Dependable Distributed Systems Notes

José Duarte

June 21, 2020

1 Blockchains

A blockchain is an immutable ledger for recording ordered transactions, maintained within a distributed network of mutually untrusting peers.

In a blockchain transactions are grouped into blocks, each block is linked (chained) to the previously mined block.

In Bitcoin a block can be thought as an header, containing meta-information and the list of transactions, these blocks have the limit size of `1MB`. The information contained in the header is the following:

- **Magic Number** - size of 4 bytes.
- **Block** - size of 4 bytes.
- **Version** - size of 4 bytes.
- **Previous Block Hash** - size of 32 bytes.
- **Merkle Root** - size of 32 bytes.
- **Timestamp** - size of 4 bytes.
- **Difficulty Target** - size of 4 bytes.
- **Nonce** - size of 4 bytes.
- **Transaction Count** - size of 1 to 9 bytes.
- **Transaction List** - size of up to 1 megabyte.

1.1 Ownership

To prove ownership Bitcoin uses public key cryptography, for a transfer between A and B of value m we have the following entry in the log:

$$[K_{pub}A,\ K_{pub}B,\ m]_{K_{priv}A}\ :\ Sign_{K_{priv}A}(T)$$

This way Bitcoin allows anyone with the full list of transfers to verify what user X owns. Furthermore, the use of public key cryptography allows users to remain anonymous, being able to create a new key pair whenever needed.

1.2 Consensus

In Bitcoin one has multiple miners concurrently trying to generate the next block N, whenever one is able to achieve said goal, it makes the new block public, broadcasting it to the network. The other miners then will stop trying to generate block N and move to trying to generate block $N + 1$.

1.2.1 Confirmation

Consider the block N, transactions on such block are required to be confirmed, the process of confirmation is done by mining blocks ahead, that is, blocks $N + 1$, $N + 2$, etc.

Transactions are typically considered to be *confirmed* when 6 ($N + 6$) or more confirmations have been received. For newly minted Bitcoins the number of confirmations required is around 100 ($N + 100$).

1.2.2 Proof-of-Work

The intuition behind Proof-of-Work (POW) is the following, for a given goal, that goal needs to be hard to achieve but easy for others to verify that such goal has been achieved.

Bitcoin uses the Hashcash POW, a summary of the POW is as follows: For the content `Hello, World!` the miner will append a nonce within the interval $[0 - 2^{240}[$, then the miner will hash the content with the appended nonce until the result starts with a given number of zeros.

This is easy to validate by other miners, just send them the content, nonce and candidate hash, they can easily hash the content and nonce to check if it matches the candidate.

1.3 Service Planes

Blockchains are complex systems and thus have a lot of moving parts, these moving parts can be defined as service planes and serve different purposes.

- **Network** - P2P Networking, Ordered Transaction.
- **Ledger** - Decentralized Logging.
- **Transaction Management** - Transaction Message Format and Verification.
- **Consensus** - POW Mechanism.
- **Cryptography** - Public Key Digital Signatures.
- **Storage** - Persistence and Data-Structures.
- **Access Control** - Permission Management.
- **Participation & Membership** - Role Management and "Gatekeeping".
- **Governance** - Node Behavior.

A block can also have other kinds of tools such as:

- External APIs & Integration Facilities
- Deployment and Operation Services
- Tools

1.4 Blockchain Procurement & Analysis

Before taking on any kind of blockchain one must evaluate the goals of the task at hand, trying its best to match them with the available blockchains. Some common picking parameters are:

- Kind of Network
- Language support
- Popularity, Activity, Community, Documentation
- Scalability and Performance
- Readiness for Deployment
- Virtualization and Isolation Support

1.5 Permissionless vs. Permissioned

1.5.1 Permissionless

Permissionless (or public) blockchains are available for anyone to join, in general a participant is only required to have

an (anonymous) identifies, these blockchains do not have access control. Bitcoin and Ethereum are examples of existing permissionless blockchains.

1.5.2 Permissioned

Permissioned blockchains are run among a set of known, verifiable and identified participants, they are also called *Consortium Blockchains* or *Private Blockchains*. This model targets a network where participants have a common goal and interact securely but do not fully trust each other. Nodes may also have different roles in the network.

The base idea behind permissioned blockchains is that to participate a party needs to fulfill a group of requirements. These requirements are such as having an authorized ID and being accepted by other nodes.

Traditionally these blockchains run traditional byzantine consensus algorithms. This works because nodes are previously known and there is membership and access control.

1.6 Ethereum

Like Bitcoin, Etherum is a public blockchain, however it has several extra features such as smart contracts and the possibility for usage of Proof-of-Stake (POS). Casper will implement two rounds of voting (the *prepare* and *commit* phases), furthermore Casper will slash bad validators. The Ethereum blockchain is also faster than Bitcoin and the reward is different.

1.6.1 Block

An Ethereum block consists of three main elements, the block header, a transaction list and an Ommer list.

Ommer List The Ommer List contains all included Uncle blocks. Uncle blocks are valid solutions to the POW that do not make the main chain. This has the objetive of decreasing centralization of the network and rewarding work. If a given Uncle block is included in a main block, 1/32 of the main block miner's reward is given to the Uncle block miner.

1.6.2 Nodes

Ethereum nodes validate all transactions and new blocks, they operate in a P2P fashion and each contain a copy of the entire blockchain. There also exist light clients, these will only execute validations, being mainly used for account balance verification.

1.6.3 Accounts & Wallets

There exist two kinds of accounts on the Ethereum blockchain, these are external owned accounts (EOA) and contract accounts. Both kinds consist of a key pair and allow for interaction with the blockchain.

External Account An external account has an associated nonce, balance, hash code and root. The hash code is the hash of the associated account code (i.e. program code). The root is the root hash of the account associated tree.

Contract Account Contract accounts can store and execute code, they have an associated nonce and balance, as well as an hash code and root.

Wallet Wallets are a set of one or more external accounts, these can be used to store and transfer *ether*.

1.6.4 Casper Protocol

The Casper protocol was proposed as a way to improve on the POW of the Ethereum blockchain, it is a smart contract that implements and monitors POS.

Proof-of-Stake The creator of the next block is chosen by some combination of randomness and a stake (e.g. a number of coins). When compared with traditional POW, POS consumes less energy and can improve throughput conditions.

Validators Validators are nodes that have two voting functions, *prepare* and *commit*, the votes are then weighed by the amount the voter staked. Each validator can only vote once.

Protocol Validators will select a pending block to be prepared, to do so, the validator will send a *prepare* vote[1] to the network, staking a certain amount of coins. When 2/3 of the network agree on the block, voting moves on to the *commit* phase. Finally, the prepared block is again required to be voted[2] by 2/3 of the network, finalizing the block.

1.6.5 Smart Contracts

Smart Contracts are computer protocols intended to digitally facilitate, verify or enforce the negotiation or performance of a contract. They allow the performance of credible transactions without third parties, as verifiable, trackable and irreversible transactions.

Smart contracts comply with the following properties:

- They are executable code.
- The source language is Turing complete.
- They function like an external account.
  - Holding funds.
  - Being able to interact with other accounts as well as contracts.
  - Contain code.
- Smart contracts can be called through transactions.

1.6.6 Contract Execution

In Ethereum the contracts run in the Ethereum Virtual Machine (EVM), there source is written in Solidity. Every full-node on the blockchain processes every transaction and stores the entire state.

Smart contracts are subject to the halting problem, that is, it is impossible to know if they terminate. To deal with this, Ethereum uses the concept of Gas.

Gas Gas is the exchange value used to run code on the blockchain. Each contract is required to provide a maximum gas usage, stopping infinite loops from running. Gas has its own price and market.

[1] Prepare votes reference the last prepared block as well as the last commited block.

[2] Commit votes reference the last prepare block.

2 Database Security

2.1 Database Architecture

A Database Management System (DBMS) is composed of several elements such as:

- DB Management Functions
  - User Management
  - DB Utilities (Administration and Maintenance)
  - External Query Applications
- DB Languages (such as SQL and DDL)
- Persistency and OLTP services
  - Transaction Management & Concurrency
  - File Management
  - Authorization Access Tables
  - Physical DB Mapping

2.1.1 Relational DBMS

These are built upon now common elements such as:

- Entity models and relations
- Tables - and their elements such as rows and columns
- Keys - primary and foreign keys
- Views
- SQL
- User Defined Functions

SQL Structure Query Language (SQL) is the language databases provide for query operations, each database provides its own "flavor" of SQL.

SQL Injection Attacks SQL injection attacks (SQLi) are one of the most common and dangerous threats, the intuition is to send a string which can be interpreted as an SQL command, they can be used to retrieve information, manipulate data or even launch a denial of service attack. When it comes to preventing SQLi attacks, there are several countermeasures available, such as:

- Defensive Coding
  - Data Sanitization and Validation
  - Static-Analysis (Application Wise)
- Intrusion Detection
  - Signature-based
  - Anomaly-based
  - Code analysis
- Runtime Prevention
  - Runtime filters with dynamic detection
  - Sensors in different nodes of routing SQL statements
  - Effects on DB state

2.2 Access Control

Techniques used to grant and revoke authorizations.

2.2.1 Cascading Authorizations

Granting/revoking authorizations will cascade throughout the chain of permissions. That is, if a users authorization A is revoked, authorizations given by means of A will also be revoked. This mechanism can also be time based.

2.2.2 DAC vs. RBAC

Discretionary Access Control Typically applies permissions in a "per-`userid`" base, or organize users according to categories such as the end users, application owners, DB admins, etc.

Role-Based Access Control The intuition behind RBAC is to extend the DAC model by defining more "fine-grain" roles. Allowing roles to be specific for role creation/deletion, permission assignment and more. This approach also allows to establish permissions between roles.

2.2.3 Typical Access Control Management Policies

Several common AC administration policies exist and work for different goals.

Centralized A small number of privileged users to grant and revoke rights.

Ownership-based The owner of a table may grant and revoke rights to the table.

Decentralized In addition to granting and revoking access rights to a table, the owner of the table may grant/revoke authorization rights to other users, allowing them to grant/revoke access rights to the table.

2.3 Inferential Threats

Inference can be considered to be indirect access to information.

Inferential Attacks Inferential attacks are usually performed by insider attackers, or by intruders who obtained valid privileges. An inference action is the process of performing authorized queries and deducing unauthorized information from legitimate responses from the server.

Inference Vulnerability An inference vulnerability exists when the combination of a number of data items is more sensitive that the individual items and their specific access authorization.

Inferential Metadata Inferential metadata refers to the knowledge of the DB structural model, as well as correlations or dependencies among data items.

Inferential Channels The inference channel is defined as the information transfer path by which unauthorized data is accessed by a sequence and combination of authorized data.

2.4 Database Encryption

Database encryption is the last line of defense against malicious actors. There exist several challenges related to DB encryption.

2.4.1 Key Management

Authorized users must access keys used to encrypt and decrypt the stored data. However it is also necessary to support a wide range of users and external applications. Furthermore it is necessary to manage several keys.

2.4.2 Operation Support

The database is required to support SQL as if it was not encrypted, this is not a simple problem as it conflicts with one of the main goals of cryptography, which is to make plain data imperceptible.

2.4.3 Scope of Encryption

DB encryption can be applied to many levels from the entire database to the field as a single unit. The finer the encryption, the more keys are required to manage.

2.4.4 DB Encryption Solution

Encrypted DB version (tables) built and released by the data owner entity as a set of rows, with contiguous binary blocks.

$$B_i = (X_{i1} \parallel X_{i2} \parallel \cdots \parallel X_{iM})$$
$$E(k, B_i) = E(k, (X_{i1} \parallel X_{i2} \parallel \cdots \parallel X_{iM}))$$

The client now must encrypt the data for retrieval. In the client front-end proxy indexes are associated with each table and for some or all of the attributes an index value is created. So we have that for each row of the unencrypted DB the mapping is:

$$(X_{i1}, X_{i2}, \ldots, X_{iM}) \Rightarrow [E(k, B_i), I_{i1}, I_{i2}, \ldots, I_{iM}]$$

$$\begin{array}{ccccc} E(k,B_1) & I_{11} & I_{12} & \ldots & I_{1M} \\ E(k,B_2) & I_{21} & I_{22} & \ldots & I_{1M} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ E(k,B_N) & I_{N1} & I_{N2} & \ldots & I_{NM} \end{array}$$

To create the indexes I_{ij} we generate ranges over the attribute values (covering all possible values) without overlapping partitions.

3 Database Cryptography

3.1 Homomorphic Encryption

Homomorphic encryption is a form of encryption that allows operations over the encrypted results as if the operations were done on the original data.

Homomorphic encryption can be categorized over three different methods.

3.1.1 Encryption Methods

Fully Homomorphic Encryption It is considered to be the holy grail of cryptography and while it has been proven to be theoretically possible there has been no progress yet. FHE supports unlimited additions and multiplications.

Partially Homomorphic Encryption PHE supports only one method, either additions or multiplications.

Somewhat Homomorphic Encryption SWHE tries to bridge the gap between methods providing both in a limited manner.

3.1.2 Beyond HE Methods

While HE provides versatility to the processing of encrypted data it is not adequate for every purpose. Some factors are required to be considered such as.

- The output is encrypted, in contrast to other existing obfuscation and functional encryption techniques, which retrieve the result as plaintext.
- All inputs must be encrypted with the same key, that is, the key that encrypts the original data is also required to encrypted the operation input.
- HE does not provide authentication or integrity guarantees on computations, the assumption is that the algorithms are correct and results are obtained by the correct execution.

3.2 CryptDB

CryptDB tries to address information leakage by encrypting the DB content. It provides practical confidentiality and encrypted query processing.

3.2.1 Adversary Model

Honest-but-Curious Admin The attacker may be the DB admin trying to extract information over client documents. This actor is considered to have full control over the DBMS, even access to RAM. CryptDB aims to provide confidentiality, the attacker is assumed to be passive, only trying to extract information, not modifying information.

Leaked Keys The adversary may have gotten access to the keys used to encrypt the entire data base. An adversary of who compromises the application server or the proxy can only decrypt data of currently logged-in users. The countermeasure CryptDB provides is not letting the inactive user keys to be available to the attacker.

3.2.2 Proposed Techniques

Using PHE This provides a set of primitive operators such as equality checks, order comparisons and aggregations. It also provides a new construction for joins.

Adjustable Query-Based Encryption Use onion encryption, that is, encrypt using several methods one after the other.

Chain-Encryption Keys Onion encryption chains, based on the password of one of the users with data access authorization. Guarantee that data is inaccessible if authorized users are not logged in. Capture application based-access control requirements and support data-sharing by a delegation model.

3.3 Cipherbase

Cipherbase was introduced by Microsoft, it is a full-fledged SQL database providing high performance and high data confidentiality. It support stored procedures, indexes, recovery procedures as well as full SQL, column-level encryption and other encryption schemes. Cipherbase makes use of special hardware for encryption processes.

4 Trusted Computing

Trusted Computing (TC) is the notion of embedding security into the hardware, making it readily available on commodity devices. The notion originated in the Trusted Computing Group[3]. The original goal of TC was the development of Trusted Platform Modules (hardware/firmware) and more recently Trusted Network Connect (TNC) a protocol specification for secure network connections with authentication, authorization and accountability.

Some general features present in TC are:

- **Secure Boot** - which allows the system to boot into a defined and trusted configuration.
- **Curtained Memory** - provides strong memory isolation (i.e. memory that cannot be read by other processes).
- **Sealed Storage** - Allows software to keep cryptographically secure secrets.
- **Secure I/O** - Thwarts attacks like keyloggers and screen scrapers.
- **Integrity Measurements** - Ability to compute hashes of executable code, configuration data, and other system state information.
- **Remote Attestation** - Allows a trusted device to present reliable evidence to remote parties of the about the running software.

Software Attestation Software attestation can be divided into local and remote attestation. Local attestation is the ability of a program to authenticate itself. Remote attestation is the ability of a system to make reliable statements about the software it is running in another system.

4.1 Trusted Platform Module

A Trusted Platform Module (TPM) is hardware designed according to the ISO/IEC 11889 standard. It is a chip built into a dedicated microcontroller which can be used as a secure cryptographic processor.

4.1.1 Issues

The has been some resistance and criticism over the generalized adoption of TPMs such as:

- Rise of privacy concerns.
- Physical/Administrative access to hardware bases.
- The private endorsement key is fundamental to the security of the TPM circuit, this key is never made available to the end-user, requiring users to trust the manufacturer.
- Rigidity and lack of relevant trusted functions for specific deployments.
- The performance of TPMs is lackluster. Vendors have little incentive to use faster (and more expensive) parts.

4.1.2 Services

TPMs provide three basic services, authenticated boot, attestation and encryption facilities.

[3] `https://trustedcomputinggroup.org`

Authentication Provides mean to unequivocally identify the TPM module and platform. Each TPM has a embedded key pair, the *Endorsement Key*. This key is signed by the manufacturer as to guarantee the correctness the chip and validity of the key.

Trusted Boot The function responsible for booting the OS in stages, ensuring that each portion of the OS is trusted and approved for use. A TC-enabled system using TPM devices maintains a list of appropriate HW and SW components for verification of the boot process.

Authenticated Boot Responsible for booting the entire OS in successive stages, assuming that each portion of the OS loaded is a version that is approved for use. At each stage the digital signature associated with the software is verified and loaded and the tamper evidence log of the entire boot process is registered to detect any tampering with the log.

Platform Configuration Registers PCRs are special registers which are primarily used to store fingerprints of a portion of the software booting on a computer, they are guaranteed to be reset upon a computer boot.

Extended Authenticated Configurations From the base functions it is possible to expand the trust-boundary to include new hardware devices, software utilities, stack layers or applications.

Certification Service Once a configuration is achieved and logged the TPM can certify the configuration to others, producing a digital certificate. There is confidence that the configuration is unaltered because the TPM is considered to be trustworthy and only the TPM possesses the TPM private key.

Remote Attestation Protocol Aims to provide a remote entity with means to assess the security if the contacted platform. Allows to retrieve from the TPM a signature called *quote* that tells the resulting platform configuration, built through the attested platform boot, by the loaded firmware and software.

Encryption Service The TPM keeps a secret master key, unique to the machine.

- **Binding Encryption** encrypts data using the TPM binding key.
- **Sealing Encryption** encrypts data so that it can only be decrypted by a machine with a certain verified configuration.

4.1.3 Functions

I/O All commands enter/exit through the I/O bus, it is the only way to communicate with the TPM functions.

Cryptography Co-Processor Processor for encryption functions, it provides utilities such as hashing, RSA encryption and digital signatures as well as AES encryption.

Key Generation Able to create RSA keypairs and generate AES symmetric keys.

HMAC & SHA-1 Engines Provides hardware HMAC and SHA-1 implementations.

Random Number Generator Useful for key-generation, nonces and generally as a source of randomness.

Power Detection Manages the TPM power state in conjunction with the platform power state.

Execution Engine Runs program code to dispatch the received commands.

Non-Volatile Memory Used to store TPM parameters in a persistent manner.

Volatile Memory Temporary storage for the execution engine, as well as for volatile parameters.