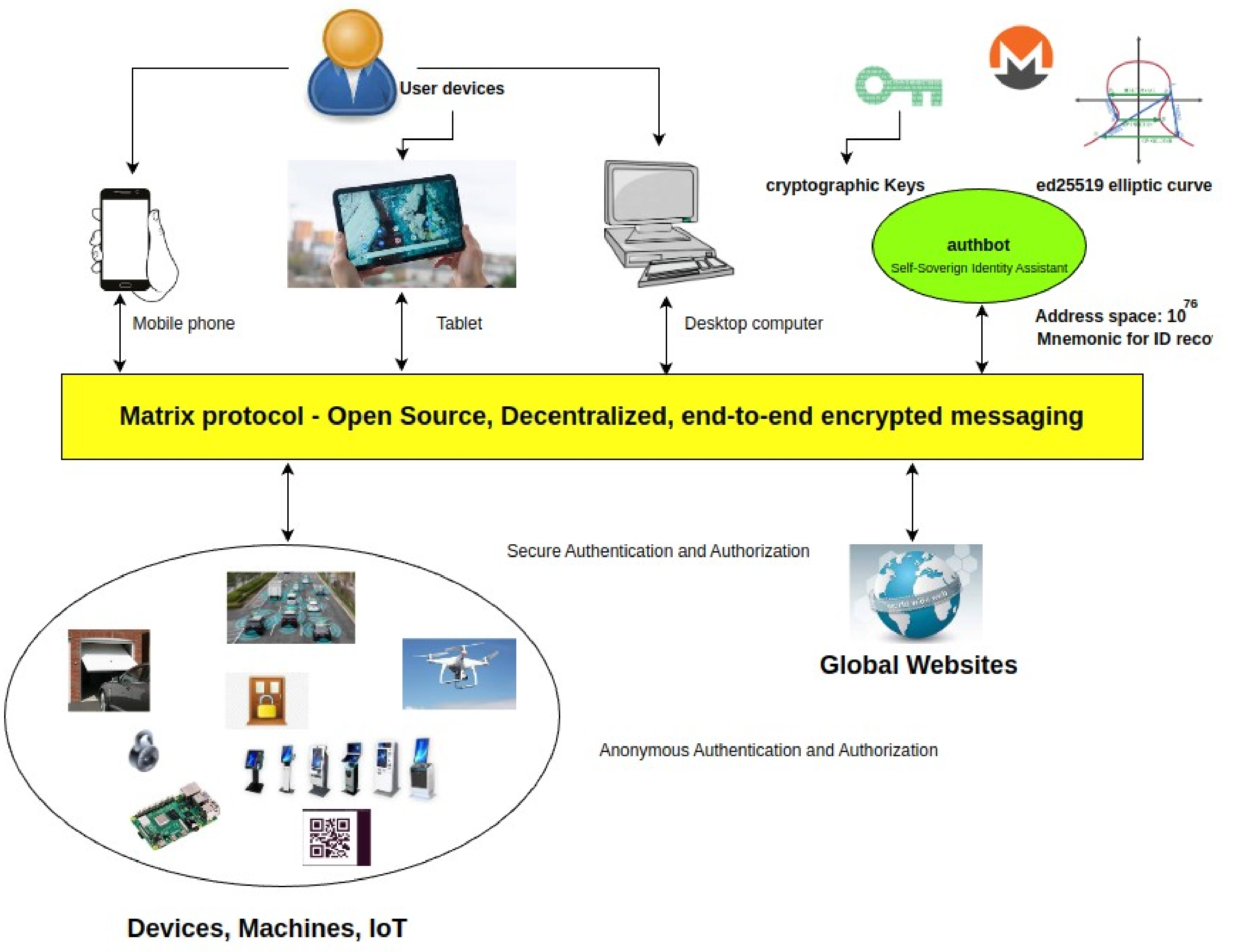
DRAFT

MoneroAuth Whitepaper

May 19, 2023

@dougbebber:matrix.org
dougbebber@protonmail.com


User devices
cryptographic Keys
ed25519 elliptic curve
authbot
Self-Soverign Identity Assistant
Mobile phone
Tablet
Desktop computer
Address space: 10^76
Mnemonic for ID reco
Matrix protocol - Open Source, Decentralized, end-to-end encrypted messaging
Secure Authentication and Authorization
Global Websites
Anonymous Authentication and Authorization
Devices, Machines, IoT

Executive Summary

MoneroAuth is a protocol that was originally designed to provide an open standard for secure authentication to digital resources, such as websites, in order to obsolete less secure approaches like user names/passwords and to eliminate the dependence on Big tech and their OAuth dominance. The MoneroAuth protocol was then extended to provide for secure authentication for access to physical resources, such as physical locks.

Because of the complexity involved in public key cryptography and key management, the initial versions of the MoneroAuth software proved to be difficult to use and presented a significant barrier to mass adoption. Because of this, we put the project on hold for a few years while we explored ways to make the protocol easy-to-use with a focus on the user-experience. We believe we have made significant advancement and are now ready to introduce the protocol to the public as an open source, permissionless, secure protocol for self-sovereign identity purposes.

To enable the ease-of-use, we have integrated the Matrix protocol, that provides decentralized, end-to-end encrypted messaging services, with the Monero protocol, to deliver a Self-Sovereign Identity Assistant that we call authbot.

The result is an easy-to-use solution for authentication to digital and physical resources, even anonymous authentication to digital and physical resources. The protocol enables custom authorization that, when combined with authentication, allows for the specification of customized policies for resource management.

With MoneroAuth solutions, users are able to logon to multiple web sites, open physical doors, and control physical devices all with a single open source, permissionless, digital ID.

The MoneroAuth protocol is open source and available on github.

Self-Sovereign Identity (SSI)

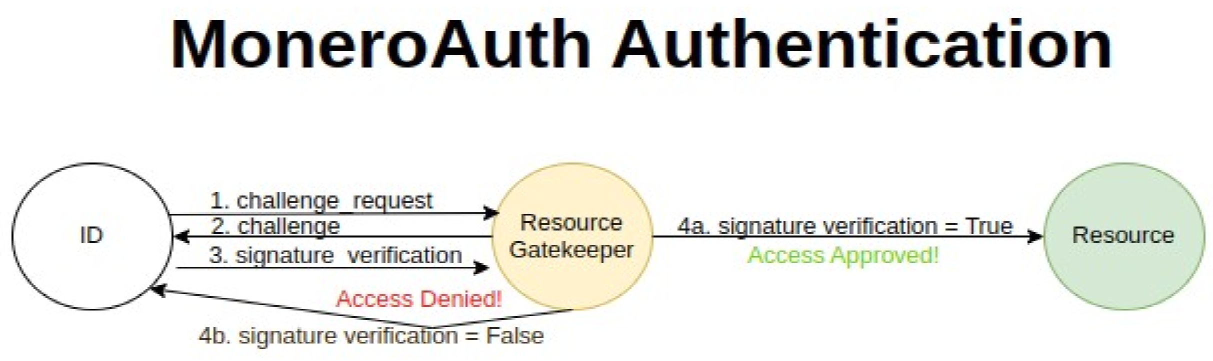
Wikipedia: https://en.wikipedia.org/wiki/Self-sovereign_identity

As stated on Wikipedia:

Self-sovereign identity (SSI) is an approach to digital identity that gives individuals control over the information they use to prove who they are to websites, services, and applications across the web. Without SSI, individuals with persistent accounts (identities) across the internet must rely on a number of large identity providers, such as Facebook (Facebook Connect) and Google (Google Sign-In), that have control of the information associated with their identity.[2] If a user chooses not to use a large identity provider, then they have to create new accounts with each service provider, which fragments their web experiences. Self-sovereign identity offers a way to avoid these two undesirable alternatives. In a self-sovereign identity system, the user accesses services in a streamlined and secure manner, while maintaining control over the information associated with their identity.

MoneroAuth provides SSI for humans and machines. A digital ID is actually a Monero wallet address that is a assigned to a human being or a machine for identity purposes. Using the Monero protocol, an ID (Monero address) is available when a Monero wallet is created. A 25-word mnemonic is provided and can be used to recover the wallet (and the associated ID) if the wallet is ever lost. The 25-word mnemonic should be stored somewhere safely for backup purposes.

MoneroAuth Authentication Overview


MoneroAuth Authentication
ID
1. challenge_request
2. challenge
3. signature_verification
Resource Gatekeeper
4a. signature verification = True
Access Approved!
Resource
Access Denied!
4b. signature verification = False

In the MoneroAuth protocol, the basic authentication process is as follows:

1. A user (human or machine) requests access to a digital or physical resource.
2. The resource gatekeeper, presents a challenge (usually a random string) to the entity requesting access.
3. The entity requesting access, digitally signs the provided challenge and submits the challenge, digital identity, and digital signature to the gatekeeper for verification.

If the digital signature is verified, the entity is allowed access. If the digital signature is not verified, the entity is denied access. **In order to provide a valid digital signature, the user must be in possession of the IDs *private key*. Thus proving ownership of the digital ID.**

The Monero protocol provides ID, signature and signature verification mechanisms. Resource gatekeepers control their algorithms for challenge phrases and they control their own resources to perform signature verifications as well as authorization policy.

To deliver an easy-to-use user experience, the MoneroAuth protocol has a standard set of messages that use json message formatting. An example *challenge* message is shown below:

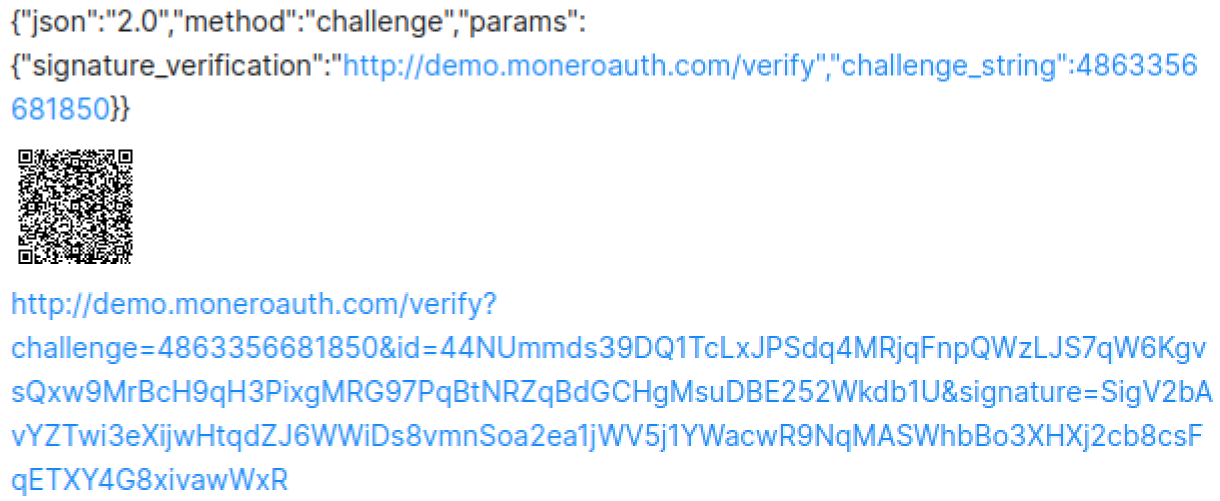
{"json":"2.0","method":"challenge","params":{"signature_verification":"http://demo.moneroauth.com/verify","challenge_string":4863356681850}}

The challenge message has two parameters:

- signature_verification: Typically a URL to verify signatures.
- challenge_string: String to be signed by the digital ID.

authbot – Self-Soverign Identity Assistant

When a user requesting access to a digital resource receives a *challenge* message, the user shares the message with his or her *authbot*. The authbot takes care of signing the challenge, and properly formatting the signature verification URL. The authbot then provides the signature verification URL in QR code and hyperlink formats as shown below:



{"json":"2.0","method":"challenge","params":{"signature_verification":"http://demo.moneroauth.com/verify","challenge_string":4863356681850}}

http://demo.moneroauth.com/verify?challenge=4863356681850&id=44NUmmds39DQ1TcLxJPSdq4MRjqFnpQWzLJS7qW6KgvsQxw9MrBcH9qH3PixgMRG97PqBtNRZqBdGCHgMsuDBE252Wkdb1U&signature=SigV2bAvYZTwi3eXijwHtqdZJ6WWiDs8vmnSoa2ea1jWV5j1YWacwR9NqMASWhbBo3XHXj2cb8csFqETXY4G8xivawWxR

The user simply clicks on the signature verification URL to gain access to the resource. The user could also present the QR code to a camera for signature verification. Once the signature has been verified, the ID has been authenticated. The resource gatekeeper/manager will then check the ID authorization and follow defined policy.

While the authbot can be used for authentication to resources, it can also be used for general purpose digital signature services, signing as well as verifying signatures. Digitally signing public and private documents can be accomplished using the authbot.

The authbot is a computer program that the user can run on his or her computer or in the cloud. We have authbots running on Raspberry Pi and comparable single board computer systems. The authbot is configured to send and receive messages in a Matrix protocol private room using end-to-end encryption. So messages between the user and his or her authbot are private and secure.

The authbot is designed to be used on computers (desktop/laptop) as well as mobile devices (smart phones and tablets). Here is a short video demonstrating the use of the authbot to authenticate to a web site.

Human-to-Machine Interface

In the same way that users can authenticate to web sites, users can also use their digital ID to authenticate to access physical resources.

One use case we frequently mention is the *Door Pass* use case. This use case involves authenticating with a digital ID to open a door for access to a physical facility.

Many organizations issue employees “badges” that are scanned to enter a door to a facility. A MoneroAuth digital ID can obsolete such employee badges.

How does it work?

The door has a QR code available for scanning in close proximity to the door. Employees scan the QR code with their mobile phone. The QR code encodes a URL that requests a challenge phrase from the door gatekeeper. When the employee clicks on the hyperlink, a challenge message is presented to the employee on their mobile phone. The employee shares the challenge message with his or her authbot. The authbot digitally signs the challenge and presents the signature verification URL to the employee. The employee clicks the hyperlink to authenticate. The door gatekeeper verifies the signature then checks to see if the ID provided in the verification is authorized to enter. If authorized, the door opens. If not, a “Entry denied.” message (and help instructions) are displayed on the employees mobile phone.

A similar example of the above use case could be used for entry to vacation rentals or venue kiosks. In these types of use cases, a financial transaction is made (payment for vacation rental during a specified time frame, or a ticket purchase for a specific show at a specified date/time). The digital ID that made the payment is an attribute on the payment receipt. A QR code at the door of the vacation rental or venue kiosk is provided to obtain a challenge from the gatekeeper. The user presents the challenge to his or her authbot, and the authbot returns the signature verification URL. When the user clicks the signature verification URL, the gatekeeper verifies the digital signature, checks to make sure a payment made by the verified ID is on record for access at the current date/time. If all checks out the gatekeeper allows entry. If not instructions on how to proceed are provided on the user’s device.

We will provide demonstration code for our **MoneroAuth Resource Manager** in our github repository. The MoneroAuth Resource Manager Service illustrates how the use cases above can be implemented.

The MoneroAuth Resource Manager Service can be implemented as a central service on the Internet or can be implemented locally at a specific geographic location.

Machine-to-Machine Interface

Machines or devices can be assigned MoneroAuth digital IDs and they can authenticate using their digital IDs to gain access to digital or physical resources just like humans. The MoneroAuth protocol provides messages that machines can send to each other for authentication/authorization purposes as well as communicate with each other in a secure manner to perform day-to-day operations.

Machines can be provided with their own authbot for digital ID services and their own Message Dispatch module. In our lab we have a Raspberry Pi 3 Model B with an assigned ID and it's own authbot. We have performed experiments where the Raspberry Pi is used to control other devices, participate in machine workflows, accept payment for work perfomed, and more. We have recently approved running the authbot on the Libre Computer – AML-S905X-CC (Le Potato). This is a cost-effective option for deploying the authbot.

We intend to share this knowledge across the MoneroAuth community.

Credentials

The MoneroAuth protocol can be used for credentialing via the use of digital signatures. Credentials are another form of digital resources and the MoneroAuth protocol can be used to issue credentials and provide secured access to credential resources.

MoneroAuth Open Source Project

We will place all of our code and documentation in the project github repositories with the goal of building a strong open source community for the MoneroAuth protocol. We intend to place a strong focus on human-to-machine and machine-to-machine interfaces to integrate the MoneroAuth protocol into the IoT space. Expect to see many new repositories under the main MoneroAuth project for details.

We plan to form a non-profit foundation with a vision to promote, manage and extend the MoneroAuth protocol to enable and support Self-Sovereign Identity moving forward.

MoneroAuth Protocol Roadmap

Our goal is for the MoneroAuth protocol to reach the mass adoption phase. To do this, we need resource managers to adopt the MoneroAuth protocol.

Individuals will not use MoneroAuth to securely authenticate to resources if there aren't any resources that support MoneroAuth. Therefore, a primary goal to reach mass adoption is to encourage resource managers to adopt MoneroAuth for authentication and authorization to their resources.

Websites are our first target. We know that the MoneroAuth protocol is easier to implement than traditional authorization schemes, is permissionless and more secure. MoneroAuth can help bring Self-Sovereign Identity to mass adoption. We will distribute free, open source packages that web site administrators can use to easily add MoneroAuth authentication to new or existing web sites.

We also intend to provide assistance to help web site administrators make the transition. We have a web enabled authentication service that can be used to attain MoneroAuth authentication via an intermediary service. We also have a REST API call to verify digital signature verification. Our thinking is that such free service may ease migration and encourage adoption.

The authbot code is now available as a free, open source package to be used as an individual's self-sovereign identity assistant.

Human-to-machine interface use cases will follow. The MoneroAuth protocol can be used to securely control remote devices and can be integrated into digital payment workflows. That opens up opportunities for smart home technology, digital locks, remote sensors/actuators, parking garage services, venue kiosks, even automating the ride-sharing industry once self-driving cars reach mass adoption. We think the opportunities are endless.

Demonstrations and educational content will be available at moneroauth.org.

In 2023 we will publish a MoneroAuth Protocol Roadmap. The roadmap will thereafter be maintained as we move forward.