

1 Introduction

Commerce on the Internet has come to rely almost exclusively on buying, analyzing, and selling sensitive user data. Search engines, social media, advertisers, and other tech companies often have unrestricted access to the habits, interests, and lifestyle of their users. While this works well to provide low-cost and personalized services, data leaks, conflicts of interest, and intrusions of privacy are common. Authoritarian regimes have used user data for repression and violence. Controls can be put in place by governments, but they have had a hard time enforcing restrictions on powerful tech companies. Once an organization possesses user data, no mechanism exists to hold them accountable for how that data is utilized.

Cryptocurrencies and blockchain have shown their value as a way to govern the flow of money and assets without relying on a government-backed court system. Miners can be thought of as voters, voting along a set of agreed-upon rules. With enough votes, assets are transferred between parties. From this has emerged an ecosystem of blockchain-based businesses known as DAO's whose contracts with customers, investors, and employees are enforced through blockchain's smart contracts rather than a traditional government's court system. In order to know how to vote miners need to be able to see all data that

is part of a smart contract. This forces information a DAO interacts with to be public. This works for business models that can operate entirely in the public eye, like trading platforms [1] or currencies [2]. However, many real-world businesses must regularly interact with sensitive private data. For example e-commerce must manage home addresses and tracking numbers while social media must manage private pictures and posts. This data cannot exist fully in the public eye. Without the ability to control private data, blockchain is limited in its ability to be applied to many common business models.

What is needed is the ability to vote not just on who owns an asset but how some sensitive data is analyzed or read. Consider the example of an on-chain social media platform where all users' images are stored in a blockchain. Only if enough miners vote to say "yes, this person can see this image" can a user then see a picture of their friend stored in the blockchain. With a system like this in place: viewing, analyzing, and exchanging of sensitive data can occur only if the contracts governing usage of the data allow for it. This whitepaper presents a method of combining economic incentives, emerging cryptographic technologies, and blockchain to create a system that would allow real-world businesses that manage sensitive data to operate as a DAO.

2 Prerequisites

Assume a collection of participants similar to miners that have available:

1. Secure connections to all other parties
2. Blockchain (like Ethereum) to use for economic incentives, coordination, and governance
3. Reliable Broadcast to all other parties
4. Fully Homomorphic Encryption + Decryption
5. Public-key Infrastructure (each parties public key is known by every other party)
6. Zero-knowledge Proofs (the ability to prove an algorithm of arbitrary complexity)
7. Distributed Key Generation (the parties can use the tools above to collectively generate a single shared key, with each party receiving a share)

3 Construction

At its most basic, a committee of participants controls a single, homomorphic private key known as the network secret. Each party is granted a share of the network secret using a secret-sharing algorithm. If enough parties collaborate to utilize their shares for some operation like decryption, it successfully occurs.

3.1 Storing Data

Anyone wishing to manage data using this network encrypts it for the current network public key and one party reliably broadcasts it to all parties. Parties are expected to store this data internally for the lifetime of their committee appointment. The ciphertext of encrypted data is generally considered 'public', similar to the ledger in a blockchain.

3.2 Reading Data

The party requesting to read data reliably broadcasts the request along with a public key to receive the data. If approved, each party performs on the original data first a homomorphic encryption operation for the receiving public key, then a partial decryption using their share, and finally reliably broadcasts the partially decrypted + re-encrypted data along with a zero-knowledge proof showing the message is valid.

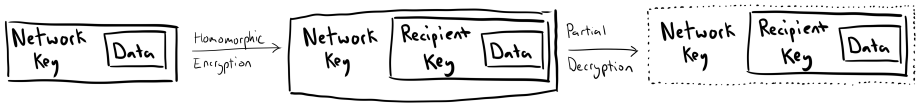

Figure 1: Data, encrypted for the network key, homomorphically re-encrypted for the recipient then partially decrypted.

Once enough parties have reliably broadcast their partially decrypted share, these pieces can be assembled into a single ciphertext containing the data encrypted under the receiving public key.

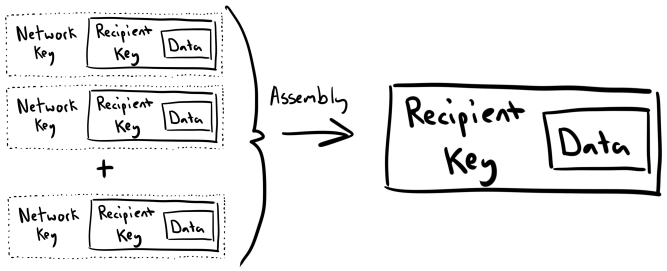

Figure 2: Several partially decrypted shares being assembled into the ciphertext encrypted for the receiving public key.

The assembled re-encrypted data is generally considered public. Parties are expected to store it, but can forget the individual shares after assembly. The number of parties required to post their partially decrypted share, known as the threshold, is governed by the network and set when the committee is created.

3.3 Approving Reads

Blockchain smart contracts are used to provide a publicly auditable set of read approval conditions. Approval conditions are specified when the data is first stored in the network. Each party in a committee can inspect the blockchain to determine whether or not a requesting party is authorized to read data from the network. Smart contracts are Turing complete and can be arbitrarily complex, and so approval conditions can also be. Valid examples are: "anyone who pays 5 dai", "only specific addresses", "after 2 years has elapsed", "licensed medical doctors", etc.

3.4 Homomorphism

Using a homomorphic encryption algorithm allows the network to grant access to not just the raw data but to the result of an arbitrary algorithm run on the data. Imagine a long list of residential addresses stored in this system. A DAO could be approved to know the number of addresses that reside within the USA without being approved to read the individual addresses themselves. To do so, an algorithm to detect and tally USA addresses is run homomorphically on the ciphertext. The result of this algorithm will still be encrypted by the network secret. This result is then treated in the same manner as a normal read: parties use their shares to collectively re-encrypt the result for the DAO.

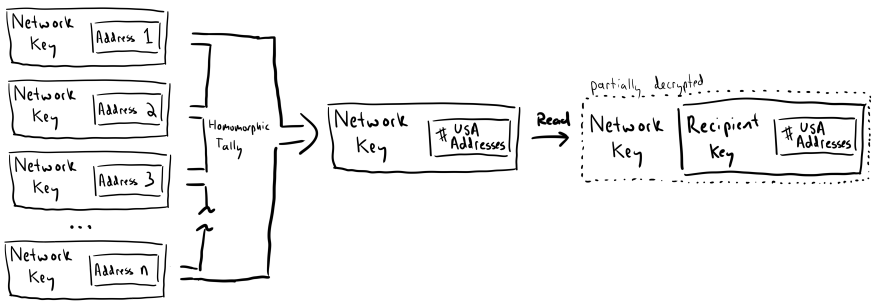

Figure 3: A single party performing a homomorphic tally and partial re-encryption.

These shares are reconstituted just like a read, allowing the DAO to see the number of USA addresses without any party knowing any underlying addresses. Any arbitrary algorithm or approval conditions could be supported, meaning complex analytics or data mining can be run and approved on a case-by-case basis. When the read request is broadcast to the committee, the requestor includes the algorithm to be computed. Assuming the approval conditions allow the requestor to read the result of that algorithm, each party homomorphically

computes the algorithm on the ciphertext. The parties treat the result like data to be read: re-encrypting, partially decrypting, and reliably broadcasting the final result.

3.5 Share Rotation

To facilitate parties leaving and joining the network, the committee and network secret must be periodically rotated. Individuals can be expected to leave the network after some time, and as the system grows new parties need to be able to join and participate. To perform a share rotation:

1. Parties that make up the new committee perform a distributed key generation, creating the next network secret.
2. The outgoing committee re-shares the current network secret with the new committee
    (a) The outgoing committee also holds shares of the previous committee's secret, but does not share those with the new committee.
    (b) The new committee is now officially established.
        i. Incoming data can only be encrypted under the newest network key.
        ii. All read, write, and other operations must be directed at the new committee.
    (c) The outgoing parties:
        i. Can forget shares of the network secret they held
        ii. Can forget any data they held on behalf of the network

3.6 Economic Incentives

In addition to cryptography, economic mechanisms are needed for security and reliability guarantees. While individual parties cannot read data stored in the system, a majority could collude off-chain. There is no way to prove the deletion of shares or data in a permissionless network. Even after rotation, old shares could collude to decrypt old data. So, the higher the committee turnover, the more likely it becomes for old shares to be misused. Despite this consideration, however, parties still need to be able to leave and join the network.

Overly limiting parties' ability to leave the network could make them susceptible to fraud: if the rules don't let them cash out, a wealthy adversary could pay them off-chain. Economic mechanisms must be put in place that balance turnover, foster participation, and encourage rule-following. There are a few basic ideas:

1. The committee rotates, and a new network secret is generated, when a new committee "buys out" the current committee.

   (a) Each share in the new committee has a fixed cost, set by the network, and made up of a 'stake' component and a 'buyout' component.
   (b) Users who wish to be a party in the new committee must put up the 'stake' component as well as the 'buyout' component.
   (c) Existing committee members indicate their willingness (or not) to be bought out and forego participation in the next committee.
   (d) Once a certain percentage of current committee members indicate a desire to be bought out, a rotation is triggered.
      i. The percentage required will generally be slightly under half the network secret's threshold, so as to guarantee that the outgoing committee members do not collectively hold enough shares to decrypt their committee's data.
   (e) Following a successful rotation, the 'buyout' portion of the new committee's funds are divided amongst the outgoing parties.
      i. The more parties joining the new committee, the larger the payout for committee members leaving the network.
2. Following a successful rotation, the outgoing parties receive any funds they have staked, including any remaining portion of their 'stake' from joining, as well as any funds that may have accumulated during their committee appointment.
3. Users of the network pay committee members when storing or reading data
   (a) A portion is given immediately to committee members as a 'tip'.
   (b) A portion is added to the funds staked on behalf of committee members. At the next share rotation committee members can withdraw any excess over the 'stake' amount required to participate in the next committee.
   (c) The more users using the network, the more funds are transferred to committee members, and the lower the incentive is to be bought out.
4. Committee members are granted newly minted tokens as rewards for participating in storing and reading data.
   (a) Inflation is kept roughly proportional to the activity in the network, as the total ownership tokens in circulation divided by the 'stake' amount sets the maximum number of shares in a committee.
   (b) These rewards will not vest for several committee rotations, reducing the incentive and ability for shares to collude off-chain.
5. To incentivize honest behavior, all funds staked and exchanged are network ownership tokens (not to be confused with shares of a network secret.)
6. Non-participating parties could see their staked funds slashed or be removed from the next committee

3.7 Data Migration

At the conclusion of share rotation current committee members each hold 2 secrets: a share of the new network secret and a share of the prior committee's network secret. Data from the prior committee becomes un-decryptable after a second rotation. Anyone storing data in the system must pay to migrate data from the prior committee to the current one before the next committee rotation.

To migrate data, an approved party reliably broadcasts the migration request and pays a fee and a 'tip', similar to storing data.

Users may want to restrict who can pay to keep their data alive, and may prefer it be forgotten. Migration can be approved like reads, where a smart contract allows users to control the precise circumstances under which their migration occurs.

4 Technical Details

Distributed Key Generation, Homomorphic Encryption, and Zero-Knowledge Proofs are required to build this system. There exist in academic literature many variants with tradeoffs in security, performance, and ease of use that could be suitable for this network. As time goes on, it is likely underlying technologies will continue to be refined. While it is important that specific algorithms, parameters, and libraries receive thoughtful attention, the aim of this whitepaper is only to demonstrate the feasibility of this network. We'll highlight and delve deep into a few of the more challenging problems to showcase feasibility while leaving open the discussion of specific implementations.

4.1 Distributed Key Generation

There are several variations of distributed key generation protocols that would work to build this system with tradeoffs in security, robustness, and performance [3, 4, 5]. The basic premise of most protocols is as such:

1. Each party generates a random polynomial of order 'threshold + 1'.
2. Each party sends every other party a share of their polynomial.
3. The parties aggregate received shares to generate their final share of the network secret

The first challenge is verifiability, as in step 2 a party could send 'fake' shares, potentially disabling the network. There are a few different methods in scientific literature that solve this issue[3, 4, 5], but most require a threshold lower than ⅓ the committee size. Consider that in our system the public keys of all parties are known by all other parties. We utilize this property to propose a distributed key generation algorithm that has a committee size greater than 1/3:

1. Each party generates a random polynomial of order 'threshold + 1'
2. Each party reliably broadcasts:

(a) The share for every party, encrypted for that party.

(b) A zero-knowledge proof proving:

i. Every parties share is made up of the polynomial with those parameters

ii. Every parties share is properly encrypted for that party

iii. The share of the network secret derives from the polynomial

(c) Enough information about the share of the network secret to suffice for the proof, perhaps a public key derived from the secret or a hash of the secret

(d) A signature over the above data

3. Each node verifies this data during their participation in the reliable broadcast.

4.2 Homomorphic Secret Sharing

Most modern homomorphic secret sharing algorithms break down when reconstituting a secret using plain shamir's secret sharing. Modern homomorphic encryption algorithms are generally based on the learning-with-errors problem. An error is incorporated into ciphertext and public keys. When reconstituting a secret that has been shared, these errors are additive and plaintexts are quickly corrupted from the error. This problem is solved using a technique similar to key-switching or bootstrapping, allowing us to 'reset' the error in the ciphertext. See Section 3 in [6] for a detailed explanation of this operation.

4.3 Zero-Knowledge Proofs

There are several variants of zero-knowledge proofs that could suffice for this project, including ZK-STARK's [7], Bulletproofs [8], and others. The algorithms needing to be proven are relatively deep, including learning-with-errors decryption with bootstrapping. Care will need to be paid to ensure these proofs are performant. With that said, there are examples in both scientific literature and in practice that use proofs over similar algorithms [9].

4.4 Performance

Several cited papers contain benchmarks that can give a rough idea of the performance of this system. Many calculations performed by parties would not be trivial, and could take several seconds [3, 9] or even minutes [6] to compute on modern hardware, especially with large committees. Performance has implications in:

- The amount of time it takes to complete a single operation like a read
- The bandwidth, or number of operations the network can complete per hour

- The cost to users of storing, analyzing, and reading data

Early iterations of this network are going to be significantly slower and more expensive than traditional methods of managing sensitive data, like a corporate-owned database. In the same way that ethereum is rarely used when shopping in-person at retailers, it is unlikely this system would be used on-demand as one browses a social media website. Initial applications would be limited to use cases that can handle the expense and latency.

5 Conclusion

This whitepaper has proposed a system that would allow real-world businesses that manage sensitive data to function as a DAO. It starts with a permissionless blockchain to provide smart contract and reliable broadcast mechanisms, but with no support for sensitive data. This work proposes a method to encrypt, analyze, and distribute sensitive data according to rules specified by a blockchain's smart contracts. It includes a way to add and remove parties from the network using economic incentives that promise stability and security. It allows real-world businesses like ecommerce, social media, and more to function as an on-chain DAO. Buying, selling, and analyzing data can only occur if owners of the data explicitly opt-in, and there is inherent transparency and control to users over what organizations can do with sensitive data.

References

[1] H. Adams, et. al, "Uniswap v3 Core," 2021

[2] "The Dai Stablecoin System," 2017

[3] S. Das, et. al, "Practical Asynchronous Distributed Key Generation," 2021

[4] S. Das, Z. Xiang, and L. Ren, "Asynchronous data dissemination and its applications," 2021

[5] N. Alhaddad, M. Varia, and H. Zhang, "High-threshold avss with optimal communication complexity," 2021

[6] M. Dahl et. al, "Noah's Ark: Efficient Threshold-FHE Using Noise Flooding," 2023

[7] E. Ben-Sasson, et. al, "Scalable, transparent, and post-quantum secure computational integrity", 2018

[8] B. Bunz, et. al, "Bulletproofs: Short Proofs for Confidential Transactions and More," 2017

[9] C. Gentry, S. Halevi, and V. Lyubashevsky, "Practical Non-interactive Publicly Verifiable Secret Sharing with Thousands of Parties," 2021