Orwell. From Bitcoin to secure Domain Name System

Michał Jabczyński, Michał Szychowiak

Poznań University of Technology
Piotrowo 2, 60-965 Poznań, Poland
{Michal.Jabczynski, Michal.Szychowiak}@put.poznan.pl

Abstract

This paper addresses the problem of distributed trust ureliability and trusted third party dependency in a decentralized network. We demonstrate that trust can be created and maintained in a decentralized way, without relying on certificate authorities but instead using the Nash equilibrium, based on individual profit, made possible by the development of cryptocurrencies. We show how Write-Once Storage can be used as the basis of a secure naming system. Finally, we present Orwell – a secure decentralized domain name system based on the blockchain protocol.

1. Introduction

Over the last 20 years, the Internet changed significantly – it became much more decentralized, and so did the services it provides. The concept of trust also evolved – it has seemingly become a resource that can be bought, sold and stolen. What did not change about trust is its centralized nature. Virtually all trust establishing systems delegate all critical decisions to *trusted third parties*. It would be *wrong* to assume that all certification authorities adhere to the same standards and provide the same level of identity assurance. In fact, many certificate authorities issue certificates without any legal or technical processes to authenticate the identity of the entity requesting the certificate, relying only on the Domain Name Registrar details to validate identity ownership.

The purpose of this paper is to address the issue of trusted third party dependency in the public key infrastructure. We will demonstrate that trust can be created and maintained in a decentralized way, without relying on certificate authorities but instead using the Nash equilibrium (Nash, 1950) based on individual profit, made possible by the development of cryptocurrencies. We will show how *Write-Once Storage* can be used as the basis of a secure naming system. We will also present Orwell – a secure decentralized domain name system based on the Blockchain protocol, introduced by Bitcoin (Nakamoto, 2009).

2. Related work

The *trusted third party* (TTP) is often misused as a building block of secure systems. Instead of being treated as a weak link in the chain, it is used as a black box that encapsulates existing problems and hides them based on the assumption of infallibility and reliability. The TTP is a costly and risky element that should be thoroughly thought through before use, and can be used either as a last resort, or an unavoidable risk. The reason for it is that introducing a TTP to the system is equivalent to introducing a potential security hole (Szabo, 2001), as it is an *a priori* weakness and an additional point of attack.

2.1. Write Once Storage

One of the most useful abstractions over a trust system is a *Write Once Storage*. Let us consider a basic key-value storage system with the following properties:

- **public access property** – any party may attempt reading or writing any key,
- **read fail property** – if no value was written for the given key, a read operation will fail explicitly,
- **read agreement property** – if the read operation succeeds, no other successful read will ever return a different value.

This storage could serve as a basis for a distributed trust system. An identity registration could be executed by attempting a write operation – the key would be equal to a user name, and the value would be the owner's public key. After the write, the node would periodically try to read a value until the read succeeds. If the returned value is equal to the public key, the registration succeeded and no other node will ever observe a different value. If the returned value is different, the registration failed and will never succeed in the future.

Using custom keys in a Write Once Storage requires agreement between nodes. The agreement can take many forms – it can be enforced by a trusted third party. Notice that the Public Key Infrastructure can be easily abstracted away into a Write Once Storage, where the read agreement property is a result of organizing certificate authorities into a tree structure. Provided that the certificate authorities always agreed on who should be associated with the particular domain (which is one of the main false assumptions of a PKI), the system could be considered a secure trust system.

2.2. Bitcoin

Bitcoin (Nakamoto, 2009) is the first successful example of a cryptocurrency. The significance of Bitcoin lies in the fact that it is a decentralized *peer-to-peer* (P2P) system, with no trusted third parties involved. There is no governing body that manages the distribution of currency, its exchange rate or any other parameter. Users participating in the network deploy nodes that collectively verify all transactions happening in real time. The fundamental concept behind Bitcoin is the *Blockchain* – a distributed ledger that tracks the state of all accounts in the system. The

Blockchain establishes total order of all transactions with no central coordination, through voting with CPU power.

2.2.1. Wallets

Bitcoin are stored in *wallets* – accounts in the Bitcoin network. Each wallet consists of an asymmetric key pair, where the public key serves as the account number and identifier, and private key allows the owner to spend coins stored in the wallet. This means that every user can generate an arbitrary number of accounts, without any central authority. Because coins are assigned to wallets, not nodes, anybody can receive funds without any direct communication.

2.2.2. Transactions

The coin transfer between accounts is called a transaction. It is a structure that contains the *outputs* (records containing which account receives a certain amount of coins), the *inputs* (identifiers of the outputs from previous transactions that this transaction spends), and signatures that verify the transaction as coming from an authorized person. The transaction is considered valid if the inputs it tries to spend are not yet spent, and if the signatures it contains are valid. The transaction is created by the node belonging to the owner of the wallet, and it is broadcasted across the Bitcoin network. However, the act of announcing the transaction is not equivalent with the definite proof that the transaction actually took place. In order for the transaction to be accepted into the network, it has to be included in the Blockchain.

2.2.3. Mining

The process of creating blocks is called *mining*. It is a computationally expensive process that involves hashing the block contents in order to produce a hash that fits within a defined integer range. Producing blocks is a profitable exercise – each block yields a defined amount of coins that is granted to the person that managed to find the correct hash for the next block. It is easy to notice that it is perfectly possible for nodes in the network to produce two blocks simultaneously. However, only one of these blocks will ever be accepted.

2.3. Namecoin

The model of Bitcoin involves a peer-to-peer system where participants are continuously validating a series of transactions without any central control. That model was directly applied to the domain name system by modifying the Bitcoin protocol and the result was called Namecoin (NMC). Namecoin aims at providing a secure naming system that is partially compatible with DNS (provides a compatible naming scheme). All information about all domains is stored inside the Blockchain, including keys, values and subdomain structure.

Despite the initial popularity, Namecoin has suffered from several limitations and numerous bugs (Gronager, 2013) leading it to become an abandoned project.

3. Orwell

Orwell is a heavily modified blockchain protocol, based on the basic idea of Bitcoin. It provides its users with tools for generating wallets, and sending currency between accounts. The fundamental difference between Orwell and Bitcoin is purpose. Orwell uses the concept of a cryptocurrency as the means of establishing and sustaining a decentralized write-once storage that can be used as a functional replacement for the DNS system. Orwell focuses on solving the problems of Namecoin and optimizing the performance of name resolution.

The goals of the system are the following:

- to provide a **fully functional replacement** for the DNS system, with ownership transfer, caching, arbitrary record types and values,
- to operate and sustain a truly **decentralized** DNS system, without trusted third parties,
- to solve the problem of **registration race**, making it possible for a domain name to be announced after the registration itself, without the necessity of disclosing it to external parties,
- to create a query-response name resolution protocol that provides **provable name resolution** without the necessity to know the whole chain (making it possible for *thin clients* to use and resolve Orwell domain names),
- to make the system **resistant to man-in-the-middle attacks**.

3.1. Design overview

An efficient design of a decentralized, cryptographically secure DNS-like system must address two technical challenges: enforcing strict mapping between names and public keys, which requires an unambiguous domain registration order, and at the same time allowing for elastic, eventually consistent updates once the domain ownership is guaranteed. To make it possible, Orwell is composed of two separate protocols. By splitting the design into two subsystems, nodes can specialize and serve only one function, greatly enhancing performance and reducing he requirements imposed on clients. The protocols are:

- **Orchain** – the Blockchain algorithm and protocol that maintains the transactions and establishes registration order and maintains the mapping between domain names and public keys,
- **Orcache** – a distributed key-value cache with partial replication and eventual consistency, maintaining the mapping between public keys and subdomain information contents (Jabczyński and Szychowiak, 2015).

3.2. Orchain

Orchain defines a cryptocurrency system, not unlike Bitcoin. Each and every user in the system can create its own asymmetric key pair and use the public part as his/her

identifier. Orchain users can transfer units or currency between one another, in exactly the same way as in Bitcoin. The user transactions can be however associated with custom payload that allows users to perform operations on domains – register, update and transfer.

3.2.1. Transactions

The fundamental data structure that describes an act of currency transfer is called a *transaction*. Each transaction can hold a value called payload. The payload is used as a method of announcing domain registrations, tickets and transfers. Orchain users can transfer units or currency (orcoins) between one another, in exactly the same way as in Bitcoin. This will be considered as a voluntary regitration fee. The transaction contains a signature (stored in the field `Proof`) signed using the private key of the wallet owner. The transaction could in principle allow multiple independent parties to issue a single transaction with multiple inputs belonging to multiple entities, however this was removed to reduce complexity and transaction size.

The Blockchain algorithm is the core of Orchain protocol. It is an ever growing tree of structures called *blocks*. Although Blockchain is always referred to as a list, it is an fact a tree structure. Nodes in the network try to find and redistribute the longest chain in the tree, and ignore the shorter branches. All nodes start the the same block, called *genesis block*, which is hard-coded into the protocol.

Transaction is considered valid if and only if it is stored in one of the blocks in the Blockchain. By using the order of blocks as means of ordering transactions, nodes can establish which transactions spend bills first, avoiding the *double spending* problem. The first transaction to spend the bill wins, and can be included in the chain. The block is considered invalid if it contains any invalid transaction. Therefore, the whole chain always contains only non-conflicting transactions.

Each block has its own unique identifier. It is a 256-bit (32 byte) integer obtained by hashing the `Header` contents using the `SHA256` (NIST, 2002) function. In order for the block to be accepted into the Blockchain, the block identifier must meet a specific criterion, called *proof of work*. Proof of work is a computationally expensive problem that needs to be solved in order for the block to be considered valid. The more difficult the problem, the more computing power the network needs in order to generate the new block. By auto-balancing the complexity of the proof of work problem (represented as the variable called `difficulty`), the network can control and maintain a fixed pace of block generation (ideally 6 blocks per hour, or 1 block per 10 minutes).

The proof of work problem is essentially a problem of randomly selecting an integer (hash value) lower than a given threshold (called `target`). The predicate for a valid identifier is the following:

$$IdentifierValid(block) \equiv block.ID() < target(difficulty)$$

Where the $target(d)$ is an upper bound limit for the random identifier value. Let S be equal to the number of possible identifier values (in our case 2^{256}). The $target$ can be obtained from the following formula:

$$target(d) = S\Big(1 - \frac{1}{2^{\frac{1}{d}}}\Big)$$

As the probability of obtaining any particular value of the hash function is considered equal (we assume that the `SHA256` function behaves like a perfect hash function), the average number of attempts required to obtain a valid identifier for the given difficulty is exactly equal to the difficulty value.

Any change of the block content results in a different hash value, and a different block identifier. The `nonce` field can be used freely for that purpose.

3.2.2. Domain Registration

Domain system is the main purpose of the Orwell protocol. Domain names are keys in the write-once storage established by Orwell. The values are public key identifiers, and point to specific cards in the Orcache protocol. Therefore Orchain deals only with the naming system and does not hold the actual domain record data.

The fundamental problem that Orwell solves is registration order – it allows for all parties to agree on which registration (act of taking the domain) took place first. The first domain registration always wins, and has a defined time period during which it is considered valid. The owner of the valid domain can transfer it at any time and to any account, including its own.

In order to prevent users from greedily registering numerous domains, there are limits and rules that describe how the domains can be registered. In each block, only a defined number of domains can be registered. If more registrations is issued than what a block is allowed to contain, registrations with higher fees are accepted. This way the price of the registration can vary in time, and influence the currency value. Moreover, miners are more highly incentivised towards mining, as it yields more profit. The number of domains allowed to be registered per block depends on the registration price (median of the previous fees). If the registration price rises, the number of allowed registrations also rises, and vice versa. This way the number of domain registrations is constantly tuned and managed by the market itself, providing a fair and affordable price while preventing greedy registrations.

Registration race is a problem related to disclosing the domain name before it is registered. If the intention of registering a domain name becomes the public knowledge before the registration succeeds, malicious parties can attempt to register the same domain simultaneously with the original party, with the intention of reselling the domain. This scenario is real in some cases with DNS – malicious entities host domain checking services, where a user can check if the domain is free to register. The domain check requests can be collected and then used to register the domains before the original user succeeds to do so. This scenario would be even more widespread in a P2P network, especially when not just the intent of registering the domain name is public, but also the attempt to do so. This would make domain registration a tricky and frustrating task.

To alleviate this issue, Orwell hides the domain name until the registration has succeeded. No party can know which domain is being registered until it is successfully claimed. To achieve this, the domain registration process is split into three steps:

1. **Registration ticket** – as the registration starts, the domain claimer computes the registration ticket. The ticket is a hash of the `Domain` data structure, which holds the domain name, designated owner and the lease time. The ticket is included in the payload of a transaction and published in the network. If the transaction fee is high enough, a miner will include the ticket in the Blockchain.

2. **Domain announcement** – after the ticket is accepted in the network, but before 144 blocks are published after it (24 hours), the `Domain` data structure is published in the payload of a transaction. The hash of the `Domain` must match the previously announced ticket. This way the network knows what was the actual domain that was registered.

3. **Domain confirmation** – exactly 144 blocks after the ticket was announced, the miner is *obliged* to check the ticket announced previously. If the ticket has a matching domain (announced in step 2) and if the domain is available (was either not registered or the previous lease has ended), the domain is included in the `Domains` array field of the new block. The ticket check is obligatory and the block would be invalid if any correctly announced ticket was ignored.

In the simplest case, only one person attempts to register a domain. After the ticket is accepted into the Blockchain, the domain is announced and then confirmed. However, the more interesting case happens when multiple parties attempt to claim the same domain simultaneously. If this happens, the ticket order defines the registration priority. If both tickets are announced inside the same block, transaction order inside the `Transactions` array in the block decides. As the domain name itself is not known at that point, the ticket order is effectively random and therefore fair.

3.2.3. Name resolution

Domain is considered valid if the matching `Domain` structure is included in the Domains array in any of the blocks, and if the expiration date specified in the `ValidUntilBlock` field is big enough. In order resolve a domain name, the client needs to have the longest chain in the Blockchain. However, the client is not required to store whole blocks – only block headers are required. As each block header is only 896 bits long (112 bytes), users need to store only 5,61 MiB of data per year. This number does not depend on the number of registrations or transactions, and remains proportional only to the block generation rate, which self-stabilizes at 6 block per hour. This amount of data is acceptable even for mobile users.

The resolution protocol contains the following steps:

1. **Download the longest chain.** This can be done continuously and in the background. The downstream rate is low, as only 112B need to be downloaded each 10 minutes. The longest chain can be downloaded by a system process and published to applications as a read-only file.

2. **Ask any full peer in the network about the domain**. The *full peer* stores the complete chain (or all currently registered domains).

3. **The client checks the received response.** The response is valid if:

   - the domain did not expire,
   - the designated block (`BlockID`) actually exists in the longest chain,
   - the `Proof` is correct (Jabczyński and Szychowiak, 2015).

It must be noted that the response verification requires no prior knowledge given by any authoritative third party – the client can obtain the correct Blockchain on its own, asking multiple peers and choosing the longest one. Assuming that the attacker has no control over what connections the client establishes, the client will always be able to get the longest chain and correctly resolve names. The authorization functionality established by Orchain can be used recursively to secure the connections between nodes themselves, further strengthening the network.

3.2.4. Network protocol and synchronization

Orchain works in an unstructured network. Nodes do not need to connect or maintain any particular structure of connections. The only assumption with respect to network topology is that the graph is connected. To minimize the chance of splitting the network, each node establishes random connections and tries to maintain at least 16. The connectedness of the network graph ensures that the gossip-based broadcast of blocks will remain efficient.

Once the nodes establish a connection, the synchronization procedure is executed periodically to ensure that both nodes agree on the longest chain. The objective of the procedure is to find a common ancestor (the latest, in the best case) in the tree of blocks and, if the remote peer has a longer chain that descends from the common ancestor, download these blocks. The chain length is compared not by the number of blocks, but the sum of difficulty values associated with blocks. This prevents the nodes from choosing artificially forged blocks with cheap production time and unbounded length. The steps of the procedure are the following:

1. Send the `MsgHead` request, containing the `Id` of the last known block in the longest chain, and `Work` – the sum of `Difficulty` values associated with all blocks in the chain.

2. Receive the `MsgTail` response. The data structure contains the total `Work` done in all blocks known by the remote peer, as well as the `Headers` array of block headers that descend from the `Id` specified in the request (if the `Id` is known by the remote peer).

3. If:

   - the `Headers` array is not empty, ask the remote peer for all the unknown blocks specified in the `Headers`, download them and return,
   - the `Headers` is empty, but the value of `Work` is greater then the local one, re-send the `MsgHead` request with the Id twice earlier then previously.
   - the `Headers` is empty and the value of `Work` is lower than the local one, return.

With the following approach, the peer performs a binary search over the chain to find a common ancestor. After the ancestor is found, the peer compares its own chain with the chain stored by the remote peer. If the remote peer has the longer chain, it is downloaded and applied locally. By using this procedure periodically and asking multiple peers, the longest chain quickly spreads across the network.

4. Summary

Orwell is a proposal of a decentralized system that provides both secure naming scheme and authentication. By leveraging the breakthrough innovation of Blockchain algorithm, completely new P2P systems can be constructed with much stronger guarantees than before. Cryptocurrency can be used to stabilize P2P networks through the promise of profit. As long as the peers in the network are attracted towards gathering more currency established by the system, the stability of consensus algorithm is guaranteed. The consensus can be used to totally order the transaction registrations, ensuring that all nodes in the network observe the same global state.

Orchain works in an unstructured network. Nodes do not need to connect or maintain any particular structure of connections. The only assumption with respect to network topology is that the graph is connected.

Orwell introduces various optimizations with respect to Bitcoin and Namecoin. It increases performance through reduction of the size of data structures, modifies the domain resolution algorithm to remove the requirement for the clients to store the complete chain, and separates the domain mapping layer from the data storage layer, increasing throughput and decreasing response time. Many further improvements can be done, specifically concerning performance with respect to memory footprint and attack vectors. Orwell will be developed further, with the focus on developing libraries for various programming languages to enable efficient address resolution. The next goal is integrating the address resolution into Mozilla Firefox via a plugin, thus making the new addressing scheme easily available for normal users.

5. References

Gronager, Michael, 2013. Namecoin was still-born, I had to switch off life-support. [on-line] https://bitcointalk.org/index.php?topic=310954.

Jabczyński, Michał and Michał Szychowiak, 2015. Orwell. Distributed trust system for a dependable domain name space infrastructure. Technical Report RA-2/15, Poznań University of Technology.

Nakamoto, Satoshi, 2009. Bitcoin: A peer-to-peer electronic cash system. [on-line] https://bitcoin.org/bitcoin.pdf.

Nash, John, 1950. Equilibrium points in n-person games. Proceedings of the National Academy of Sciences 36(1):48-49.

NIST, 2002. Federal Information Processing Standards Publication 180-2: Secure Hash Standard. [on-line] http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf.

Szabo, Nick, 2001. Trusted third parties are security holes. [on-line] http://szabo.best.vwh.net/ttps.html.