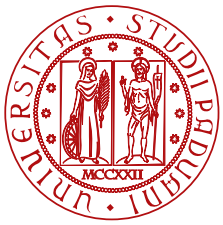

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DEPARTMENT OF INFORMATION ENGINEERING

BACHELOR'S DEGREE IN COMPUTER ENGINEERING

Secure VPN authentication through Bitcoin and Blockchain: an implementation

Supervisor
Prof. Giovanni Perin

Candidate
Luca Parolini

ACADEMIC YEAR 2024-2025

Date 10/03/2025

Il più grande ringraziamento va alla mia famiglia. Ai miei genitori, Beatrice e Paolo, grazie che mi sostenete sempre nel perseguire la mia felicità e i miei obbiettivi, senza se e senza ma. Ai miei nonni, Letizia, Gino, e Fiorenza, che mi consigliano sempre saggiamente, mi insegnano stare al mondo e mi sono sempre vicini. Una ringraziamento particolare, a Chiara, la mia sorellina che mi ispira sempre a fare meglio e ad essere una persona migliore.

Ringrazio con tutto il cuore Mila, per starmi accanto nei momenti difficili e per avermi mostrato il mondo con occhi diversi. Grazie, perchè mi ha insegnato ad amare e ad apprezzare i piccoli momenti.

Un grazie infinito anche ai miei amici più cari, che mi hanno accompagnato in questo percorso di crescita come dei fratelli.

Abstract

The objective of this thesis is to describe the implementation of an authentication method for a VPN server through Bitcoin payments. Bitcoin payments enable the exchange of value in a pseudonymous and decentralized manner, while VPN technology ensures secure and protected connections between different network components. The integration of VPN technology with Bitcoin payments thus allows for permissionless and anonymous access to a fully protected connection, providing users with a centralized service that remains entirely anonymous and secure.

Through the analysis of this implementation, we will evaluate its advantages and disadvantages, offering guidelines for designing an authentication method that leverages Bitcoin payments while safeguarding user privacy.

Contents

List of Figures

Chapter 1

Introduction

1.1 Context and motivation

Nowadays, VPN services have become increasingly popular, with many individuals using them for work or to protect their privacy. These services are now widely adopted even by people who are not particularly tech-savvy. This thesis aims to address and resolve the most significant data leak issue in paid VPN services: the payment phase itself.

VPN services offer privacy protection and secure connections, but when a user proceeds to pay for the service, a major privacy concern arises: the need to provide banking information.

Moreover, the limitations and vulnerabilities of centralized authentication have been extensively studied by researchers across various fields and with different distributed ledger technologies. These studies highlight the risks associated with centralized credential storage, such as single points of failure, data breaches, and user privacy concerns. Alternative approaches leveraging blockchain and cryptographic mechanisms have been proposed to enhance security and decentralization in authentication systems.

One notable example of such a decentralized approach is BCTrust, a blockchain-based authentication mechanism specifically designed for Internet of Things (IoT) environments with constrained computational resources [1]. Given the increasing interconnectivity of billions of autonomous devices in the industrial, medical and agricultural sectors, traditional centralized authentication systems become impractical due to security vulnerabilities and resource limitations. BCTrust introduces a robust and efficient authentication framework that ensures device integrity and secure communication without relying on a central authority.

Similarly, the use of blockchain for secure authentication has been explored in the healthcare domain. A study on decentralized authentication for distributed hospital networks demonstrates how blockchain can enhance patient identity verification and inter-hospital communication without requiring re-authentication [2]. Traditional authentication mechanisms in healthcare systems

often depend on a trusted third party, introducing delays and increasing security risks. By adopting a decentralized model, the proposed solution reduces authentication overhead, improves response time, and ensures data integrity across interconnected healthcare facilities.

These examples underscore the transformative potential of blockchain in authentication frameworks. By decentralizing credential verification and removing the reliance on a central authority, blockchain-based authentication systems mitigate the risks associated with traditional methods while improving efficiency and security. This thesis explores the application of these principles in a novel authentication mechanism for Virtual Private Networks (VPNs), using Bitcoin transactions as proof-of-identity to provide a secure and decentralized access control mechanism.

1.2 Problem statement

Sharing banking information creates a privacy leak and exposes a vast amount of sensitive data to a third-party provider, even though this information is unnecessary for accessing the VPN service itself. However, this sharing becomes unavoidable when using traditional payment systems that rely on banks.

Beyond privacy considerations, another significant limitation that is often underestimated is the necessity of owning a bank account. Although this requirement may seem like a minor hurdle for individuals living in developed countries, it is far from straightforward for the rest of the world. According to studies conducted by the Federal Reserve Bank of Philadelphia, even within the United States alone, there remains a 4% unbanked population [3]. Given a total population of 341 million inhabitants (2025 census [4]), this figure corresponds to approximately 13.4 million people.

In the rest of the world, particularly in developing countries, the percentages are substantially higher. Looking back roughly a decade, in 2009 half of the global population did not have access to a bank account [5]. More recent studies indicate that this figure has decreased to 1.4 billion people without a bank account [6], with 54% of these individuals being women. Therefore, although the trend appears to be in notable decline, the numbers remain substantial and cannot be disregarded. In today's digital era, the ability to access online services plays a crucial role in safeguarding privacy and, consequently, in preserving individual freedom. There are numerous movements and foundations endeavoring to combat this dependence on the traditional financial system. One example is "Bitcoin Dada" [7], which aims to restore women's freedoms in Kenya through financial education and Bitcoin, in order to attain financial independence.

The loss of these freedoms becomes even more pronounced in contexts where autocratic or dictatorial states repress fundamental liberties, with freedom of expression and speech being

foremost among them.

1.3 Proposed solution

This thesis proposes replacing traditional payment methods with digital alternatives, specifically utilizing Bitcoin and its public blockchain to authenticate users while enabling a more private and secure use of the VPN service.

The VPN server will only be responsible for generating a Bitcoin address to which the user must send the payment. Once the payment is successfully completed, the user will simply sign a digitally provided message—along with the Bitcoin address—using the same private key used for the payment. This signed message will then be sent to the server for verification, ensuring a seamless and privacy-preserving authentication process.

This payment method allows the user to remain anonymous and maintain privacy without disclosing any sensitive information. By making a single transaction, namely the payment, the user carries out two essential steps for the service provider: first, they pay, and second, they authenticate themselves. During the payment process, the user shares a public key that is published via the public blockchain.

1.4 Thesis structure

The discussion of these topics will be divided into the following chapters:

- **Chapter 2**: Overview of VPN technology functionality, Bitcoin payments, and Blockchain
- **Chapter 3**: Description of our implementation's features and possible improvements
- **Chapter 4**: In-depth analysis and description of the implementation code and technical aspects
- **Chapter 5**: Conclusions on possible improvements applicable to the implementation and future developments

Chapter 2

Background

2.1 VPNs and Authentication

A *Virtual Private Network (VPN)* is a technical remedy that allows a secure encrypted tunnel over a generally susceptible network, i.e. the Internet. The main reason a VPN is implemented is in order to allow users or devices securely join private networks meanwhile also protecting from interception or abuse [8]. In recent a decade, usage of a VPN increased dramatically as a consequence of increased awareness about security, rising demands for corporate anonymity, as well as a growing need to avoid geographical restrictions [9].

Virtual Private Networks (VPNs) work by wrapping users' data in a secure tunnel that protects it from interception or modification by third parties. Various protocols can be utilized in doing so, with OpenVPN, WireGuard, and IPsec as examples. Such protocols are supported by cryptographically secure methods that involve symmetric cryptography, asymmetric cryptography, digital signatures, as well as hashing. In a typical case, it involves a key exchange between two points, which can involve a combination of both TLS/SSL protocols or Diffie-Hellman key exchange in order to have just those approved in possession with decryption as well as comprehension over transmitted data.

A fundamental component in the security that is offered by Virtual Private Networks (VPNs) is *authentication*, which is intended to verify both a device or user identity as well as its legitimacy in making a connection attempt. Typical methods generally involve a combination of a username and password, digital certificates, or a variety of forms of multi-factor authentication (MFA). The traditional methods, though, are dependent on centralized user databases, third-party certificate authorities, or devices holding identifiable private data, hence presenting single points of failure.

Current research explores new decentralized authentication methods [10]. It combines blockchain technologies with cryptographically-based authentication methods in order to mini-

mize central authoritative dependencies with a possibility of secure identity validation.

2.2 Integrating Blockchain and VPN

The concept of merging blockchain technology with VPN architectures has attracted attention in the academic community. For instance, blockchain platforms enable decentralized identities or payment channels that replace conventional reliance on banks and centralized billing. In many proposals, the blockchain ledger itself can record payment proofs or membership states in a manner that is difficult to falsify [10]. Moreover, combining VPNs with smart contracts can allow on-demand or pay-per-use services in a trustless environment.

Although this thesis focuses on *implementing* a simplified proof-of-concept, a brief exploration of related research helps clarify the potential use cases of integrating blockchain solutions with VPNs. These approaches are still evolving, but many see the merge of secure tunneling and decentralized ledgers as a powerful tool, particularly where pseudonymity, resistance to censorship, or cross-border usage are primary concerns.

In the VPN industry, several companies already offer their services through Bitcoin payments, including ProtonVPN [11], NordVPN [12], and Mullvad [13]. Although these services accept Bitcoin or other cryptocurrencies as payment methods, their authentication systems are not based on cryptographic proof of payment. Instead, they typically rely on conventional methods such as email-password authentication or multi-factor authentication (MFA).

2.3 Bitcoin and the Blockchain

Bitcoin is widely recognized as the first decentralized digital currency, empowering peer-to-peer transactions without the intervention of a centralized authority. It is built on a distributed database known as the *Blockchain*, which records all transactions in a transparent and tamper-resistant way [14].

In the Bitcoin ecosystem, each participant manages one or more cryptographic key pairs. These keys are the foundation of ownership and control. A transaction transferring Bitcoin from one address to another must be digitally signed by the sender, proving the ownership of the currency. The decentralized nodes in the network validate and record such transactions, preventing malicious actors from double-spending or fraud.

2.3.1 Public and Private Keys

Bitcoin utilizes an asymmetric cryptographic model based on *Elliptic Curve Cryptography (ECC)*. In particular, Bitcoin uses the `secp256k1` elliptic curve, which is defined by the Na-

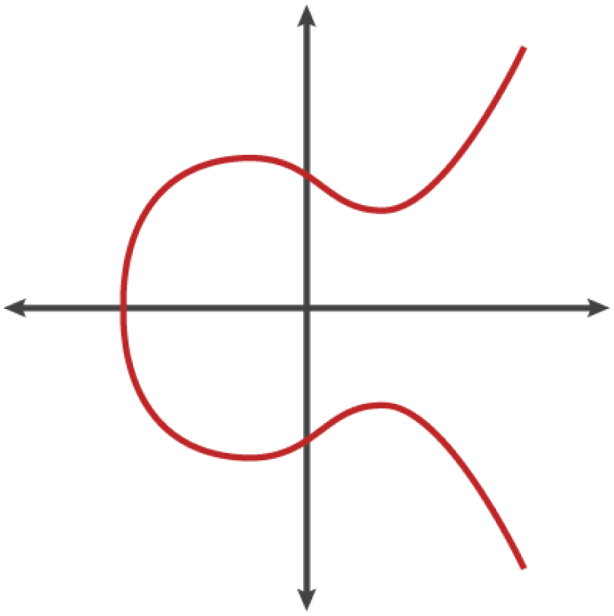

Figure 2.1: Elliptic curve

tional Institute of Standards and Technology (NIST). The curve follows the equation:

$$y^2 \mod p = (x^3 + 7) \mod p$$

where p is a very large prime number. This equation defines a finite field of prime order p.

The generation of a public key involves selecting a randomly determined number k, which is then multiplied by a predefined point on the curve, known as the *generator point* G. This generator point remains the same for all Bitcoin keys. The resulting public key K is given by the equation:

$$K = G \cdot k$$

This operation ensures that while the public key K can be freely shared, the private key k remains securely hidden, reinforcing the cryptographic security of Bitcoin transactions [14].

A Bitcoin address, commonly represented by Base58 encoding, is ultimately a hashed form of this public key. The Base58 encoding removes from the English alphabet all the ambiguous characters such as 1 and l or 0 and O.

To spend Bitcoin, the sender must sign the transaction using the private key, proving ownership of the funds. The Bitcoin network then validates the signature through the corresponding public key, ensuring that only the entity with rightful ownership can authorize the transfer.

2.3.2 Transaction Script and Bitcoin Script

The Bitcoin transaction scripting language is not similar to modern programming languages; instead, it follows a reverse notation style, similar to Forth. Most transactions today are of the P2PKH type, which we will discuss later. This type of transaction follows the standard ”Alice pays Bob” model. However, Bitcoin transactions are not limited to this behavior: Through scripting, it is possible to define complex payment conditions.

The scripting language includes many operators, but is deliberately restricted: it lacks complex control flow structures beyond basic conditional checks and loops. This limitation ensures that the language is not *Turing complete*, preventing inherent vulnerabilities such as infinite loops.

Bitcoin's scripting language is considered *stateless*, meaning that all necessary information for execution must be included within the script itself, as there is no persistent state before execution.

The validation engine relies on two types of script: a locking script and an unlocking script. Historically, the lock script has been referred to as `scriptPubKey` because it typically contained a public key or a Bitcoin address. Every Bitcoin client can verify the validity of transactions on the blockchain by performing these locking and unlocking operations. Only a valid transaction (i.e., one with a valid unlocking script) can alter the global state of the blockchain.

Bitcoin's scripting language is stack-based, meaning that operations are performed on data in the order they are pushed onto the stack. The script executes each operation from left to right. Any combination of lock and unlock scripts that results in a `OP_TRUE` value is considered valid.

The following is an example of a locking script:

```
3 OP_ADD 5 OP_EQUAL
```

The corresponding unlocking script is simply:

```
2
```

The validation software combines these scripts, resulting in the following final script:

```
2 3 OP_ADD 5 OP_EQUAL
```

Following the execution steps illustrated in Figure 2.2, the script evaluates to `OP_TRUE`, making the transaction valid.

2.3.3 Pay to Public Key Hash (P2PKH) Transactions

A widely used transaction format in Bitcoin is *Pay-to-PubKey-Hash (P2PKH)*. Here, the recipient's address is effectively the hash of their public key. The transaction includes a

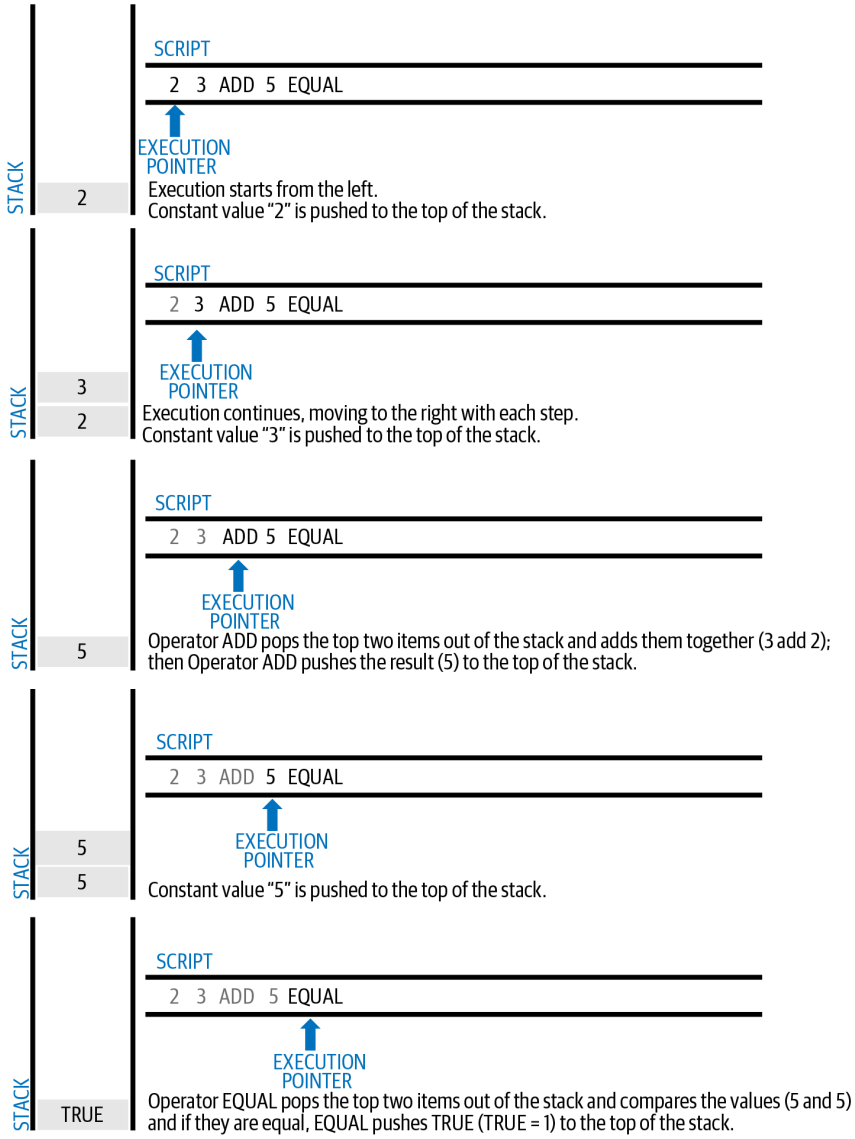

Figure 2.2: Bitcoin script validation

`scriptPubKey` specifying that only the valid signature of the holder of the corresponding public key can use the output. In more detail, the address-creation process involves:

1. Applying `SHA-256` to the public key,
2. Then applying `RIPEMD-160` to the resulting hash (often noted as `OP_HASH160`),
3. Encapsulating that value into Base58 notation to form a Bitcoin address.

As we have seen previously, in Bitcoin's scripting language, these operations are translated as follows:

```
OP_DUP OP_HASH160 <PubHash> OP_EQUALVERIFY OP_CHECKSIG
```

The combination of these cryptographic steps ensures that it is extremely difficult for an attacker to retrieve the private key from someone's address.

2.3.4 ECDSA Digital Signatures

Bitcoin implements the *Elliptic Curve Digital Signature Algorithm* (ECDSA), which enables users to prove ownership of funds without ever revealing their private key.

Since only the rightful owner can sign legally, the system prevents any unauthorized party from moving funds. This property is why ECDSA forms the core of the Bitcoin security model [14].

2.4 ECDSA for VPN Authentication

The core cryptographic mechanisms underlying Bitcoin, particularly ECDSA, can also serve as a robust authentication strategy in a VPN context. Traditionally, VPN services rely on conventional username-password pairs, sometimes supplemented with SMS-based one-time passwords or certificates issued by a trusted authority. Although these methods are familiar and easy to implement, they often require user registration or external identity verification.

In contrast, leveraging ECDSA authentication from an existing Bitcoin wallet provides a *pseudonymous* alternative and reduces dependence on centralized user databases. Here, users demonstrate their identity simply by *signing a challenge message* with the private key corresponding to a Bitcoin address that has recently sent payment to the VPN service. As long as the ECDSA signature matches the known public key, the VPN server can be confident in the user's authenticity — without collecting sensitive information.

Moreover, pseudonymous authentication using Bitcoin private keys can allow for a more fluid payment mechanism, whereby a user pays for VPN access and then proves that payment

through the signing process. This means the entire *authorization* and *value transfer* are transferred to the Bitcoin network, benefiting from the well-tested cryptographic guarantees [8]. This approach is especially appealing in use cases where users seek privacy, cross-border payments, or censorship resistance.

In the following chapters, we will illustrate how to integrate these authentication steps and how Bitcoin's existing architecture naturally supports such a VPN application. The proposed system demonstrates that it is possible to exploit well-known cryptographic primitives already present in Bitcoin to achieve a secure and pseudonymous method for VPN login.

Chapter 3

Functionalities

3.1 Authentication Flow

The authentication process takes place in the following phases (Figure 3.1):

1. **Initial Connection and Challenge:** The first access attempt fails on purpose; the VPN server rejects the user's connection but responds with a newly generated public Bitcoin address and a challenge message to be signed. This step ensures that each prospective user is provided with a unique destination address for payment and a unique message to sign.

2. **Payment and Signature Generation:** The user makes a payment transaction in a specified Bitcoin address. Then he signs a challenge message with a private key that corresponds with that address. This cryptographic step check that the user controls the funds that were just transferred.

3. **Signature Verification and Access Grant:** Upon receiving the signed message, the VPN server verifies it against the public key derived from the payment transaction. If the verification is successful, the user is granted access to the VPN service. In other words, only the entity who made the payment can generate a valid signature.

Our VPN server exclusively supports legacy Bitcoin wallets and is designed to work only with simple transactions such as P2PKH (Pay-to-Public Key Hash) on the Regtest network.

The decision to use OpenVPN instead of other protocols (such as WireGuard) was made considering that, along with the VPN protocol, it provides a suite of infrastructure software, such as Access Server. This suite allows for the implementation of VPN servers with customized authentication systems, whose operation will be further examined in the following chapter.

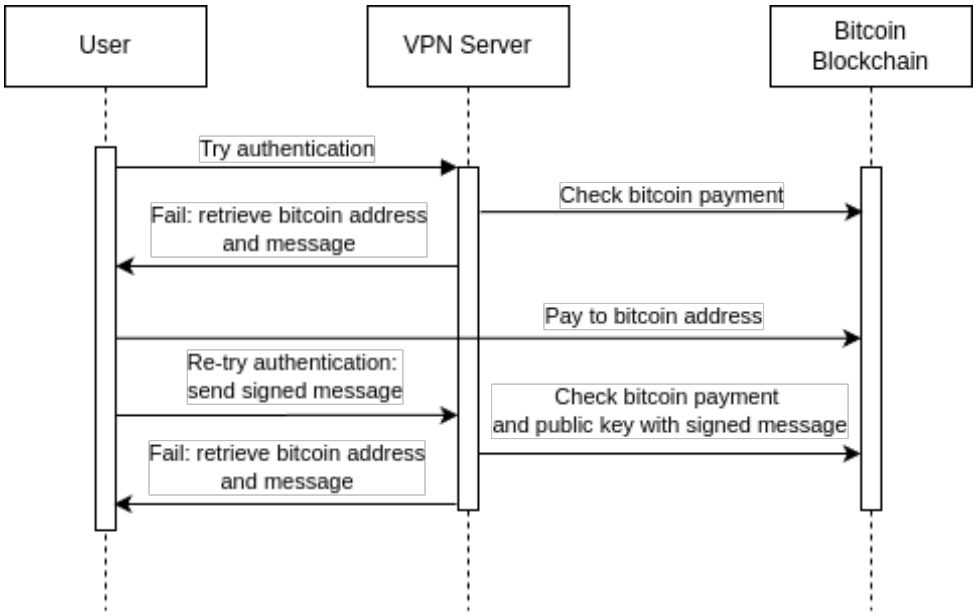

Figure 3.1: Authentication flowchart

3.2 Practical Considerations

3.2.1 Transaction Confirmation Delays

In our implementation, we do not account for real waiting times. In our case, the transaction is included in the block regardless of the transaction fee paid, the block is mined instantly, and the typical six-block confirmation wait time required for a transaction to be securely validated is also ignored.

In a realistic scenario, such as when using the Bitcoin blockchain `mainnet`, performance degradations related to transaction confirmation times could arise. The waiting time for a Bitcoin transaction can vary, but on average it takes approximately 10 minutes for a transaction to be included in a block, assuming that the transaction pays a sufficiently high fee. Additionally, it takes around 60 minutes for full confirmation, which requires six additional blocks to be mined.

A possible solution to this problem could be the implementation of a payment system based on the *Lightning Network*.

The Lightning Network is a layer-2 solution built on top of the Bitcoin protocol that enables instant payments. This protocol addresses Bitcoin's scalability issues by utilizing off-chain transactions. Payments are settled off-chain between two users, and only the final result of a series of transactions is recorded on-chain [15]. This approach not only significantly reduces transaction waiting times but also minimizes transaction fees to nearly zero, paving the way for unprecedented micro-payment capabilities. For example, a user could access the VPN service for only the necessary duration and pay on a per-minute basis, ultimately being charged only for

the actual usage time of the service.

3.3 Limitations and Simplifications of the Implementation

Our implementation serves solely as a proof-of-concept of what a real-world implementation should look like. The project does not enforce a payment threshold, which means that any amount sent to the VPN server allows the user to access the service. Additionally, the payment is a one-time action that grants indefinite access, as the system does not track the number of times a user connects. Consequently, once a payment is made and the corresponding public key is registered, the user can use the service indefinitely.

3.4 Security Properties

This project does not introduce additional security properties beyond those inherent in the Bitcoin blockchain and the ECDSA signatures.

Although the system uses robust cryptographic primitives, practical security also depends on correct implementation, secure key storage on the client side, and the reliability of the chosen Bitcoin network environment. Any weaknesses in wallet handling or in the broader infrastructure (e.g., unpatched operating systems, vulnerable network services) can reduce overall security.

3.5 Privacy Concern

The use of public keys as authentication's method offers several advantages, as discussed previously. However, at the moment of payment, the user inadvertently exposes their entire balance and transaction history to the VPN service provider. Given the public nature of the blockchain, it is possible to analyze all fund movements associated with a given address, as well as its current balance. This results in the exposure of sensitive financial information, which, while not directly linked to the user due to the pseudo-anonymity of Bitcoin addresses, still poses a privacy risk.

This issue could be directly addressed by adopting a type of transaction known as *PayJoin*. PayJoin transactions enhance privacy by obfuscating the ownership of inputs and outputs in a Bitcoin transaction. Unlike standard payments, where inputs typically belong to the sender and outputs to the recipient, PayJoin transactions mix inputs from both parties, making blockchain analysis significantly more difficult. By integrating this approach, users could pay for VPN services while reducing the exposure of their wallet balance and transaction history [16].

The following it is an example of how the PayJoin protocols works:

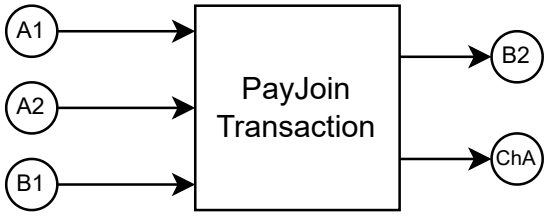

Figure 3.2: PayJoin

Bob sends the recipient's address and amount. Alice creates and signs a transaction in which she sends the specified amount to Bob's address and provides her change address to receive the remainder and then sends the transaction to Bob. Bob checks the transaction and creates a new transaction by appending his inputs to the transaction created by Alice. Then he alters the output amount and adds the coins he added by inputs to the final amount. He signs his inputs and sends this new transaction to Alice. Alice checks and signs the transaction and broadcasts it to the network. Figure 3.2 illustrates a simple form of PayJoin where ChA is Alice's change address [16].

Chapter 4

Implementation

4.1 Software and tools

This section provides an overview of the main software and libraries used within the provided Python scripts and configuration files. Since this project operates in a Bitcoin regression testing ("regtest") environment and leverages cryptographic operations, a set of specialized tools and Python libraries are required.

- **Python 3.x**: The scripts are written in Python 3. They are based on features of the modern language, so it is recommended to use Python 3.7 or higher;
- **OpenVPN Access Server**: this software serves as the backbone of the project, providing the infrastructure that allowed me to implement authentication via Bitcoin and Blockchain in a straightforward manner. It is a VPN server that integrates traditional authentication methods (such as username-password, 2FA, MFA, etc.) but also allows the use of custom Python scripts (in our case, see `pas-bitcoin.py`), known as Post-Authentication Scripts 4.1;
- **python-bitcoinlib**: A Python library for interacting with Bitcoin Core[18];

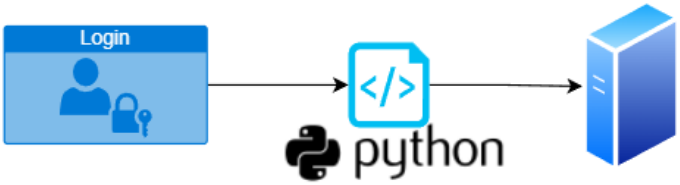

Figure 4.1: Post-Authentication Script workflow [17]

- **python-ecdsa**: The `ecdsa` library is used for elliptic curve cryptography (signing message and verifying signatures) [19];
- **base58**: The `base58` library is used for encoding and decoding operations, especially to convert private keys between the hex format and the `WIF` (Wallet Import Format) [20];
- **hashlib**: These are standard cryptographic functions (such as SHA-256, RIPEMD-160) from Python's built-in and from `ecdsa` for hashing message content and keys;
- **Bitcoin Core (`bitcoind` and `bitcoin-cli`)**: The Bitcoin infrastructure enables the operation of a network node and the creation of a testing network (such as testnet or regtest). Through its APIs, it is possible to perform transactions, manage the VPN server's wallets, and inspect the blockchain [21];
- **Nix**: Although not essential for the project's operation, it is worth mentioning the use of Nix as a package manager. It allowed me to maintain a deterministic development environment, ensuring complete control over the versions of the libraries and software used [22].

4.2 Code breakdown

The following is a concise analysis of each file and its role within this project [23], providing the pseudo-code.

4.2.1 `pas-bitcoin.py`

This script implements a Post-Authentication Script for OpenVPN by leveraging Bitcoin Core (in the `regtest` blockchain) to verify that a user has paid a specific address. It uses `ECDSA` to verify signatures (via the `verify_signature` function) against known public keys derived from user transactions, loads and unloads specific Bitcoin wallets (such as `vpn` and `user`), and relies on `Proxy` objects (from `bitcoin.rpc`) to communicate with the local Bitcoin node.

The following is the entire Python source code for the file, followed by an explanation of each significant block and function.

```
fun post_auth_cr(authcred, attributes, authret, info, crstate)
    if (VPN authentication session) then:

        Set Bitcoin regtest as default
        proxy <- init Proxy for RPC calls
```

```
// Get the dinamic response
signature <- authcred.get("response")
new_address <- proxy

// Retrieves all transaction ids
transaction_ids <- scan_transaction(proxy)

sender_pub_keys <- Init set
foreach tx_id in transaction_ids do:
    pub_key <- get_public_key(proxy, tx_id)
    Add the pub key to the set sender_pub_keys

if (signature is provided) then:
    set authentication_status to FAIL
    set client_reason to "No matching signature"

    foreach pub_key in sender_pub_keys do:
        if verify_signature(SIGN_MESSAGE, signature, pub_key) then
            set authentication_status to SUCCEED
            set client_reason to "Signature matching."

else:
    set authentication_status to FAIL
    set client_reason to "No signature provide"

return authentication_status
```

Imports and Constants.

- `pyovpn.plugin` it is a "*module to interface with Access Server's authentication mechanisms*" [24];
- `bitcoin.rpc` and `bitcoin.core` from the `python-bitcoinlib` package allow interaction with a Bitcoin Core node via RPC calls;
- `ecdsa` is utilized for ECDSA signature verification and curve operations;
- `SIGN_MESSAGE` is a constant string that each user is required to sign;

- `BITCOIN_NETWORK`, `VPN_WALLET_NAME`, `USER_WALLET_NAME` store various configuration details for Bitcoin test environments;
- The constants `WALLET_ALREADY_LOADED_ERROR_CODE` and `WALLET_ALREADY_UNLOADED_ERROR_CODE` handle specific RPC error scenarios that should not trigger any error;
- `AUTH_NULL` and `RETAIN_PASSWORD` manage how the VPN server processes client credentials and sessions [24].

`post_auth_cr(authcred, attributes, authret, info, crstate):` this is the main function that is executed for the client's authentication. It checks if the current authentication method is `session` or `autologin`, in which case it does not perform a challenge–response procedure. If the authentication session is of type VPN (the authentication attributes contain `vpn_auth`), it attempts to retrieve the user-provided signature from `authcred`; otherwise it uses the classic username-password (for instance, in the case of Web authentication). The `SelectParams` selects the Bitcoin network `regtest` and initializes a `Proxy` for JSON-RPC communication with the Bitcoin node. A new Bitcoin address is requested (`to_pay_btc_address`), so a user can pay to that address every time a new address is generated in this way, anyone cannot track VPN's transaction, improving the privacy of the owner of the service. Then, all transaction IDs from the loaded wallet are fetched with `scan_transactions` and for each transaction, it extracts the sender's public key with `get_public_key` and stores it in `sender_pub_keys`. If the client's signature was provided, the code attempts to verify that signature (`verify_signature`) against each discovered public key until it finds a match. Finally, if a valid signature is discovered, authentication succeeds; otherwise, the VPN rejects the client.

`scan_transactions(proxy):` Calls the Bitcoin RPC function `listtransactions` to obtain all transactions known to the current wallet and extracts and returns a `set` of transaction IDs (`txid`).

```
def scan_transaction(proxy):
    tx_ids <- Init set
    transaction_list <- proxy.call("listtransactions")

    foreach tx in transaction_list do:
        extracts id and adds it to tx_ids

    return tx_ids
```

`get_public_key(proxy, transaction_id):` Given a transaction ID, retrieves the raw transaction data via `getrawtransaction`. Then, checks for transaction inputs, finally decodes the first input's `scriptSig` and returns the public key is returned in hexadecimal format.

```
def get_public_key(proxy, transaction_id):
    raw_tx <- proxy.getrawtransaction(transaction_id)
    tx <- extract transaction object

    if (tx has no input) then:
        \\ The only transaction without input are coinbase reward
        return

    if (tx.input has scriptSig) then:
        pub_key <- extract second element of the scriptSig

    return pub_key
```

`verify_signature(message, signature, public_key):` converts the signature from Base64 and the public key from hexadecimal form to byte arrays. Constructs an `ecdsa` `VerifyingKey` from the provided public key bytes and uses `verify` to check if the signature is valid for the given message and it returns `True` if the signature is valid and corresponds to the provided public key, otherwise `False`.

```
def verify_signature(message, signature, public_key):
    b_signature <- convert signature from base64 to bytes
    b_pub_key <- convert pub from hex to bytes
    b_message <- convert message from string to bytes

    \\ Use ecdsa's functions and Objects to verify signature
    verifying_key <- VerifyingKey.from_string(
            b_pub_key, curve <- SECP256k1, hashfunc <- sha256
        )
    return verifying_key.verify(b_signature, b_message, hashfunc <- sha256)
```

4.2.2 `bitcoin.conf`

Configuration file for Bitcoin Core [23]. Sets the network to `regtest`, enables full transaction indexing (`txindex=1`), and configures the node to run in the background (`daemon=1`).

The `regtest` let you create bitcoins on a local blockchain for testing purposes. Allows RPC commands (`server=1`) for interaction with the Python scripts. Contains debugging parameters (`debug=1`) to facilitate detailed logs for development and testing.

The `fallbackfee=1.0` parameter sets a default fee rate when estimation is unavailable. `maxfeerate=10.0` prevents fees from exceeding a certain threshold. `maxtxfee=1000.0` caps the total fee each transaction can pay [25].

4.3 Utility scripts

Here are other scripts useful for building and testing the PAS script.

4.3.1 `verify-signature.py`

A standalone Python script that checks if a given ECDSA signature matches a message and a provided public key, using the `ecdsa` library (`VerifyingKey` from `SECP256k1`). It accepts command-line arguments in the form `<message> <signature> <public-key>` and prints whether the signature is valid or has been tampered with.

4.3.2 `pay.py`

This Python script sends Bitcoin transactions from the `user` wallet to a specified address on `regtest` using `sendtoaddress`. It imports `get_public_key` to retrieve the public key from the transaction for reference or debugging, and it also demonstrates switching between loaded wallets by unloading the `vpn` wallet and loading the `user` wallet. The output includes the transaction ID (`txid`) as well as the private key of the newly derived address for demonstration.

4.3.3 `sign-message.py`

This script signs arbitrary messages using a private key in `WIF` format, converting the `WIF` private key to a raw hexadecimal private key and then signing the message with `ecdsa`. It outputs the ECDSA signature in Base64 format for easy transport or verification and is typically invoked as `sign-message.py "Who is John Galt?" <WIF_private_key>`.

4.3.4 `get_pub_key_from_tx.py`

This script retrieves the sender's public key from a given transaction ID on the `regtest` blockchain by connecting to the Bitcoin node via the `Proxy` class and using

`getrawtransaction` in verbose mode to decode transaction information. Then it extracts the public key in hexadecimal format from `scriptSig` and prints it to the console.

Chapter 5

Conclusion

The integration of Bitcoin-based authentication for VPN access is an effective means of extending privacy and security in digital communications. This thesis has discussed the design, implementation, and testing of a VPN authentication system using Bitcoin transactions and ECDSA cryptographic signatures. By substituting traditional payment-based authentication with a cryptographic proof of payment, this system facilitates permissionless, pseudonymous, and secure access to a VPN service, without relying on centralized user credentials.

The solution deployed is within a Bitcoin `regtest` setup, thereby ensuring a controlled and deterministic test environment. The authentication protocol uses Bitcoin transactions as proof-of-access, whereby clients sign an agreed challenge message with their private key. After verifying the signature with the public key of the transaction, the VPN server grants access to the service. This method does away with centralized identity authentication, thereby significantly improving the privacy of users and minimizing the attack surface of credential storage.

Although its benefit, the suggested deployment has specific disadvantages. First, dependence on on-chain Bitcoin transactions implies inherent confirmation delays, which can potentially hinder real-time access to the VPN service.

Future work can consider implementing the Lightning Network addition, allowing instant micropayments while keeping the existing proof-of-ownership cryptographic model. Also, the existing implementation does not support payment limits or timed access control, leading to infinite access following a single payment. Support for session-based authentication associated with micropayments made at intervals can likely help improve resource utilization and balanced use of the service.

Security-wise, the solution leverages the strength of Bitcoin's cryptographic primitives like ECDSA for signing and SHA-256 for transaction integrity. However, there are still practical implementation risks, particularly in private key management on the client side. Public key exposure by blockchain transactions also entails potential privacy concerns since transaction

history is still publicly traceable.

More generally, this research contributes to the body of work on decentralized authentication methods, demonstrating the feasibility of blockchain-based access control in the absence of centralized brokers. The framework produced during this thesis can be exported to other systems besides VPNs, including authenticated login for secure services, IoT device authentication, and anonymous access control for censorship-resistant systems. In summary, this thesis has effectively demonstrated a working proof-of-concept of Bitcoin-based VPN authentication, both the advantages and the trade-offs. Future research has to be directed towards the optimization of transaction speed, user anonymity, as well as investigating alternative blockchain designs that might be able to even better balance security, usability, and decentralization. As privacy issues persist in defining the online world, decentralized authentication protocols such as the one introduced in this research provide an appealing alternative to conventional identity-based access control mechanisms.

Bibliography

[1] M. T. Hammi, P. Bellot, and A. Serhrouchni, "Bctrust: A decentralized authentication blockchain-based mechanism," in *2018 IEEE wireless communications and networking conference (WCNC)*, IEEE, 2018, pp. 1–6.

[2] A. Yazdinejad, G. Srivastava, R. M. Parizi, A. Dehghantanha, K.-K. R. Choo, and M. Aledhari, "Decentralized authentication of distributed patients in hospital networks using blockchain," *IEEE journal of biomedical and health informatics*, vol. 24, no. 8, pp. 2146–2156, 2020.

[3] P. S. Calem, C. Henderson, and J. Wang, "Who remains unbanked in the united states and why?," 2025.

[4] U. C. Bureau, *Population*, `https://www.census.gov/topics/population.html`, Accessed: February 28, 2025.

[5] A. Chaia, A. Dalal, T. Goland, M. J. Gonzalez, J. Morduch, and R. Schiff, "Half the world is unbanked," *Financial Access Initiative Framing Note*, pp. 1–10, 2009.

[6] A. Demirgüç-Kunt, L. Klapper, D. Singer, and S. Ansar, "The global findex database 2021," *World Bank Group*, 2021.

[7] *Bitcoin dada*, `https://btcdada.com/`, Accessed: February 28, 2025.

[8] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 7th. Pearson, 2016.

[9] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*, 7th. Pearson, 2016.

[10] A. Ghosh, S. Srivastava, and P Supraja, "Advancing trust in the internet: Blockchain-driven decentralized vpns leveraging smart contracts and pos as trust factor calculators," 2024.

[11] ProtonVPN, `https://protonvpn.com/`, Accessed: February 28, 2025.

[12] NordVPN, `https://nordvpn.com/`, Accessed: February 28, 2025.

[13] Mullvad, `https://mullvad.net/en`, Accessed: February 28, 2025.

[14] A. M. Antonopoulos, *Mastering Bitcoin: Programming the Open Blockchain*. O'Reilly Media, 2023, isbn: 978-1491954386.

[15] A. M. Antonopoulos, O. Osuntokun, and R. Pickhardt, *Mastering the lightning network*. " O'Reilly Media, Inc.", 2021.

[16] S. Ghesmati, W. Fdhila, and E. R. Weippl, "Bitcoin privacy-a survey on mixing techniques.," *IACR Cryptol. ePrint Arch.*, vol. 2021, p. 629, 2021.

[17] OpenVPN, *Plugins using access server's post-authentication programming hook*, `https://openvpn.net/as-docs/plugins.html`, Accessed: February 26, 2025.

[18] P. Todd, *Python-bitcoinlib*, `https://github.com/petertodd/python-bitcoinlib`, Version: 0.12.2.

[19] *Python-ecdsa*, `https://github.com/tlsfuzzer/python-ecdsa`, Version: 0.19.0.

[20] keis, *Base58*, `https://github.com/keis/base58`, Version: 2.1.1.

[21] *Bitcoin core*, `https://github.com/bitcoin/bitcoin`, Version: 27.1.

[22] E. Dolstra, *Nix*, `https://github.com/NixOS/nix`, Version: 2.24.12.

[23] L. Parolini, *Pas-bitcoin*, `https://github.com/cooparo/pas-bitcoin`.

[24] OpenVPN, *Post-authentication python script guide*, `https://openvpn.net/as-docs/post-authentication-script-reference.html`, Accessed: February 26, 2025.

[25] *Rpc api reference*, `https://developer.bitcoin.org/reference/rpc/index.html`, Accessed: February 26, 2025.