zk − qrcode: Visual Anonymous Credentials from zkSNARKs and Blockchain

Yue Zhou[1]

Australian National University
yue.zhou@anu.edu.au

Abstract. QR codes are widely used in various practical scenarios, such as opening a website, ordering meals, and checking in, due to their ease of use and rapid development. However, QR codes also pose security threats, including forgery and privacy leakage. To address these issues, we propose a novel scheme zk-qrcode based on anonymous credentials and zero-knowledge proofs. Our scheme leverages the following features: 1). Blockchain-based credential issuance: We eliminate the need for credential issuers to hold signing keys by allowing them to issue credentials to a smart contract on the blockchain, simply as maintain a bulletin board. 2). Flexible and composable identity statements: We enable users to prove complex and expressive statements over credentials without revealing any unnecessary information. 3). QR-code based interaction: We consider a scenario where the user interacts with the service provider by displaying or scanning a QR code on their mobile phone, which contains the identity proof and the access control request. We also implement and evaluate our scheme in a realistic use case of using a zk-qrcode to anonymously enter a bar. Our results show that our scheme is efficient and practical, as the access control proof generation and verification take less than 650ms.

Keywords: Anonymous credentials, acess control, Zero-knowledge proofs, Blockchain, Smart contract

1 Introduction

Privacy-preserving identification aims to balance the trade-off between identity verification and privacy protection, which is an increasingly important goal in today's internet landscape. Many public facilities require users to prove certain identity attributes-such as age, residency, or vaccination status-to access specific content or services. For instance, bars and casinos in Australia impose age restrictions and mandate users to present official IDs to verify their age. These requirements, however, pose significant risks to users' privacy and security by exposing their personal information. Privacy-preserving cryptography provides a solution to this problem through the use of anonymous credentials. These credentials enable users to prove that they meet certain access criteria (e.g., being over 18) without disclosing any additional personal information. Additionally, anonymous credentials can support complex and composite identity statements, such as proving residency to access online library resources or to petition local elected representatives. This approach not only safeguards user privacy but also ensures that necessary identity verification processes are fulfilled.

Anonymous credentials have been thoroughly investigated in the academic literature, as highlighted by studies such as those[10,30], However, their practical deployment remains limited. This discrepancy largely stems from the unrealistic assumptions underpinning most existing schemes, which fail to align with real-world identity systems. Key assumptions often include the presence of a single issuer for each identity attribute (e.g., address), the compatibility of formats used by different issuers for the same attribute, and the existence of trustworthy authorities capable and willing to maintain signing keys, verify identity attributes, and execute complex cryptographic protocols for issuing anonymous credentials. Moreover, these schemes presuppose that all necessary attribute formats for a credential are predefined and that the set of authorities for each identity attribute can be listed at the time of system initiation. These assumptions are, at best, partially met by existing anonymous credential schemes, which generally require, among other things: a single issuer per identity property, compatible formats when multiple issuers are involved, reputable authorities willing to handle signing keys and verify identity properties, predefined attribute formats, and an enumerated set of relevant authorities at system setup.

Decentralized access control schemes involve multiple parties in making access control decisions, rather than relying on a single central authority. In these systems, each resource or user may have unique access control rules or policies, or access may be granted based on the consensus of multiple entities. This approach provides several significant advantages over centralized systems. Decentralized access control brings about increased resilience, as it eliminates single points of failure. Unlike centralized systems, decisions are not reliant on one server or database, making these systems more robust against attacks and downtimes. Furthermore, decentralized systems offer greater flexibility due to their lack of dependence on predefined rules and policies. This flexibility allows them to adapt more effectively to changing requirements and conditions. Another advantage is enhanced privacy. Decentralized systems do not depend on a central authority that has access to all sensitive user information, thereby offering a higher degree of privacy. Additionally, the decision-making process is distributed among multiple parties, which promotes fairness and accountability. Lastly, decentralized systems can scale more efficiently than centralized ones, as they are not confined by the limitations of a single server or database for decision-making. Overall, these benefits make decentralized access control an appealing and practical approach for contemporary, adaptive, and secure systems.

Blockchain technology has the potential to transform the management of access control within distributed anonymous credential systems. By functioning as a distributed ledger, a blockchain operates across a network of multiple locations rather than relying on a centralized database. Each node in the network maintains a complete copy of the blockchain, ensuring transparency and integrity. This decentralization prevents any single entity from controlling or manipulating the data. Nodes independently verify transactions and reach consensus on the validity of each block, thereby establishing a system-wide agreement on data integrity. The blocks within a blockchain contain transaction data that cannot be altered or backdated without detection. Any tampering attempts are identified and rejected by the nodes, which contrasts sharply with traditional record-keeping methods that can be easily manipulated without detection. This immutability enhances the security and reliability of data storage and verification. Smart contracts, which are programmable codes running on the blockchain, enable the creation of decentralized, distributed systems for managing access and permissions securely and reliably. One significant advantage of using blockchain for access control in anonymous credential systems is its decentralized nature, which eliminates single points of control or failure issues. This decentralization makes the system more resistant to attacks, such as distributed denial of service (DDoS) attacks, which aim to disrupt normal operations by overwhelming the network with traffic. Furthermore, blockchain-based access control systems offer flexibility, allowing them to adapt easily to changing requirements or circumstances. For instance, if the system needs to be temporarily shut down for maintenance, a smart contract could automatically revoke access for all users until the maintenance is complete. Additionally, a blockchain-based access control system can be more efficient and cost-effective than traditional systems, as it automates many tasks that would otherwise need to be performed manually. In summary, blockchain technology, with its decentralized ledger and smart contract capabilities, provides a robust framework for managing access control in distributed anonymous credential systems, offering enhanced security, flexibility, and efficiency.tasks that would otherwise need to be performed manually.

The widespread adoption of mobile devices equipped with cameras makes them ideal platforms for deploying services in a user-friendly and secure manner. Mobile devices can also act as consolidation platforms that avoid the inconveniences and security threats of using multiple special-purpose devices and tokens, such as identity cards, one-time password generators, etc. We argue that simple designs, combined with software security principles and prudent engineering practices, can result in multipurpose, service-oriented mobile platforms that are more secure and privacy-compliant than our current habits. The QR (Quick Response) Code represents a two-dimensional Barcode(or a type of matrix barcode). A typical QR code comprises black squares arranged in a square grid against a white background, designed to be read by imaging devices like cameras. The encoding capacity of QR codes varies based on their version, characterized by distinct numbers of modules comprising the black and white dots forming the code. These codes possess omnidirectional readability, facilitated by position detection patterns located at three of their four corners. Additionally, they exhibit error correction capabilities, enabling data restoration even if the QR code sustains partial dirt or damage. Data within QR code is encoded in both vertical and horizontal directions, with the highest version (version 40) accommodating encoding of up to 7089 numerical only characters-several hundred times more data compared to conventional barcodes. Generally, a smartphone serves as the primary tool for scanning QR codes, functioning to display the code and convert it into a practical format. For instance, the

QR code might be transformed into a standard URL for a website, eliminating the necessity for users to manually type the URL into a web browser. The substantial data encoding capacity and robust error correction features of QR codes are highly relevant to our approach, making them particularly well-suited for the intended application scenarios.

1.1 Existing Work Limitations

Verifiable anonymous credentials are becoming increasingly popular among individuals and businesses worldwide due to their robust user privacy features and their ability to address several major issues associated with the current identity management system. There are several issues with current traditional anonymous credential verification systems

- **Increased Risk of Data Breaches**. Centralized data management systems store large amounts of user information, increasing the risk of data breaches.
- **Lack of User Control over Personal Data:**. Organizations store vast amounts of personal data, leaving users without control over who has access to their information, where their documents are stored, and how they are being used. Physical credentials, such as driver license often disclose more information than necessary during verification.

With digital credentials (e.g., PDFs), verifying the origin and authenticity of documents involves: 1) Confirming that the issuing organization has the authority to issue the credential (e.g., a driver's license provider). 2) Ensuring that the person submitting the credential is its rightful owner. 3) Checking that the credential is valid, not expired, modified, or revoked. This verification often requires manual contact with the issuing organization, making the process long, tedious, and expensive. Verifiable anonymous credentials simplify the process by facilitating automatic and secure verification, thereby increasing both efficiency and reliability. In decentralized anonymous credential systems[19],credentials are recorded in a transparency log. This log can be managed in various ways: it can be centralized and subject to audit, distributed among cooperating parties, or operated in a decentralized manner using a Byzantine fault-tolerant system or blockchain technology. This method effectively addresses a major challenge in credential issuance: the requirement of holding signing keys. However, the specific protocol faces limitations in both performance and operational efficiency. For instance, it necessitates that all clients possess the complete list of issued credentials. Moreover, existing protocols for both anonymous and non-anonymous credentials typically assume that the same party is responsible for verifying claimed identity properties and signing cryptographic credentials. This dual responsibility creates the need for a single trusted party to perform two distinct tasks: verifying identity attributes and managing signing keys. This linkage often complicates deployment and is frequently unnecessary. In practice, many applications of anonymous credentials do not involve identity attributes that require verification by a trusted party for credential issuance. For instance, an issuer-less Privacy Pass-like construction[16] can issue Sybil-resistant anonymous tokens via blockchain payments, eliminating the need for trusted parties to verify user identities or sign credentials. Even when a trusted entity is needed to verify identity attributes, it is not necessary to trust an additional party to manage the cryptographic keys. Instead, minimal trust assumptions can be achieved by replacing the signing process with a list maintained by a centralized authority or a distributed bulletin board. When a trusted party is responsible for both verifying identity attributes and issuing credentials, maintaining a list is safer than managing signing keys. In current anonymous credential schemes, key compromise is often undetectable, requiring rekeying and reissuing to address breaches. However, using a list for issuance allows for detectable and easily reversible compromises, even if the list is maintained by a single party.

1.2 Our Contribution

We propose the `zk-qrcode` application tailored for camera-equipped devices. Unlike the previous anonymous credential research works using traditional blind signatures based proof systems, our approach applies general purpose zero-knowledge proofs. It is able to support flexible access criteria and even allows this criteria to be created after the credential issuance. This application is designed to issue users with anonymous credentials, allowing them to present required credential information via QR codes in a manner that protects privacy when interacting with verifiers. Our `zk-qrcode` provides publicly auditable and verifiable credentials, the issued credential list is maintained by the smart contracts on the public blockchain platform. Notably, our scheme also supports on-chain verification of credentials and access criteria. The functionality extends to parse these QR codes and execute designed functions on a smart contract, enabling verification of user compliance with access criteria.

1.3 More Related Work

In this section, we present more related work in the literature.

- **Anonymous Credentials.** There are several research works and industrial solutions related to anonymous credentials. After Chaum[14] initials the idea, a lot of work had been done [13,7,5,3,11,31] with continuously efficient and flexible anonymous credentials which have already been applied in plenty of practical applications[16,1]. These schemes have been generalized and extended to offer more complex properties, including the issuance of hidden attributes, updatable[4], selective disclosure of attributes[30], k-show[2], rate-limit[7,12], and issued by a decentralized organization[19,21]. Blockchains are being leveraged as a verifiable data registry to improve both on-chain and off-chain certificates. By utilizing general-purpose zero-knowledge proofs and Blockchain, our approach might be capable of fulfilling all these advanced properties.
- **On-chain Access Control.** Lastly, all interactions with the system's smart contracts are recorded as transactions on the blockchain, ensuring a more secure system with the capability to verify and maintain the linked information. Egala[17] proposed a novel blockchain-based architecture for a decentralized Electronic Health Record (EHR) system that automates services using smart contracts while maintaining strong security and privacy. Li[26] presented a fine-grained access control system for Vehicle Ad Hoc Network (VANET) data using blockchain technology to replace third-party service providers for user identity management and data storage. Recent works, such as iden3[22] and Polygon ID[28], enable private on-chain access control solutions using zk-SNARKs to solve the distributed digital ID problems.
- **Decentralized Identity.** Another area of research in decentralized identity has gained significant attention in both academic and industrial fields, with substantial efforts being directed toward standardization. A notable contribution in this field is the Decentralized Identifier (DID) introduced by the W3C. A DID is a unique digital address that represents an individual's identity on the internet. It allows people to control their own personal information without relying on centralized authorities or third-party providers, providing a more secure and private way of managing digital identities. DIDs are typically associated with cryptographic concepts, such as the use of private and public key pairs. The goal is to delegate the management of identifiers to users, enabling an identifier to serve as an identification factor for any subject, including individuals, organizations, and data models. DIDs are particularly beneficial for applications that require self-administered and cryptographically verifiable identifiers, such as those in personal, organizational contexts. The proofs of data associated with a Decentralized Identifier (DID), such as signatures and hashes, are stored on a blockchain. Data recorded on a blockchain are immutable, and their integrity and decentralization are preserved by all participants who maintain a copy. Additionally, decentralization ensures the complete availability of the data.

2 More Related Works

- **Anonymous Credentials.** There are several research works and industrial solutions related to anonymous credentials. After Chaum[14] initials the idea, a lot of work had been done [13,7,5,3,11,31] with continuously more efficient and flexible anonymous credentials which have already been widely applied in plenty of real-world applications[16,1]. Initially, presenting a credential allowed little more than (unlinkably) showcasing a signed token associated with a user's pseudonym. However, these schemes have been generalized and extended to offer more sophisticated properties, such as the issuance of hidden attributes, rate-limiting[7,12], k-show[2], updatable[4], efficient selective disclosure of attributes[30], and issued by a decentralized organization[19,21]. Blockchains are being leveraged as a verifiable data registry to improve both on-chain and off-chain certificates. By utilizing general-purpose zero-knowledge proofs and Blockchain, our approach is capable of fulfilling all these advanced properties.
- **On-chain Access Control.** Lastly, all interactions with the system's smart contracts are recorded as transactions on the blockchain, ensuring a more secure system with the capability to verify and

maintain the linked information. Egala[17] proposed a novel blockchain-based architecture for a decentralized Electronic Health Record (EHR) system that automates services using smart contracts while maintaining strong security and privacy. Li[26] presented a fine-grained access control system for Vehicle Ad Hoc Network (VANET) data using blockchain technology to replace third-party service providers for user identity management and data storage. Recent works, such as iden3[22] and Polygon ID[28], enable private on-chain access control solutions using zk-SNARKs to solve the distributed digital ID problems.

- **Decentralized identity.** Another area of research in decentralized identity has garnered attention in both academic and industrial circles, including efforts toward standardization. A notable contribution in this field is the Decentralized Identifier (DID) introduced by the W3C. A DID is a unique digital address that represents an individual's identity on the internet. It allows people to control their own personal information without relying on centralized authorities or third-party providers, providing a more secure and private way of managing digital identities. DIDs are typically associated with cryptographic concepts, such as the use of private and public key pairs. The goal is to delegate the management of identifiers to users, enabling an identifier to serve as an identification factor for any subject, including individuals, organizations, and data models. DIDs are particularly beneficial for applications that require self-administered and cryptographically verifiable identifiers, such as those in personal, organizational contexts. The proofs of data associated with a Decentralized Identifier (DID), such as signatures and hashes, are stored on a blockchain. Data recorded on a blockchain are immutable, and their integrity and decentralization are preserved by all participants who maintain a copy. Additionally, decentralization ensures the complete availability of the data.

3 Overview

3.1 Security Definition

The security definitions for our scheme are presented through an ideal functionality depicted in Fig 2, aligning with the standard security properties of anonymous credentials: unforgeability, correctness, and unlinkability. Furthermore, this framework introduces an extra security property known as session binding. This property ensures that credential presentations are intrinsically linked to the channel or session in which they are used, thereby effectively mitigating the risk of replay attacks.

We consider the possibility that all issuers, verifiers, and the majority of users may act maliciously and collude with one another in the threat model of our system. To ensure the anonymity property, we adopt the standard requirement that at least two honest users with valid issued credentials must exist within the system. Additionally, we assume the presence of a reliable method for involved parties to reach consensus on the list of issued credentials. We argue that $\texttt{zk} - \texttt{qrcode}$ maintains security against computationally-bounded adversaries who may collude with any other parties, including users, verifiers, and issuers. Our analysis assumes the security of our scheme under the condition of a single issuer, without loss of generality, as any corrupted issuer could potentially compromise the integrity of the credential list.

3.2 Anonymous Credentials

A traditional anonymous credential system typically involves two participants: users and organizations. The structure of our proposed scheme consists of three entities: User (who owns the anonymous credential), Issuer (organization that has authority to issue the anonymous credentials. e.g. government department issues a driver license), and Verifier (who validates the anonymous credential). The framework of the involved entities shows in Fig 1. Each user owns one or multiple credentials from one or multiple organizations. Each credential is a commitment $\texttt{Cm}(\texttt{nymk}, \texttt{rk}, \texttt{attrs}; \texttt{rand})$, where $\texttt{nymk}$ is the pseudonym key of user in a specific organization. $\texttt{rk}$ is a private random value used to generate rate limited tokens to solve the $k - show$ problem. $\texttt{attrs}$ is a set of hidden attributes belonging to each user, such as date-of-birth, gender..., and $\texttt{rand}$ is another random value binds to each commitment. Because of the nature hidden property of commitment scheme, all the values in the credential commitment stays private, nothing will be revealed unless users disclose partial personal attributes deliberately. In a simple word, an anonymous credential is able to represent any type of information related to the user. An issuer noted by $\texttt{O}$ stands for an organization who can issue a credential $\texttt{Cred}$

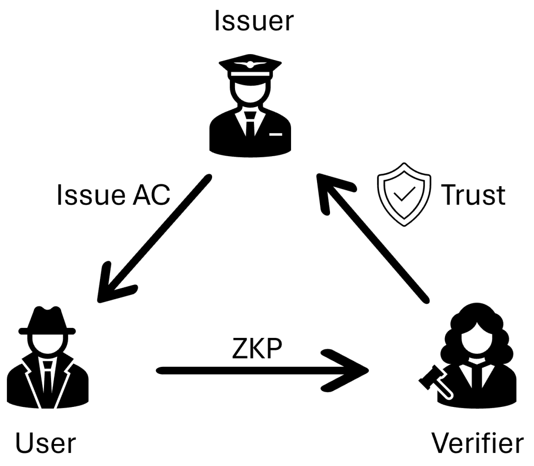

Fig. 1: An illustration of three entities involved in our anonymous credential system.

if the $\texttt{Cred}$ exists on $\texttt{O}$'s credential list $\texttt{CL}$. In our model, we use the Verkle tree to store $\texttt{CL}$s. Every issuer has a list of issuance criteria $\texttt{IC}$ that user has to satisfy all the requirements to have their $\texttt{Cred}$ get issued. For instance, if the driver's license number provided by the user in the $\texttt{Cred}$ matches the record stored in the government's database, and this satisfies the issuer's criteria, the issuer will first add the $\texttt{Cred}$ to their list. Subsequently, the issuer will issue an authentication path α, which provides membership attestation, proving that the user meets certain requirements. A user can present a credential by displaying a zero-knowledge proof via a QR code, asserting that they possess a valid issued credential with attributes that meet the verifier's access requirements. The verifier can then scan the QR code using a camera-equipped device, granting access to the user upon successful verification. Note that we assume that verifier and issuer must trust each other.

The full algorithms of the anonymous credential system are defined below.

- $\texttt{Setup}(1^\lambda) \rightarrow \texttt{pp}$. Inputs a security parameter λ, then generates several system parameters
- $\texttt{CredReq}_\texttt{U}(\texttt{pp}, \texttt{IC}, \texttt{attrs}, \texttt{aux}) \rightarrow \texttt{cred}$. Inputs the system parameters, issuer's criteria $\texttt{IC}$, user's attributes and session context $\texttt{aux}$, then requests a credential $\texttt{cred}$ under the criteria $\texttt{IC}$.
- $\texttt{CredIssue}_\texttt{I}(\texttt{pp}, \texttt{IC}, \texttt{CL}, \texttt{cred}) \rightarrow (\texttt{CL}', \eta)$. Inputs the system parameters, issuer's criteria $\texttt{IC}$, issuer's credential list $\texttt{CL}$, $\texttt{cred}$, It then determines whether to grant the requested credential $\texttt{cred}$, if successful, added it to the credential list $\texttt{CL}$ and updated the list to $\texttt{CL}'$, finally returns the corresponding issuance attestation α to user.
- $\texttt{CredDisplay}_\texttt{U}(\texttt{pp}, \alpha, \eta, \texttt{CL}, \texttt{cred}, \texttt{aux}) \rightarrow (\pi, \texttt{aux})$. Inputs the system parameters, the authentication path α, credential list $\texttt{CL}$, $\texttt{cred}$ and $\texttt{aux}$, then shows that an issued $\texttt{cred}$ meets the access criteria ζ, return the proof π(which will be encoded into a QR-code).
- $\texttt{CredVerify}_\texttt{V}(\texttt{pp}, \alpha, \texttt{CL}, \pi, \texttt{aux}) \rightarrow \texttt{ok}?$. Inputs the system parameters, the authentication path α, credential list $\texttt{CL}$, proof π and $\texttt{aux}$, then verify if the credential is valid or not.
- $\texttt{CredRemove}_\texttt{I}(\texttt{pp}, \texttt{CL}, \texttt{cred}) \rightarrow \texttt{CL}'$. Inputs the system parameters, credential list $\texttt{CL}$, user's $\texttt{cred}$, then removes the $\texttt{cred}$ from the list and update the list to $\texttt{CL}'$.

As shown in Fig 2

3.3 Design Features

Our proposed method to build the anonymous credentials has several important features.

- **Customized access criteria**. Our approach supports plenty of features commonly discussed in the existing literature on anonymous credentials. These features include hidden-attribute credentials, rate limiting and equality and expiry checks. Notably, our approach guarantees that any violation of the rate limit, such as credential sharing, leads to the identification and optional revocation of the credential. Moreover, our system allows users to selectively disclose information, enabling them to reveal only the necessary data for a specific use case rather than presenting their entire credential.

Fig. 2: A generic ideal functionality of $\texttt{zk-qrcode}$

1

- $\texttt{Setup}_{\texttt{U}}(1^{\lambda})$:
  1. Generates the system required parameters to perform the following functions.
  2. Creates issuance criterion $\texttt{IC}$ for requesting and issuing a credential.
  3. Creates an access criterion $\zeta \in \texttt{AccessCriteria}$ as a precondition for displaying the corresponding credential.
- $\texttt{CredReq}_{\texttt{U}}(IC, attrs)$:
  1. The function samples random $\texttt{cred}, \texttt{r}$, generates the commitment and corresponding opening.
  2. Creates the specific credential using the input parameter $\texttt{attrs}$, as well as the random value r, $\texttt{UserCreds}_{\texttt{U}} \leftarrow \texttt{GenerateCreds}(\texttt{attrs}, \texttt{r})$.
  3. Generates auxiliary data $\texttt{aux}_{\texttt{u}}$ to justify the issuance of credential.
  4. Store a pair of parameters $(\texttt{cred}, \texttt{aux}_{\texttt{u}})$ and will be used in $\texttt{IssueGrant}_{\texttt{I}}$.
  5. If the previous steps fails, then returns 0 and terminates the function.
  6. Else, returns α and a tuple of parameters $(\texttt{cred}, \texttt{r}, \alpha)$ will be sent to user U.
- $\texttt{CredIssue}_{\texttt{I}}(IC, cred, aux_u)$:
  1. If the information user provided is satisfied the issuance criteria $IC(attrs, aux_u) = 1$, then constructs the credential $\texttt{UserCreds}_{\texttt{U}}[\texttt{cred}] \leftarrow \texttt{GenerateCreds}(\texttt{attrs}, \texttt{r})$.
  2. The issuer has the privilege to deny any $\texttt{cred}$ if he makes a proper refusal statement S to user U.
  3. Otherwise, sample a random α, and modify the $\texttt{IssuedCredsList}[\alpha] = \texttt{cred}$.
  4. Send the updated $\texttt{IssuedCredsList}$ to issuer I, and send α to user U.
  5. Broadcast to all involved parties that a new $\texttt{cred}$ has been issued.
- $\texttt{CredDisplay}_{\texttt{U}}(\alpha, cred, r, aux_r)$:
  1. This function samples a random value s, and obtains the random value $r^{'}$ using $\texttt{UserCreds}_{\texttt{U}}[\texttt{cred}]$.
  2. First check whether $\texttt{r} \overset{?}{=} \texttt{r}^{'}$. If equality doesn't hold, then terminates this function directly.
  3. Then assign $\alpha^{'} = \alpha$ and creates the proof using the credential $\texttt{cred}$, $(\alpha^{'}, \texttt{cred}, \texttt{r}, \texttt{aux}_{\texttt{u}}) \leftarrow \texttt{DisplayProof}_{\texttt{U}}[\texttt{s}]$.
  4. Finally, return a pair of parameters $(\texttt{s}, \texttt{aux}_{\texttt{u}})$ to user U.
- $\texttt{CredVerify}_{\texttt{V}}(\alpha, s, aux_u)$:
  1. Obtain the required parameters $(\alpha^{'}, \texttt{cred}, \texttt{r}, \texttt{aux}_{\texttt{u}}^{'}) \leftarrow \texttt{DisplayProof}_{\texttt{U}}[\texttt{s}]$, $\texttt{cred}^{'} \leftarrow \texttt{issuedCredsList}(\alpha)$ and $(\texttt{IC}, \texttt{r}^{'}, \texttt{attrs}) \leftarrow \texttt{UserCreds}_{\texttt{U}}[\texttt{cred}]$.
  2. Then checks whether $\alpha \overset{?}{=} \alpha^{'}$ and $\zeta \in \texttt{AccessCriteria}[\texttt{V}]$, whether $\texttt{cred} \overset{?}{=} \texttt{cred}^{'}$, $\texttt{r} \overset{?}{=} \texttt{r}^{'}$, $\texttt{aux}_{\texttt{u}} \overset{?}{=} \texttt{aux}_{\texttt{u}}^{'}$.
  3. If the user U is malicious, and the verifier V is honest, then checks whether $\alpha(\texttt{attrs}, \texttt{aux}_{\texttt{u}}) \overset{?}{=} \texttt{1}$.
  4. If any above checks fails, return **false** to verifier V.
  5. Else, return **true** to V.
- $\texttt{CredRemove}_{\texttt{I}}(cred)$:
  1. Given the authentication path α, find the corresponding $\texttt{cred} \leftarrow \texttt{IssuedCredsList}[\alpha]$.
  2. If not found, did nothing and terminate this function.
  3. Else, assign empty value to replace $\texttt{cred} : \texttt{IssuedCredsList}[\alpha] = \texttt{nil}$.
  4. Finally broadcast to all involved parties that $\texttt{cred}$ has been removed.

- **Publicly auditable issuance**. In our implementation, anyone can access the list of credentials and verify both their issuance and the reasons for it. This transparency contrasts sharply with traditional signature-based credential systems, where it is impractical to list and audit each credential signed with a given key. Importantly, the credential data is not stored in plaintext on a public ledger. Instead, it is hashed and organized within a Verkle tree. This method ensures that only essential information is revealed. A zero-knowledge proof is utilized to confirm the consistency of the committed credential data on the public ledger with the requirements during the verification process. This approach balances transparency and privacy, allowing for public auditing without compromising sensitive information.
- **Trust-free issuance**. When untrusted nodes can evaluate the list of valid credentials, the prover can compile this list from network broadcasts and provide it to the verifier during credential presentation. Blockchain technology inherently mitigates the problem of a malicious verifier altering the prover's view to insert a credential that, while validated through the issuance process, was not broadcast to other involved parties. By maintaining a consistent view among numerous nodes within an adversarial network, the blockchain ensures the integrity and reliability of credential validation. This consistency is effectively maintained by ensuring high network connectivity and using computational proofs of work to create a hash chain. This decentralized verification process prevents any single entity from compromising the integrity of the credential validation.
- **Easy witness management**. Our method uses Verkle trees for issuing the credential due to the dynamic nature of the issued credentials list. Users are required to maintain an up-to-date witness to verify the inclusion of their credentials in the issuer's list. Proving possession of a credential entails demonstrating its membership in a Verkle tree. Moreover, users can maintain an up-to-date witness by downloading updates of logarithmic size from the Verkle tree.
- **Efficient credential revoke**. In certain situations, it may be necessary to revoke an issued credential, thus invalidating its authentication path α through more complex mechanisms than

simple checks like age verification. Many current methods require costly asymmetric cryptographic operations for each revocation. Schemes like EPID[8] involve credential presentations that perform work proportional to the number of revoked private keys. Other approaches use RSA accumulators, requiring recomputation of accumulator witnesses for each revocation, incurring costs linear to the number of revocations. Although more efficient methods exist[29], they often involve revoking credentials by replacing the leaf node in the Merkle tree with a hash of nil, thus increasing system complexity by introducing a Merkle forest. In contrast, our method simplifies the revocation process. By merely deleting the credential from its Verkle tree, the approach incurs only logarithmic costs proportional to the number of issued credentials.

4 Preliminaries

We briefly present the extra cryptographic preliminaries used in this chapter, before explaining the construction of our `zk-qrcode` application in the subsequent sections.

Commitment Scheme. A commitment scheme is a mapping $\mathtt{Cm} : \mathcal{M}^n \times \mathcal{R} \to \mathcal{C}$ from a (vector) message space $\mathcal{M}^n$ and a random mask space $\mathcal{R}$ to a commitment space $\mathcal{C}$. A commitment scheme is *homomorphic*, if for any $\vec{\mathsf{m}}_1, \vec{\mathsf{m}}_2 \in \mathcal{M}^n, \mathsf{r}_1, \mathsf{r}_2 \in \mathcal{R}$:

$$\mathtt{Cm}(\vec{\mathsf{m}}_1; \mathsf{r}_1) \cdot \mathtt{Cm}(\vec{\mathsf{m}}_2; \mathsf{r}_2) = \mathtt{Cm}(\vec{\mathsf{m}}_1 + \vec{\mathsf{m}}_2; \mathsf{r}_1 + \mathsf{r}_2)$$

Definition 1 (Schwartz-Zippel). *Given two univariate polynomials:*

$$f[X] = (X - a_1)...(X - a_n), g[X] = (X - b_1)...(X - b_n)$$

, Schwartz-Zippel says that for a random input $\gamma \leftarrow \mathbb{F}$ the probability that different polynomials $f[X] \neq f[X]$ evaluate to the same value $f[\gamma] = g[\gamma]$ is very low.

$$Pr[f[\gamma] = g[(]\gamma] \mid f[X] \neq g[X] \leq \frac{max(d_f, d_g)}{\mid \mathbb{F} \mid}$$

It holds if every root of $f[X]$ and $g[X]$ is shifted by a randomly chosen constant $\delta \leftarrow \mathbb{F}$.

4.1 Cryptographic Building Blocks

zk-SNARKs The development of zk-SNARKs (Zero-Knowledge Succinct Non-interactive Arguments of Knowledge) is rooted in decades of research into interactive proof systems [20,15] and probabilistically checkable proofs [18]. While the theoretical feasibility of zk-SNARKs was demonstrated through the application of classical PCP theorems and Verkle trees, practical implementations of zk-SNARKs have only emerged in recent years, beginning with the Pinocchio protocol[**?**]. There are two main types of practical zk-SNARKs:

1. *Trusted Setup*: This category requires a trusted entity to produce specific public parameters that the prover employs to generate a proof [6,32]. These parameters are based on secret information, ensuring that, without this secret, it remains computationally infeasible to validate a false statement. The inclusion of a trusted setup eases the zk-SNARK verification process, leading to more concise proofs and efficient verification.
2. *Transparent Setup*: This category eliminates the need for a trusted entity to establish public parameters [25,32]. Instead, these parameters can be generated without relying on any secret information, allowing for a transparent setup. Transparent setups offer significant benefits for applications where it is crucial to avoid relying on a central party for unbiased statement verification, such as in decentralized finance. However, it is important to acknowledge that these zk-SNARKs often generate larger proof sizes and incur higher verification costs. As a result, they are less practical for implementation on blockchain platforms.

Verkle Tree Verkle trees, first introduced by[24], are conceptually similar to Merkle trees, which are well-known and currently implemented in Ethereum. Both Merkle and Verkle trees feature nodes that can be in one of three states: 1) an empty node, 2) a leaf node containing a key-value pair, and 3) an intermediate node with a fixed number of children (the tree's width), where the value of the intermediate node is the hash of its children. The primary structural difference is that Verkle trees are much wider. In a Merkle tree, proving membership involves presenting all sibling nodes along the path from the root to the target node. This proof encompasses every node that shares a parent with any node within that path. Conversely, in a Verkle tree, the proof is independent of the sibling nodes. This is achieved using vector commitments or polynomial commitments to compute an inner node from its children. Fig. 5 shows an example of Verkle tree. We possess a proof for one or more values within a Verkle tree. The core component of this proof includes the intermediary nodes along the path to each specific node. For every node provided, it is necessary to prove that it is indeed a child of the node above it and occupies the correct position. In the example of a single-value proof, the proof for a value in the tree does not require any sister nodes, it only needs the path itself along with a few brief proofs to connect each commitment along the path to the next. So for the given example in Fig. 5, we required proofs to demonstrate the following:

– *P3*: The proof 3 that the key: *8bd* node is at the position d child of the intermediate node with prefix: *8b*.
– *P2*: The proof 2 that the intermediate node is at the position *b* child of the intermediate node with prefix: *8*.
– *P1*: The proof 1 that the intermediate node with prefix *8* is at the position of *8* child of the root.

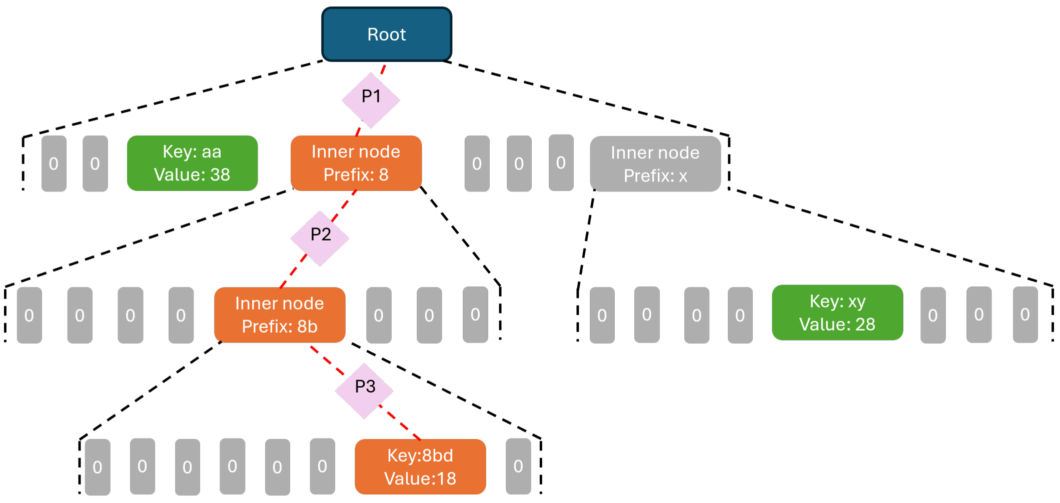

Fig. 3: An illustration of given example Verkle tree with width 8.

Bilinear Pairing A useful property of elliptic curve groups is bilinear pairing. A *bilinear pairing* is a mapping $\mathtt{e} : \mathbb{G}_1 \times \mathbb{G}_2 \mapsto \mathbb{G}_T$, where $\mathbb{G}_1, \mathbb{G}_2$ are cyclic groups of prime order p, such that

$$\mathtt{e}\langle \mathtt{g}^x, \mathtt{h}^y \rangle = \mathtt{e}\langle \mathtt{g}, \mathtt{h} \rangle^{xy}$$

for any $\mathtt{g} \in \mathbb{G}_1, \mathtt{h} \in \mathbb{G}_2$, and $x, y \in \mathbb{F}_p$. There are pairing-friendly groups that admit efficient bilinear pairing [9].

A key consequence of pairing is that given $(\mathtt{g}^x, \mathtt{h}^y, \mathtt{k})$, a verifier can verify whether $\mathtt{k} \overset{?}{=} \mathtt{g}^{xy}$ by checking the following:

$$\mathtt{e}\langle \mathtt{k}, \mathtt{h} \rangle \overset{?}{=} \mathtt{e}\langle \mathtt{g}^x, \mathtt{h}^y \rangle = \mathtt{e}\langle \mathtt{g}, \mathtt{h} \rangle^{xy} = \mathtt{e}\langle \mathtt{g}^{xy}, \mathtt{h} \rangle \tag{1}$$

By the computational Diffie-Hellman assumption, the above verification does not need to reveal (x, y), which may be used to represent some private data.

KZG Polynomial Commitment A concrete realization of a polynomial commitment scheme is KZG polynomial commitment scheme [23]. Lemma 2 is a basic fact about factoring a polynomial.

Lemma 2. *Given a Laurent polynomial $f[X]$ and value z, then the polynomial $f[X]-f[z]$ is divisible by $X-z$, namely, $f[X]-f[z]=(X-z)\cdot q[X]$ for some Laurent polynomial $q[X]$. Intuitively, it is because $X=z$ is a root of $f[X]-f[z]$.*

KZG polynomial commitment scheme is a concrete polynomial commitment scheme that can be verified efficiently with constant time complexity. KZG polynomial commitment scheme is shown to satisfy correctness, knowledge soundness, and computational hiding under the computational Diffie-Hellman assumption [23].

Plonk We describe a universal updatable trusted-setup zkSNARK schema, Plonk[**?**], which has a great performance improvements compared to general-purpose zkp protocols widely used before, e.g. Groth16[**?**] and SONIC[27]. We provide a high-level overview of Plonk's functionality.

– *Arithmetization*: The Plonk system is not capable of directly proving general statements in real-world scenarios. Instead, it transforms practical problems into a format composed of arithmetic circuits. These arithmetic circuits consist of two types of gates: addition gates and multiplication gates. The process of arithmetization decomposes a general computation into a system of a set of independent polynomials.
– *Constraint*: A general computation's set of polynomials called the constraint system. Every polynomial in that set is constrained to a value and each polynomial is of the same form. A standard Plonk constraint polynomial shows below:

$$Q_l x_l + Q_r x_r + Q_m x_l x_r = Q_o x_o + c$$

, where Q_l, Q_r, Q_o, $Q_m \in \{0,1\}$ are boolean selectors, x_l, x_r, $x_o \in \mathbb{F}$ are values used in the arithmetic, $c \in \mathbb{F}$ is an optional constant used to assign constraint system values to a constant public input.
– *Polynomial Commitments*: A univariate polynomial commitment scheme enables a prover to commit to a univariate polynomial in advance as a secret and later open evaluations at specific points with a proof. This proof demonstrates that the evaluated polynomial matches the committed one. Such a scheme ensures the prover's honesty and prevents cheating. Plonk applies the KZG polynomial commitment.

5 Construction

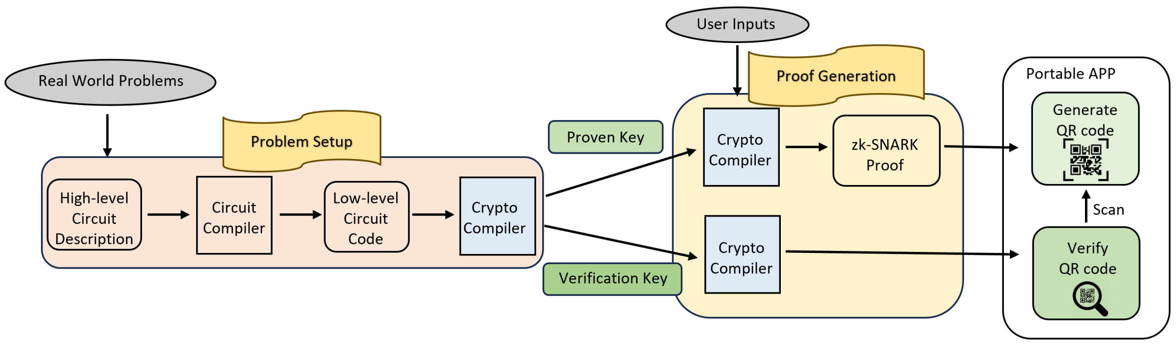

Fig. 4: An illustration of the foundational concept of converting a real-world problem into a zk-SNARK proof within the `zk-qrcode` framework.

In this section, we introduce a privacy protection framework that integrates anonymous credentials, zero-knowledge proofs, and QR codes. Our approach utilizes a two-step verification process, encompassing both user identity authentication and the validation of specific required associated information. Together with these components, we develop an effective solution to support privacy-preserving QR code verification, as illustrated at a high level in Fig 6, as well as a low level idea of our

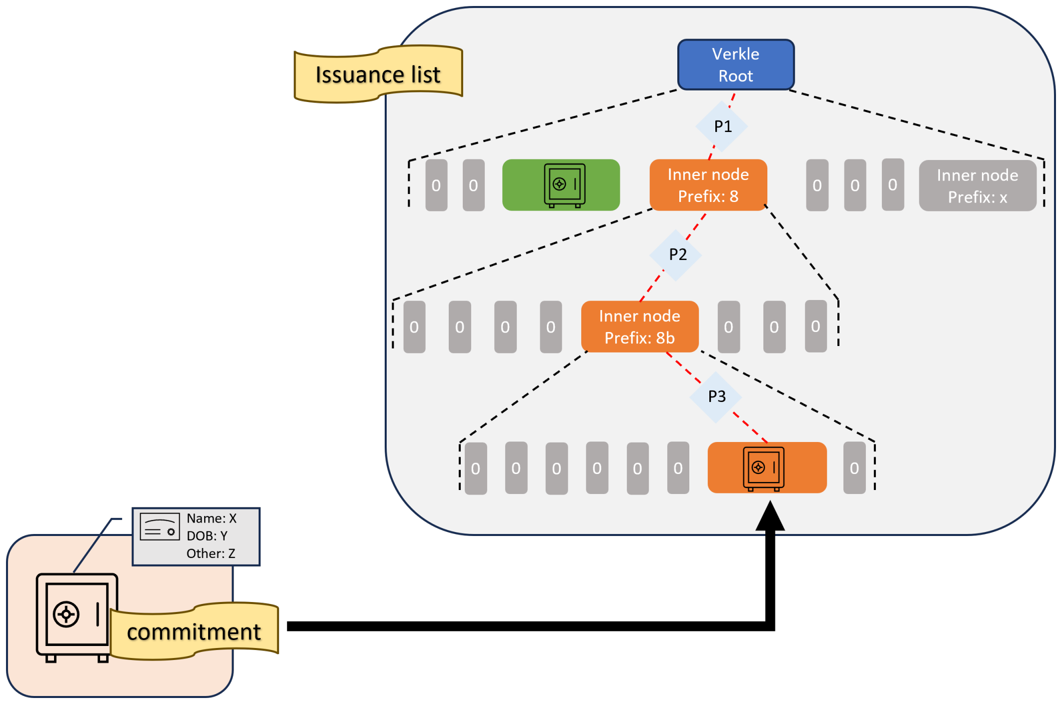

Fig. 5: An illustration of an anonymous credential in `zk-qrcode` is demonstrated by adding the credential to a Verkle tree, provided it meets the specified issuance criteria.

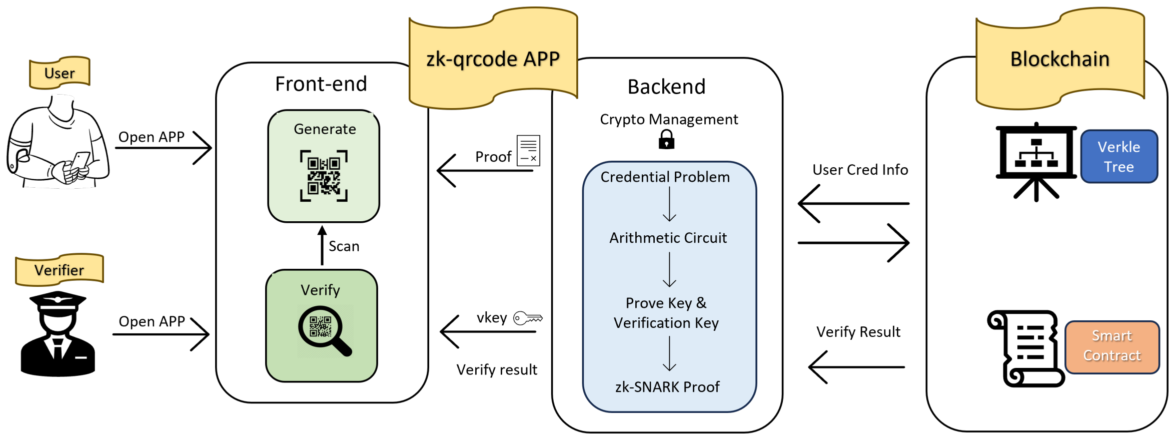

Fig. 6: An illustration of our solution for privacy-preserving access criteria in-person verification via QR code.

solution in Fig 4. We assume a list of credentials information stored in the Verkle tree maintained by a blockchain, as well as smart contracts containing specific verification methods, as shown in Fig 5. Our system provides several sets of functionalities: `CredReq`, `CredIssue`, `CredDisplay`, `CredVerify` and `CredRemove`. A user is granted a credential through the issuance process, wherein the credential is added to the issuance list upon request and approval. The user can subsequently use this credential to demonstrate compliance with specific access criteria. The application encodes the credential information into a zero-knowledge proof composed of polynomial commitments, which is then converted into a QR code containing mainly cryptographic parameters. Finally the verifier scans the QR code to complete the in-person interaction for anonymous credential proof presentation and verification. The process of utilizing user's credential typically has two steps: first, providing a membership proof to confirm the credential's inclusion in the issuance list. Second, demonstrating that the committed attributes satisfy specific access criteria. In our `zk` − `qrcode` application, the Verkle tree is employed to verify membership, while range proofs are used to ensure compliance with access criteria. We now provide an overview of the components within `zk-qrcode`, and a full description of our protocols' functionalities shows in Fig. 7, 8 and 9.

Fig. 7: `zk-qrcode` concrete construction part 1 for system operations

1

- `Setup`$(1^{\lambda}, d)$:
  1. Set a large prime order p from curve bn256, let $\mathbb{G}$ be an order-q subgroup of $\mathbb{Z}_{l}^{*}$, initial bilinear group bg.
  2. Construct a committer $C = new(d)$, where d is the domain size of the Verkle tree, also known as the width of the nodes.
  3. Let Cir_{VT} represents the arithmetic circuit of public inputs $(VT_{root}, \texttt{cred})$, where VT is a Verkle tree with node width of d, VT_{root} is the commitment of the root. An authentication path α attesting to $\texttt{cred} \in V$
  4. Generate a one-time for all crs, $crs = KZG10.crs(bg, d)$
  5. Set a access criteria θ for some attributes meet the requirement. Set an issuance criteria ζ for the information user provided meet the requirement.
  6. Let Cir_{α} represents the arithmetic circuit of public inputs $(VT_{root}, cred, aux)$, which asserts the relation: $\theta(nymk, rk, attrs, aux) = 1 \wedge cred$ opens to $(nymk, rk, attrs)$; Let Cir_{ζ} represents the arithmetic circuit of public inputs $iaux$, which asserts the relation: $\zeta(attrs, iaux) = 1 \wedge cred$ opens to $(nymk, rk, attrs)$
  7. Let $pp := (bg, crs)$
  8. Return pp.
- `CredRemove`$(pp, VT, \texttt{cred})$:
  1. Search $cred \in VT$. Terminate the function if not found.
  2. Update the Verkle tree $VT^{'} = VT.remove(cred)$.
  3. Return $VT^{'}$

Fig. 8: `zk-qrcode` concrete construction part 2 for anonymous credential

1

- `CredReq`$(pp, attrs)$:
  1. Randomly pick commitment nonce r, pseudonym key $nymk$, and rate key rk.
  2. Calculate the commitment $\texttt{cred} = \texttt{Commiter.com}(\texttt{nymk}, \texttt{rk}, \texttt{attrs}; \texttt{r})$.
  3. Calculate the witness $\omega = (r, nymk, rk, attrs)$.
  4. Generate the proof $\pi_{\zeta} = Plonk.Prove(crs, (cred, iaux), \omega)$
  5. Send $cred$ to issuer.
  6. Update the root of modified credential tree $VT^{'}$ and the corresponding Verkle tree authentication path α, contains the information that $cred \in VT^{'}$.
- `CredIssue`$(pp, VT, cred)$:
  1. Verify $Plonk.Verify(crs, pi_{\zeta}, (cred, iaux))$
  2. Continue the function upon pass the verification, or terminate the function if failed.
  3. Add the latest approved $cred$, $VT^{'} = VT.Insert(cred)$.
  4. Create the authentication path $\alpha = VT^{'}.AuthPath(cred)$
  5. Send α to user U.

Fig. 9: `zk-qrcode` concrete construction part 3 for access criteria

1

- `CredDisplay`$(pp, crs, \alpha, VT, cred, \omega, aux)$:
  1. Generate the credential proof, $\pi_{VT} = Plonk.Prove(crs, (VT_{root}, cred), \alpha)$, then create the corresponding QR-code, $qr_{VT} = QR.Gen(\pi_{VT})$
  2. Generate the access control proof, $\pi_{\alpha} = Plonk.Prove(crs, (VT_{root}, cred, aux), \omega)$, then create the corresponding QR-code, $qr_{ac} = QR.Gen(\pi_{\alpha})$.
  3. Let $\hat{qr} = (qr_{VT}, qr_{ac})$, Display the QR-codes $\hat{qr}$.
- `CredVerify`$(pp, crs, VT, \hat{qr})$:
  1. Scan the QR-codes $\hat{qr}$ by camera, get the parsed result: $(\pi_{VT}, \pi_{\alpha}) = QR.Read(\hat{qr})$.
  2. Verify the QR-codes $Plonk.Verify(\pi_{VT}, \pi_{\alpha}, crs)$.
  3. Accept upon pass the verification, or reject if failed.

5.1 Verkle Tree

Rather than employing the widely-used Merkle tree for accumulating credentials and proving membership, `zk-qrcode` utilizes the Verkle tree. The membership proof asserts that $cred \in VT$ for a Verkle tree VT. This approach offers significant performance improvements over the traditional Merkle tree. The key advantage of Verkle trees is their efficiency in proof size. In our model, the credential shown by the user is essentially a membership proof in a Verkle tree, verifying whether the node provided by the user is a valid node within that Verkle tree. Our system permits the issuer I to modify the list of issued credentials whenever necessary. This capability ensures that users can always keep their witness updated, thereby verifying their credential's membership within the tree.Modifying the tree only requires updating the intermediate commitments at each level to reflect changes from the leaf to the root. We aim to enhance efficiency by leveraging the homomorphic property of polynomial commitments. Given a polynomial commitment $C = \texttt{Cm}(\texttt{P})$, we can compute $C' = \texttt{Cm}(\texttt{P} + \texttt{Q})$ by setting $C' = C + \texttt{Cm}(\texttt{G})$. In our `zk-qrcode` we use the Verkle tree VT with the following functions. We store data elements as key-value pairs, with the key and value both being 32-byte strings.

– $VT.Insert(k, v) \rightarrow VT^{'}$ This function takes a key-value pairs, insert into the available position in Verkle tree VT and returns a updated Verkle tree $VT^{'}$.
– $VT.Remove(k) \rightarrow VT^{'}$ This function removes a value v from a valid Verkle tree VT by its key k and returns a updated tree $VT^{'}$.
– $VT.VerifyPath(k) \rightarrow \alpha$ This function creates a membership proof α to prove that $k \in T$. α corresponds directly to the height of the tree.

5.2 QR Codes

To demonstrate our approach, we developed a proof-of-concept application called `zk-qrcode`. This demo uses QR codes to transmit zero-knowledge proofs via the visual channel provided by camera-equipped devices. The highest version of QR codes (version 40) can encode up to 7089 numerical only characters, which is sufficient to encode an entire zk-SNARK proof for an arithmetic circuit. Fig 10 illustrates a QR code from the user who is entering a pub to verify he has a valid credential and is over 18 years old. The QR code has error correction level L (Low), and contains a zero-knowledge proof generated by Plonk plus other insensitive necessary information. For the access criteria proven QR code, it embeds the Plonk range proofs, which. It contains important information shows below:

– **Transcript**: The required **transcript** for Plonk includes the commitments, which are actually points(a pair of value x, y) on the elliptic curve, along with several field elements required to complete the verification process.
– **Protocol name**: e.g. *plonk.* The application is able to support variable protocols.
– **Curve name**: e.g. *bn128.* The application is able to support variable curves if the curve is compatible with use on a smart contract.
– **Public signal**: the public parameters if available.

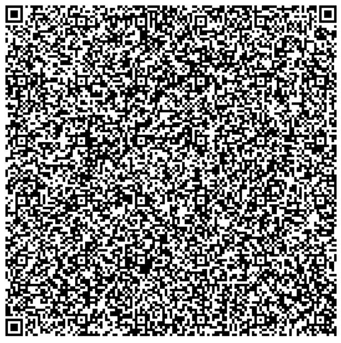

Fig. 10: An example QR code containing Plonk age proof information generated by the user before entering a bar.

5.3 On-Chain Operation

Our zk-qrcode system operates on a publicly accessible decentralized platform for distributing the credential list. We have implemented this by deploying the credential list and verification process as a smart contract on blockchain. To issue a credential, the issuer submits an issuance request to the smart contract. This method enables auditors to download the complete list, reconstruct a local Verkle tree, and use zero-knowledge proofs to verify the correct issuance of each credential. While any party can independently audit the entire list, users also have the option to trust another entity to update and verify the credential list promptly on their behalf. Additionally, users can obtain their credential's authentication path and the latest root from the smart contract at any time. We also enable verifiers to perform CredVerify on the smart contract, demonstrating the feasibility of full decentralization of our scheme. Note that in this chapter, we primarily present the feasibility of our zk-qrcode. The primary objective of this research is not to reduce verification costs, despite the high gas expenses associated with executing smart contracts. Typically, on-chain verification gas costs can be mitigated by a widely studied method: batching verification, which can be a focus of our future work.

5.4 Extension Features

Since each node in the Verkle tree can store up to 32 bytes data, the attributes and extra information have a flexible storage. In our zk-qrcode construction:

- k-show: The rate key (rk) is utilized for producing rate-limit tokens. To do this, the user generates a random seed (rk) and incorporates this value into the commitment of *cred*. In scenarios involving one-time show credentials, this seed can be directly disclosed. For k-show credentials, where k is decided by the issuer or verifier. the user evaluates a pseudorandom token from the Verifiable Random Function VRF on the seed combined with a secret counter sc, that is $token_{rate} = VRF(i||sc)$, providing a provable and secure method of showing the credential $sc < k$ times.
- Hidden and Flexible Attributes: It is easy to add extra attributes since the node has enough storage. Such as a date attribute *expire* to assert the credential is expired or not, just simply check whether $expire > Time.now$. All the attributes $attrs = (a_0, ..., a_m) \in \mathbb{Z}_l$ will be encoded.

5.5 NIZK Setup

For the proof generation and verification process, we employ the Plonk polynomial commitment scheme [?]. Plonk necessitates an updatable one-time trusted setup to produce a set of parameters known as the common reference string (CRS). This setup is both "universal and updatable". Instead of requiring a separate trusted setup for every program one wants to prove(e.g. Groth16[?]), there is a single trusted setup for the entire scheme, which can then be used with any program up to a predefined maximum size. The distributed setup for Plonk CRS has been effectively addressed through multiparty computation setup ceremonies, multiple parties can participate in the trusted setup process, ensuring its security as long as at least one participant is honest. In addition to the total number of participants does not need to be predetermined, new participants can add themselves at any point. This design facilitates a large number of participants in the trusted setup, enhancing its practical safety. These ceremonies were widely verified and used in cryptocurrency.

5.6 Proof of Security

We assert that zk-qrcode maintains security against computationally-bounded adversaries who can statically corrupt users, verifiers, and even issuers, with the potential for arbitrary collusion among these parties. Our primary strategy involves demonstrating that for each adversary $\mathcal{A}$ attacking the credential system in the real world, we can create an ideal-world adversary $\mathcal{S}$ to against the ideal-world system. The objective is to ensure that the transcript of $\mathcal{A}$ interacting with the real system is computationally indistinguishable from the transcript generated by $\mathcal{A}$ interacting with $\mathcal{S}$. For this proof, we assume that the zk-SNARK tools employed include an efficient extractor and simulator, with parameters generated through a trusted setup process. It is important to note that in the random oracle model, this assumption holds true for the Fiat-Shamir proofs we use, provided that the proofs are executed sequentially. For simplicity and without loss of generality, we assume the security of our protocol under a single issuer, recognizing that any corrupted issuer could potentially compromise the entire credential list.

We define $\mathcal{F}$ as the ideal functionality that corresponds to the algorithms illustrated in Fig 2, and S as the system's simulator. Initially, the simulator $\mathcal{S}$ runs the `Setup` phase to acquire the necessary trapdoors for simulating and extracting zero-knowledge proofs during `Setup`. $\mathcal{S}$ maintains a mapping table that links credentials from the simulated real-world protocol to their ideal counterparts in $\mathcal{F}$. This table is updated whenever a real-world object needs to be created for an ideal-world one, or vice versa.

Since every message exchanged in our security protocol was encoded by zero-knowledge proofs, the simulator is able to extract information from messages produced by adversaries. (e.g., during `CredIssue` or `CredDisplay`). Afterwards, it locates the matching ideal-world credentials in its table and forwards the requests to the ideal functionality. When dealing with honest interactions with the ideal functionality, the simulator replicates the adversary's perspective of the real-world protocol. This is achieved by simulating zero-knowledge proofs based on random commitments and pseudorandom function outputs for rate limiting. There are two specific scenarios that need special consideration. First, since both honest and corrupted issuers can deny credential issuance, the simulator's table should be updated only when issuance is successful. Second, given that both honest and corrupted parties have the ability to revoke credentials, the simulator must ensure that revocations are properly synchronized.

The security of `zk-qrcode` relies on several key foundational assumptions. Firstly, it assumes that *Plonk* operates as a perfectly zero-knowledge and simulation-extractable protocol within the Algebraic Group Model. Secondly, it requires the Poseidon hash function to be collision-resistant and function as a secure pseudorandom function (PRF). Additional assumptions are made that follows the standard requirements: Schnorr signatures with the property of unforgeable, and Pedersen commitments with computationally binding and perfectly hiding under the discrete logarithm assumption. Given these conditions, the adversary's perspective of the real-world protocol is indistinguishable from that of a simulated environment in which honest parties interact with the ideal functionality $\mathcal{F}$. As a result, the `zk-qrcode` implementation, as detailed in Fig 7, 8 and 9 maintains security against malicious adversaries capable of static corruption.

6 Case Studies

We provide two practical scenarios for `zk-qrcode`. Our application demonstrates that `zk-qrcode` is capable of supporting privacy-preserving identity verification in real-world scenarios, even when non-trivial access criteria are involved.

– **Case 1: Are you a student?** Discount prices, such as those for amusement park entry tickets and public transportation fees, are typically available to students upon presenting a valid student ID. In this context, we demonstrate how `zk-qrcode` can securely verify a student's status while keeping personal identity details confidential, such as the student ID, school name, and date of birth. The process involves the student pressing the "I am a student" button, which uses `CredDisplay` to generate a QR code asserting their student status. The clerk then press the "Scan & Verify" button to use `CredVerify` to scan the QR code, receiving a response that confirms the validity of the student's status. The underlying constraints for this interaction are $occupation \in student \wedge expiry > today$. Additionally, to further enhance verification, a feature can be implemented where, after scanning the QR code, a photo of the user is displayed on the verifier's phone. This ensures the QR code is linked to the correct individual. Implementing this feature requires adding an extra constraint to verify the hash value of the photo.
– **Case 2: Are you over 18?** In Australia, entering a pub or casino requires presenting a valid photo ID to verify that the individual is over 18. Typically, this involves showing a driver's license or passport, which reveals personal details such as name, gender, address, and date of birth. Our goal in this scenario is to protect the user's privacy at the maximum possibility, the only revealing information is that they possess an unexpired ID and are over 18. In this setup, both the user and the pub bouncer or casino security would have the `zk-qrcode` application on their phones. The user would press the "I am over 18" button to use `CredDisplay`, generating a QR code with the age assertion. The bouncer would then press the "Scan & Verify" button to use

`CredVerify`, scan the QR code displayed on the user's phone, and receive a response indicating whether the user is over 18. Unlike checking the user's driver license or passport, this interaction discloses only the minimal necessary information. The constraints behind this interaction are $dob + 18 \leq today \wedge expiry > today$.

Table 1: `zk-qrcode` Anonymous credential evaluation results. Domain size is the node width of the Verkle tree, the larger,the more information each node can contain.

Domain size	CRS Generation	Credential Proof Generation	Credential Verification	Credential Proof size
16	0.0025s	6.27s	5.44s	1.31KB
32	0.005s	7.78s	6.85s	1.44KB
64	0.012s	9.34s	8.31s	1.58KB
128	0.023s	11.08s	9.71s	1.71KB

Table 2: `zk-qrcode` case study evaluation results. `ACDisplay` is the time it takes to generate the QR code to prove the user is a student/age over 18. `ACVerify` is the time it takes to verify the corresponding QR code

Case	ACDisplay	ACVerify	Proof size	Verification key size	Gas cost(wei)
Student	0.63s	0.46s	2.20KB	1.99KB	258136
Age	0.64s	0.47s	2.19KB	1.99KB	258136

7 Performance Evaluation

In this section, we analyze the efficiency of the anonymous credential proven and verification, QR code generation, the computation cost of the QR code verification, and the communication cost. All the experiments were executed on an Intel i7-10700KF CPU computer, running Windows 10. We deploy our `zk-qrcode` application on iPhone 15 pro for emulating the interactions of the prover and verifier. We implement our smart contract in Solidity and deploy our contract on the test Ethereum network, set the security parameter $\lambda = 128$. Plonk proof was constructed over alt-BabyJubjub BN128 elliptic curve. We use the Poseidon hash function to generate the challenges. Table 1 shows the performance of the anonymous credential system. We conducted the experiments under different domain sizes. Both the credential proof generation and verification time are within reasonable time. Note that the credential proof only needs to be generated once within a valid period (e.g. one year). Table 2 gives the performance of access control. It takes users 640 ms to generate an occupation assertion QR code on the phone, and the verifier can verify the assertion in only 470 ms. Verify each QR code on the smart contract costs around 258K gas. The price of gas is highly volatile. As of late November 2023, Ethereum gas prices range from around 25 to 30 Gwei(1 Gwei is 10^{-9} ETH), and 1 ETH is about \$2062 USD. So verifying one QR code costs about \$12 - \$14USD. Notice that the gas cost can be further optimized, we only provide an experiment result that doesn't reflect the actual cost in the production environment.

8 Conclusion

In this chapter, we construct a QR code based distributed anonymous credential and access control application, `zk-qrcode`. This application is secure in the random oracle model under standard cryptographic assumptions. Our approach integrates QR codes, zero-knowledge proofs, anonymous credentials, and smart contracts, resulting in efficient and reliable identification and access criteria verification system. This system is ideal for various scenarios requiring physical interactions between provers and verifiers. The empirical evaluation results show good performance on both communication and computation costs. Despite these results, our study presents several avenues for future investigation. First, We have only applied Plonk on `zk-qrcode`, the approach we develop requires a trusted setup, as such, implementing other proof systems without a trusted setup or with a universal setup is highly possible. Additionally, optimizing the verification process within smart contracts remains an open challenge. Given the extreme high cost associated with smart contract execution, there exists potential for minimizing verification expenses to enable practical deployment. This optimization represents a promising area for future research.

References

1. Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al.: Hyperledger fabric: a distributed operating system for permissioned blockchains. In: Proceedings of the thirteenth EuroSys conference. pp. 1–15 (2018)
2. Baldimtsi, F., Lysyanskaya, A.: Anonymous credentials light. In: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. pp. 1087–1098 (2013)
3. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and noninteractive anonymous credentials. In: Theory of Cryptography: Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008. Proceedings 5. pp. 356–374. Springer (2008)
4. Blömer, J., Bobolz, J., Diemert, D., Eidens, F.: Updatable anonymous credentials and applications to incentive systems. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 1671–1685 (2019)
5. Boneh, D., Boyen, X.: Short signatures without random oracles. In: International conference on the theory and applications of cryptographic techniques. pp. 56–73. Springer (2004)
6. Boneh, D., Drake, J., Fisch, B., Gabizon, A.: Efficient polynomial commitment schemes for multiple points and polynomials. Cryptology ePrint Archive (2021)
7. Brands, S., Légaré, F.: Digital identity management based on digital credentials (2002)
8. Brickell, E., Li, J.: Enhanced privacy id from bilinear pairing. Cryptology ePrint Archive (2009)
9. Buchanan, W.J.: Cryptography. River Publishers (2017)
10. Camenisch, J., Dubovitskaya, M., Haralambiev, K., Kohlweiss, M.: Composable and modular anonymous credentials: Definitions and practical constructions. In: Advances in Cryptology–ASIACRYPT 2015: 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29–December 3, 2015, Proceedings, Part II 21. pp. 262–288. Springer (2015)
11. Camenisch, J., Dubovitskaya, M., Haralambiev, K., Kohlweiss, M.: Composable and modular anonymous credentials: Definitions and practical constructions. In: Advances in Cryptology–ASIACRYPT 2015: 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29–December 3, 2015, Proceedings, Part II 21. pp. 262–288. Springer (2015)
12. Camenisch, J., Hohenberger, S., Kohlweiss, M., Lysyanskaya, A., Meyerovich, M.: How to win the clonewars: efficient periodic n-times anonymous authentication. In: Proceedings of the 13th ACM conference on Computer and communications security. pp. 201–210 (2006)
13. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Advances in Cryptology—EUROCRYPT 2001: International Conference on the Theory and Application of Cryptographic Techniques Innsbruck, Austria, May 6–10, 2001 Proceedings 20. pp. 93–118. Springer (2001)
14. Chaum, D.: Security without identification: Transaction systems to make big brother obsolete. Communications of the ACM **28**(10), 1030–1044 (1985)
15. Cormode, G., Mitzenmacher, M., Thaler, J.: Practical verified computation with streaming interactive proofs. In: Proceedings of Innovations in Theoretical Computer Science Conference. pp. 90–112 (2012)
16. Davidson, A., Goldberg, I., Sullivan, N., Tankersley, G., Valsorda, F.: Privacy pass: Bypassing internet challenges anonymously. Proceedings on Privacy Enhancing Technologies (2018)
17. Egala, B.S., Pradhan, A.K., Badarla, V., Mohanty, S.P.: Fortified-chain: a blockchain-based framework for security and privacy-assured internet of medical things with effective access control. IEEE Internet of Things Journal **8**(14), 11717–11731 (2021)
18. Fortnow, L., Rompel, J., Sipser, M.: On the power of multi-power interactive protocols. In: Annual Conference on Structure in Complexity Theory. pp. 156–161 (1988)
19. Garman, C., Green, M., Miers, I.: Decentralized anonymous credentials. Cryptology ePrint Archive (2013)
20. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: Interactive proofs for muggles. In: ACM Symposium on Theory of Computing. p. 113–122 (2008)
21. Halpin, H.: Nym credentials: Privacy-preserving decentralized identity with blockchains. In: 2020 Crypto Valley Conference on Blockchain Technology (CVCBT). pp. 56–67. IEEE (2020)
22. iden3: On-chain verification contracts (2022), `https://github.com/iden3/contracts/tree/master/contracts/validators`

23. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: International Conference on Theory and application of Cryptology and Information Security. pp. 177–194 (2010)
24. Kuszmaul, J.: Verkle trees. Verkle Trees **1**(1) (2019)
25. Lee, J.: Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In: International Conference on Theory of Cryptography. pp. 1–34 (2021)
26. Li, H., Pei, L., Liao, D., Chen, S., Zhang, M., Xu, D.: Fadb: A fine-grained access control scheme for vanet data based on blockchain. IEEE Access **8**, 85190–85203 (2020)
27. Maller, M., Bowe, S., Kohlweiss, M., Meiklejohn, S.: Sonic: Zero-knowledge snarks from linear-size universal and updatable structured reference strings. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS). pp. 2111–2128 (2019)
28. polygon: Polygon id contrats (2024), `https://github.com/0xPolygonID/contracts`
29. Rosenberg, M., White, J., Garman, C., Miers, I.: zk-creds: Flexible anonymous credentials from zksnarks and existing identity infrastructure. In: 2023 IEEE Symposium on Security and Privacy (SP). pp. 790–808. IEEE (2023)
30. Sonnino, A., Al-Bassam, M., Bano, S., Meiklejohn, S., Danezis, G.: Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers. arXiv preprint arXiv:1802.07344 (2018)
31. Tan, S.Y., Groß, T.: Monipoly—an expressive q-sdh-based anonymous attribute-based credential system. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 498–526. Springer (2020)
32. Thaler, J.: Proofs, arguments, and zero-knowledge (2022)