Middle East Technical University

Informatics Institute

A Survey on Password-free Authentication Method: Passkey

Advisor Name:

Assoc. Prof. Dr. Cihangir TEZCAN

(METU)

Student Name:

Sait Göktuğ DOĞAN

Department of Cyber Security

January 2025

TECHNICAL REPORT

METU/II-TR-2025-

Orta Doğu Teknik Üniversitesi

Enformatik Enstitüsü

Parolasız Kimlik Doğrulama Metodu: Passkey

Danışman Adı:

Doç. Dr. Cihangir TEZCAN

(ODTÜ)

Öğrenci Adı:

Sait Göktuğ DOĞAN

Siber Güvenlik Bölümü

Ocak 2025

TEKNİK RAPOR

ODTÜ/II-TR-2025-

REPORT DOCUMENTATION PAGE

1. AGENCY USE ONLY (Internal Use)	2. REPORT DATE 15.01.2025

3. TITLE AND SUBTITLE

A Survey on Password-free Authentication Method: Passkey

4. AUTHOR (S)	5. REPORT NUMBER (Internal Use)
Sait Göktuğ DOĞAN	**METU/II-TR-2025-**

6. SPONSORING/ MONITORING AGENCY NAME(S) AND SIGNATURE(S)

Cyber Security Master's Programme, Department of Cyber Security, Informatics Institute, METU

Advisor: Assoc. Prof Dr. Cihangir TEZCAN Signature:

7. SUPPLEMENTARY NOTES

8. ABSTRACT (MAXIMUM 200 WORDS)

When a need for to authenticate someone or something, the first answer that comes to mind is to use passwords. While implemented and used properly, passwords are more than enough to securely authenticate. But like everything in the world, passwords have many flaws. It has problems such as guessable passwords, data breaches leaking user passwords, and reuse of the same passwords in different places. To increase security of password authentication, Multi-factor Authentication (MFA) solutions are used together with passwords. But now, a new approach for authentication has arrived, passkeys, claimed to provide better authentication experience in terms of both security and usability. Passkeys are governed by the FIDO Alliance and have a standard called FIDO2. It uses public key cryptography to achieve password-free authentication. Since no passwords are used, passkeys are resistant to threats such as offline cracking, replay attacks, phishing attacks, and reuse of passwords. The study aims to introduce passkeys with explanation of weaknesses in currently used text-based password authentication systems and how technically passkeys can fix these issues and achieve secure password-free authentication. The current state of adoption of passkeys will also be examined. In conclusion, there will be discussion on the advantages and disadvantages of passkeys.

9. SUBJECT TERMS	10. NUMBER OF PAGES **24**

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. Introduction

The term “the password problem” describes security and usability problems associated with text-based passwords [1]. Today, authentication systems heavily rely on text-based passwords. Users have multiple accounts in multiple systems that require them to manage multiple IDs and multiple passwords. To prevent attack techniques such as brute force and dictionary attacks, passwords must be long enough and have high entropy. Systems that store passwords of the users are expected to keep passwords securely, via techniques such as password-hashing algorithms and key derivation functions. On top of these precautions, Multi-factor Authentication (MFA) is employed with the usage of Time-based One-time Passwords (TOTP) and SMS. MFA provides protection against password leaks and common password usage, but there are attacks in literature bypassing MFA security such as MFA fatigue attack[1], SMS phishing attack [2] and SIM swapping attack [3]. Generally, Two-factor Authentication (2FA) is employed, indicating two different factors such as passwords and TOTP are used together to authenticate users. Finally, there are password managers that generate, and store randomly generated passwords for users, instead of memorizing low entropy passwords. Google Password Manager, 1Password, Bitwarden and KeePassXC are examples of such password managers. They store passwords securely, usually in their own servers, and users can access their passwords by memorizing a single master password. This greatly increases high entropy and unique password requirements, but the users need to put their trust into these password managers. Password managers can be susceptible to attacks. For example, widely used Lastpass password manager leaked its password vaults in 2022[2], leading to many users losing access to their accounts and financial losses.

Compared to other authentication methods, such as certificate-based authentication and biometric authentication, text-based password authentication is the most dominant method for authentication. Research conducted in 2016 showed that average password length of users is 9.46 characters, and most passwords have simple structure, such as using only numbers or having meaningful data in them [4]. Password security fails when passwords are easy to guess. By looking

[1] https://www.beyondtrust.com/resources/glossary/mfa-fatigue-attack
[2] https://www.forbes.com/sites/daveywinder/2022/12/23/lastpass-password-vaults-stolen-by-hackers-change-your-master-password-now/

at the results of the research, many users are at risk of having their accounts being stolen because of weak password usage. There are security measures aiming to prevent usage of weak passwords by enforcing policies such as special character usage, a mix of lowercase-uppercase character usage, and rejecting usage of common words. However, weak passwords are just one of the threats that text-based passwords face. Another issue is how passwords are stored. Failing to use cryptographically secure password storing techniques poses a serious security risk. For example, when passwords are leaked from databases, since many users reuse their passwords in other platforms, attackers can gain access to multiple accounts. Text-based passwords are responsible for over 80% of data breaches, and 51% of passwords are reused[3]. Users are generally expected to create passwords that are at least 10 characters long, and each password must be unique for different accounts. These expectations reduce the usability of passwords, making them difficult to use for the users.

While designing a secure authentication system, both security and usability should be considered. Overly complex or cumbersome designs leads users to try bypassing security features, inadvertently creating vulnerabilities. Achieving both strong security and seamless user experience can be called "usable security". By prioritizing user-friendly interfaces and minimizing cumbersome security features, usable security ensures the balance between security and usability. This balance is critical to encourage usage of strong authentication methods, maintaining trust, and preventing errors that could compromise security.

There are many attack vectors for passwords that require specialized knowledge to prevent them. NIST provided many security policies for text-based password authentication systems in their 800-63B publication [5]. While following the prevention methods in this document enhances security, many companies do not adhere to the recommended guidelines. For example, requiring longer passwords may reduce comfort of users and makes passwords harder to remember. This may even lead users to include guessable words in their passwords for easier memorization. Unusable password policies also lead to loss of productivity of users in business environments, as shown in the study conducted in 2010 [6]. And lastly, recommendations for secure password

[3] https://fidoalliance.org/wp-content/uploads/2023/06/June-26-FIDO-EDWG-Spring-2023_Paper-1_Introduction-FINAL.docx.pdf

storage techniques require technical expertise for companies implementing authentication methods in their systems. When combined, these factors for text-based password authentication have too much requirement on both companies and users, making them difficult to secure.

The FIDO Alliance introduced a new authentication method, passkeys, which are claimed superior to passwords[4]. Passkeys offer both enhanced security and improved usability compared to passwords. They use public key cryptography to authenticate the user. Passkeys eliminate the need for two-factor authentication, as the secret required for authentication is stored on the user's device, which is itself used for authentication. This speeds up the authentication process and improves usability compared to passwords. Research by the FIDO Alliance shows that passkey awareness increased by 57% as of 2024[5]. The study also indicates that online phishing attacks have increased this year. Passkeys are resistant to phishing attacks, making them an appealing choice.

Passkey compatibility already integrated into the devices of big providers, such as Apple, Google, Microsoft[6]. Many Android, iOS and Windows devices support passkeys. Users can add a passkey as their authentication method for their Google and Microsoft accounts, for example. Currently, there is a transition period for shifting from passwords to passkeys as the primary authentication method, but the passkey specification needs more work to accomplish this goal. Major platforms have already added support for passkeys, but older devices and operating systems do not fully support passkeys.

In summary, text-based password authentication, while dominant, faces significant challenges in balancing security and usability. Weak passwords, poor password storage practices, and being weak to attacks such as phishing and brute force make passwords a major issue for the design of secure authentication systems. Measures such as enforcing strong password policies, using password managers, and implementing multi-factor authentication can address some issues but often at the cost of usability, leading users to adopt insecure practices. Passkeys, introduced by the FIDO Alliance, offer a promising alternative by combining enhanced security and seamless user

[4] https://fidoalliance.org/passkeys/
[5] https://fidoalliance.org/wp-content/uploads/2024/10/Barometer-Report-Oct-31-2024-2.pdf
[6] https://passkeys.dev/device-support/

experience through strong cryptography. Resistant to phishing and brute force attacks, passkeys are gaining attention with major platforms integrating support of passkeys into their platforms. However, the transition to passkeys as the primary authentication method remains ongoing, with some issues like compatibility across older devices and ecosystems needing further integration to ensure widespread adoption.

2. Passkeys

Many flaws of text-based passwords come from how passwords are transmitted and how they are stored. Essentially, a password is shared secret between user and server that is used to authenticate the user. The security of a password comes from the assumption of it being only known by the user. Other parties only know a mutated version of this knowledge, achieved by using methods such as password hashing functions. But the user does not have any guarantee of how their password is stored in other parties. If the user's password is stored as plaintext by another party, an attacker who gains access to the password database can leak the user's password. Moreover, if the user uses the same password for multiple accounts, those accounts are also at risk of being compromised. This type of attack is called a Password Spraying Attack. Another type of attack that targets passwords is phishing attacks. The attack typically proceeds as follows:

1. The attacker sends a message to the victim containing a link that looks like a legitimate website. For example, suppose that a victim has an account with a bank whose website is www.specialbank.com. The attacker sends a message to the victim, claiming they have earned a special award. To claim it, the victim must visit the website provided in the message, www.spec1albank.com.
2. The victim clicks on the link and sees a login page identical to the bank's original website.
3. The victim enters their username and password information and clicks to the login button.
4. The attacker receives the login information and now can access the victim's account.

This attack is called a Phishing Attack. Even though many websites have prevention for such attacks, such as multi-factor authentication and unusual login attempt lockups, phishing attacks remain effective to this day.

A new type of authentication method, passkeys, has been introduced to address the issues associated with passwords. Passkeys are fundamentally different from passwords. Instead of relying on a shared secret between user and server, passkeys use public key cryptography to generate a public-private key pair. The private key stays with the user, while the public key is shared with the server that the user is trying to authenticate with. The server stores the public key and uses it to verify authentication request sent by the user. The underlying protocol behind passkeys is called FIDO2 protocol. FIDO2 is built upon open standards and consists of two protocols:

WebAuthn [7] and CTAP2 (Client to Authenticator Protocol). Passkey authentication involves two flows: the Registration flow and the Authentication flow. In each of these flows, there are 3 parties involved: the Relying Party (RP), the Client, and the Authenticator.

In the context of FIDO2, an authenticator represents a device such as a user's smartphone or special USB security device. The authenticator creates credentials, generates a public-private key pair, and responds to authentication challenges. An authenticator could be a roaming authenticator, a dedicated hardware subsystem integrated into the client device, or a software component of the client or client device. A client is typically a web application running on a user's browser or a mobile application running on user's smartphone. Its primary role is to act as a bridge between the relying party and the authenticator, using WebAuthn API. Finally, the relying party represents a server where the user can create an account and later sign in. WebAuthn API handles communication between the RP, client, and authenticator. The API is responsible for the management of registration and authentication flows. Meanwhile, CTAP2 handles communication between a roaming authenticator and client, which will be explained in section 2.5.

Whenever a passkey is generated, it is expected to never leave the device they were created inside, dictated by WebAuthn specification. A user must use either CTAP2 to use their passkeys in different devices or set different passkeys in all of their devices to use passkey authentication for their accounts. This property of passkey specification limits usability of passkeys. Moreover, if the user loses access to their devices, their passkeys become unusable. Account recovery procedures can be used for transferring passkeys from one device to another, but currently there is no formal procedure for such a transfer operation. To solve this problem, FIDO Alliance is working on a new protocol, Credentials Exchange Protocol (CXP), to enable transfer of passkeys from one device to another. If both devices support CXP, they will be able to transfer their passkeys between them, with the consent of the user. Details of Credential Exchange Protocol will be discussed in section 2.6.

2.1 Authentication with Passkeys

Websites that are supporting passkeys put an option for creating account with passkeys in their user registration pages. Since there is an ongoing process of adoption of passkeys, FIDO recommends websites to not make passkey registration mandatory. If users do not want to use passkey, they can set password for their accounts. At the first step, users decide on their username while registering to website as in Figure 1. Then, users see a pop-up screen. The website asks whether to create their account using a passkey or skip passkey generation and continue with creating password. Figure 2 represents this pop-up screen. If the user clicks “Create a passkey” option button, another pop-up is now shown in screen. If the user is using a Windows machine, they see Windows operating system’s Windows Hello pop-up screen, asking their PIN code as a verification method, like in Figure 3. If the user enters their Windows machine’s PIN code, FIDO registration flow starts between authenticator, browser and the website. The user is not interrupted during the flow, meaning once the flow is completed, account creation of the user is completed as well. In Figure 4, created account’s detail for the user on the website is shown. After account creation is successfully created, the website knows public key part of user’s passkey. The user keeps private key part of their passkey, which is required for any future authentication operations. Without this private key, nobody can access the account created.

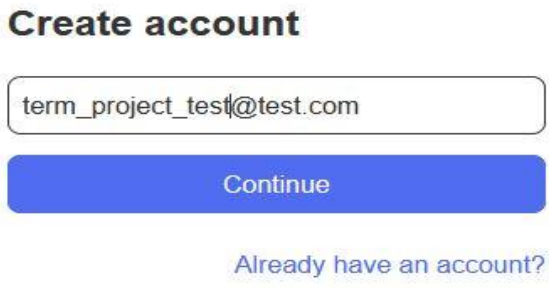

Figure 1. The user enters their username while registration to the relying party.

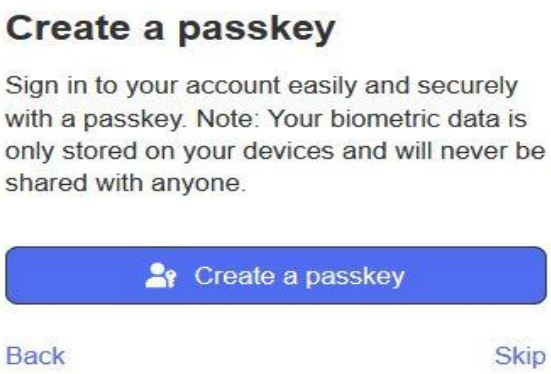

Figure 2. The user sees "Create a passkey" option while creating their account.

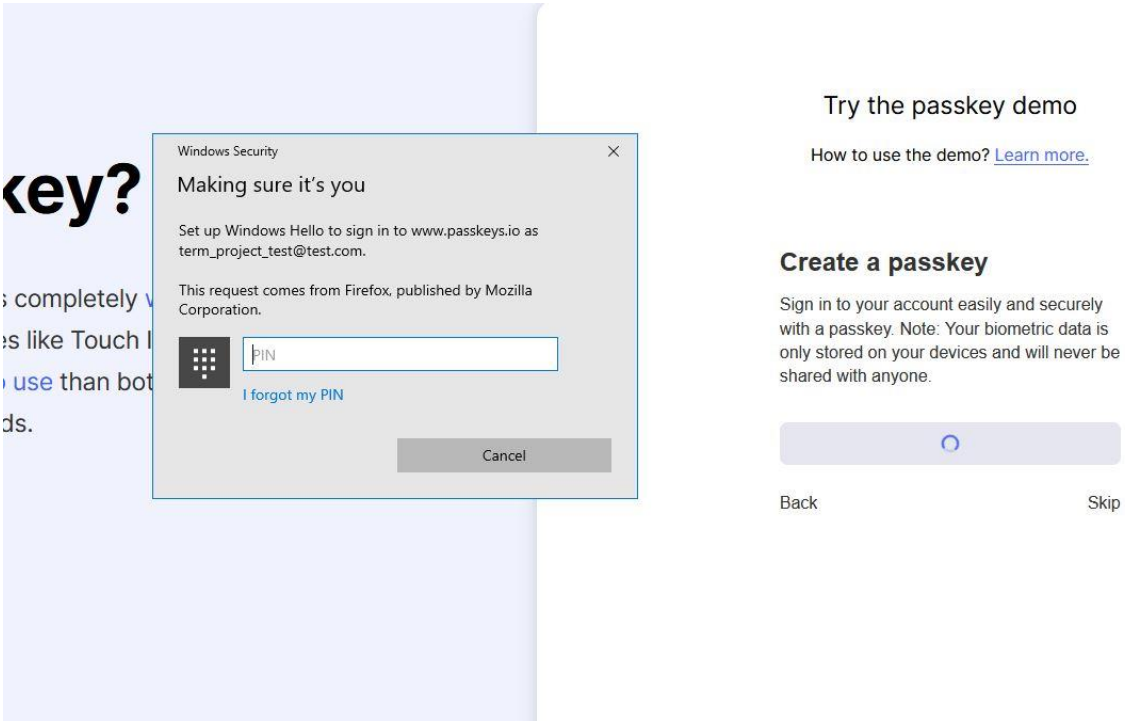

Figure 3. Windows Hello PIN approval pop-up is displayed for passkey creation.

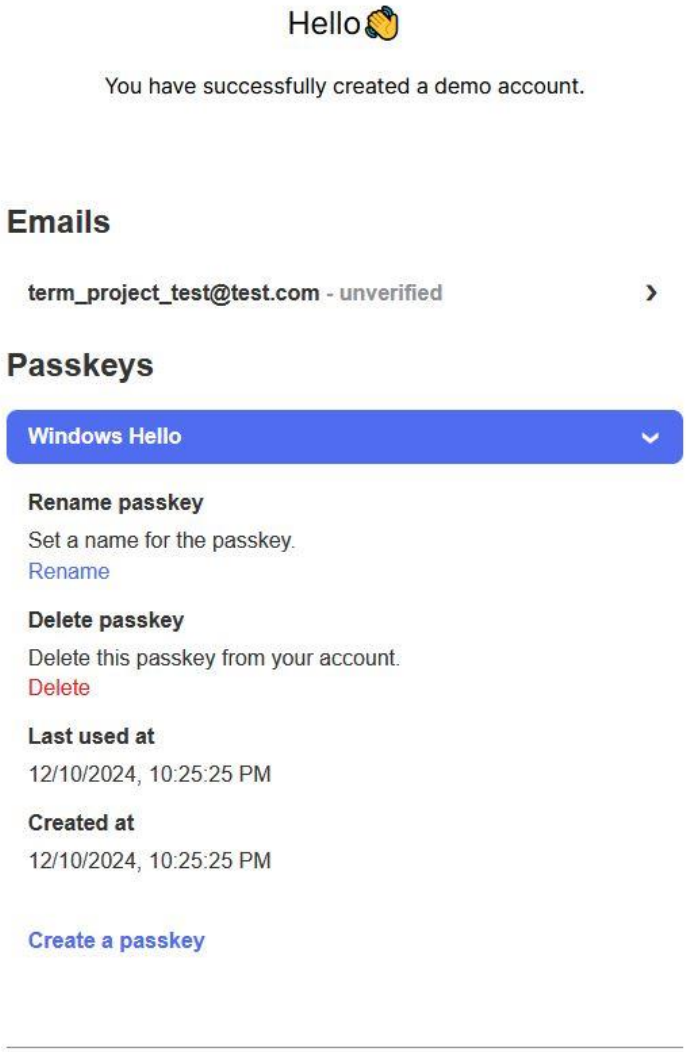

Figure 4. Information of the created account after registration.

After account creation, the user can access their account with sign in feature of the website. With passkey, the user clicks “Sign in with a passkey” option button as shown in Figure 5. After the user clicks this button, a pop-up screen from Windows Hello asks for PIN code for verifying login attempt of the user like in Figure 5. Once PIN code is verified successfully, FIDO authentication flow starts between authenticator, browser and the website. If the user has only one passkey setup for the website, the user is not interrupted with any confirmation screens during the

flow. Windows Hello automatically selects existing passkey in this scenario for the authentication flow. In the end, if the authentication flow is successful, the user can access their account as it is shown in Figure 4.

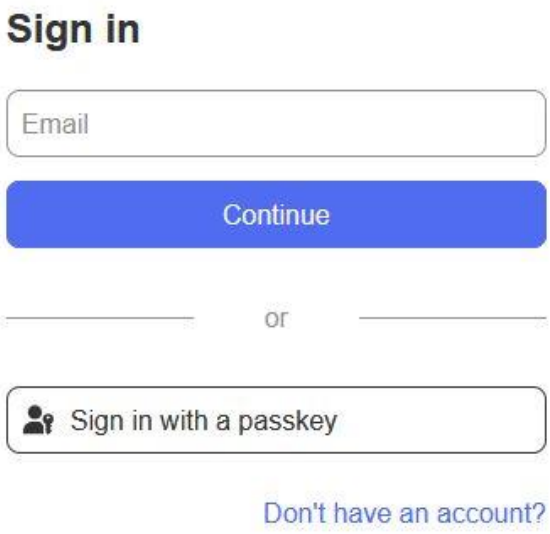

Figure 5. Sign in screen that performs authentication for the user with a passkey.

Authentication is also possible with using multiple devices. If the user creates an account with their smartphone, they can also sign in to their account using their computer. Client to Authenticator Protocol 2 (CTAP2) enables this type of authentication. Technical details of CTAP2 and an example of authentication with CTAP2 will be discussed later.

2.2 Registration Flow

The purpose of registration flow is to create authentication information about the user to be authenticated to service in future. The flow starts with user initiating registration. Client initiates a request to relying party to register authenticator for the user. The relying party responds to client with a couple of information, called *PublicKeyCredentialCreationOptions*[7] in WebAuthn specification. This object contains information such as user, relying party and options for credential creation. A challenge is also provided in this object to prevent replay attacks. Next, the client will

[7] https://www.w3.org/TR/webauthn/#dictdef-publickeycredentialcreationoptions

call the authenticator to initiate the credential creation process. Authenticator asks user consent to continue the process. Consent usually requires PIN codes, fingerprints, face scans etc. After verifying consent, the authenticator creates a new asymmetric key pair, consisting of a public and private key. The private key is safely stored inside, and the public key becomes a part of attestation, which the authenticator signs over with the private key. After the authenticator is ready, it sends this information to client, which will send them to the relying party. Finally, the relying party validates incoming information and accepts it to finish registration flow. Sequence diagram of the registration flow is shown in Figure 6.

During registration, a relying party may want information about authenticators for many purposes. Authenticity of authenticator is an important aspect for user authentication. Webauthn provides this support using an attestation mechanism. By using the attestation, the relying party can verify authenticator and its properties. This provides a means for the relying party to enforce security policies for user authentication process. In FIDO authentication, attestation statements can be exchanged during user registration.

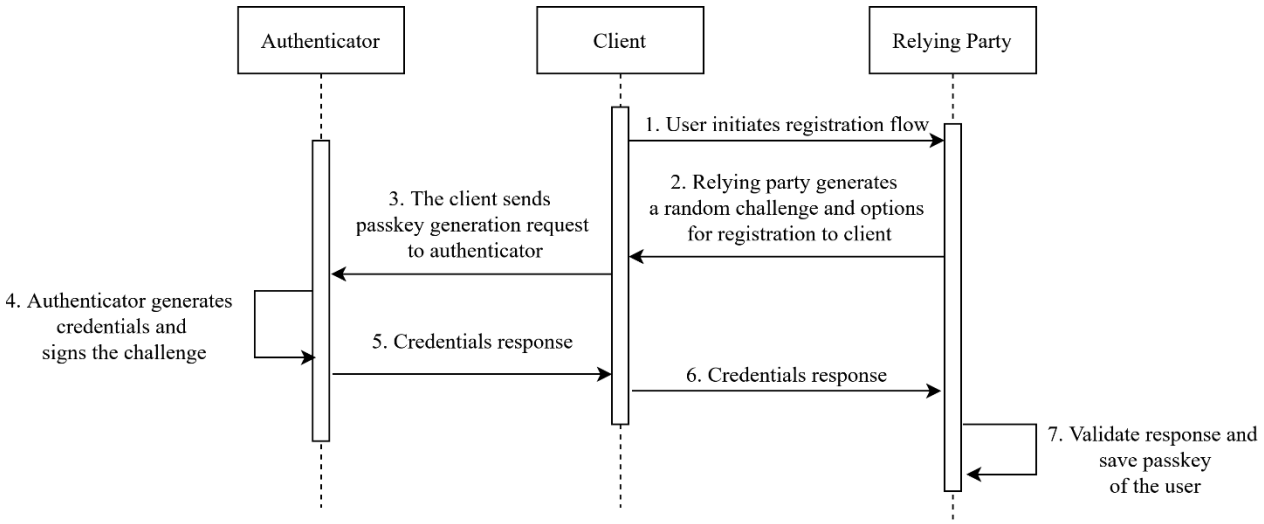

Figure 6. Sequence diagram of the registration flow.

2.3 Attestation

Attestation is the process of verifying authenticity of the user's authenticator during FIDO registration flow. During registration, authenticator creates user's authentication passkey, not depending on whether attestation is involved or not. At this stage, the authenticator generates a unique asymmetric key pair. One of the pair, private key stays inside authenticator, stored securely, while public key is shared with the relying party. After passkey generation, the authenticator may provide an attestation to the relying party for assurance about the integrity of authenticator.

If attestation is requested by relying party, authenticator provides an attestation by using its attestation key to sign the AAGUID (Authenticator Attestation Globally Unique ID) along with previously created passkey public key. In the case of authenticator cannot provide an attestation, the relying party can authenticate user with its passkey, also may obtain additional information about authenticator such as AAGUID, but the relying party may not have evidence about whether required authenticator properties are present or not.

Usage of attestation has many use cases for both relying party and end user. For relying parties, attestation can provide assured authenticator security and compliance, verification with attestation signature and incident handling. For end users, attestation increases trust in services, compliance of authenticators, transparent registration and on-boarding. There are 4 different types of attestation. These are:

1. **Self Attestation:** In this type, attestation statement is signed by using user's passkey. This kind of attestation provides integrity protection for the attestation statement but provides no further assurances for the relying party.
2. **Basic Attestation:** For this type of attestation, attestation statement is signed by a key created by manufacturer of the authenticator and put into authenticator. An important note is that the key used for attestation is not unique, because of the manufacturer of authenticator using same key for same authenticator model. To address privacy issues, FIDO Alliance recommends using the same key for at least 100.000 devices.

3. **Attestation CA or Anonymization CA:** Like basic attestation, except in this case the attestation statement is signed by a TPM (Trusted Platform Module) Attestation Key. The TPM is a hardware module where cryptographic operations can be securely processed, and storage of secrets can be achieved that secrets never leave the module. TPM stores attestation key, and this key's certificate signed by a trusted authority managing the authenticator.
4. **Enterprise Attestation:** This type of attestation enables a relying party to receive further information about authenticators. Generally, this attestation type is used by enterprises to have better handling of threat incidents or forensic investigations.

The AAGUID (Authenticator Attestation GUID) is used for unique identification of the authenticator's manufacturer and its model. An important property of AAGUID is it does not uniquely identify a specific authenticator. When relying party requests attestation, the authenticator returns its AAGUID. The relying party may use AAGUID to identify authenticator's manufacturer and its model. If the relying party has policies such as denying a specific authenticator for registration of users, the relying party can use AAGUID of authenticator. In fact, AAGUID is look-up value of FIDO's MDS (Metadata Service). MDS contains information about FIDO approved authenticators, and its publicly available for relying parties. It should also be noted that AAGUID does not provide integrity or authenticity. Relying parties are expected to trust provided value from authenticators.

One of the fundamentals of attestation is the relying party's trust in the authenticator's attestation key. The key must be created by a trusted source and protected by the authenticator. In this case, trusted source is manufacturer of the authenticator. However, in Attestation CA or Anonymization CA type of attestation, the relying party can assert authenticity of the authenticator with a trusted source or certification authority. In this scenario, the relying party can fetch the public key part of authenticator's attestation key using FIDO MDS.

Attestation provides information about the type of authenticator to the relying party. Also, it provides the following benefits:

- If an attacker intercepts a registration message, the attacker would not be able to swap out public key of user, since attestation signature would not match.
- Allows relying parties to know which or what type of authenticator is being used for registration.

Some stricter relying parties (financial industry, healthcare industry etc.) may want to know more about the device that is being used to access their systems. If the relying party wants stricter controls about authenticators, they can use enterprise attestation to retrieve more information about the authenticator that is used. Enterprise attestation allows relying parties to extend capabilities of attestation, retrieving more information about the authenticator.

User registration to relying party starts with user initiating registration flow. The relying party fetches information about the authenticator that is being used for registration. After identifying the authenticator, the relying party fetches the authenticator's AK (Attestation Key) certificate from FIDO MDS. After, the relying party sends passkey creation request to the authenticator. The authenticator generates a passkey, putting public part of the passkey into attestation object. Then, the authenticator signs the attestation object with its own attestation key and sends the signed attestation object to the relying party. The relying party validates the received attestation object, using AK certificate of the authenticator. After successful validation, the relying party can accept passkey of the user. The registration flow is visualized in Figure 7.

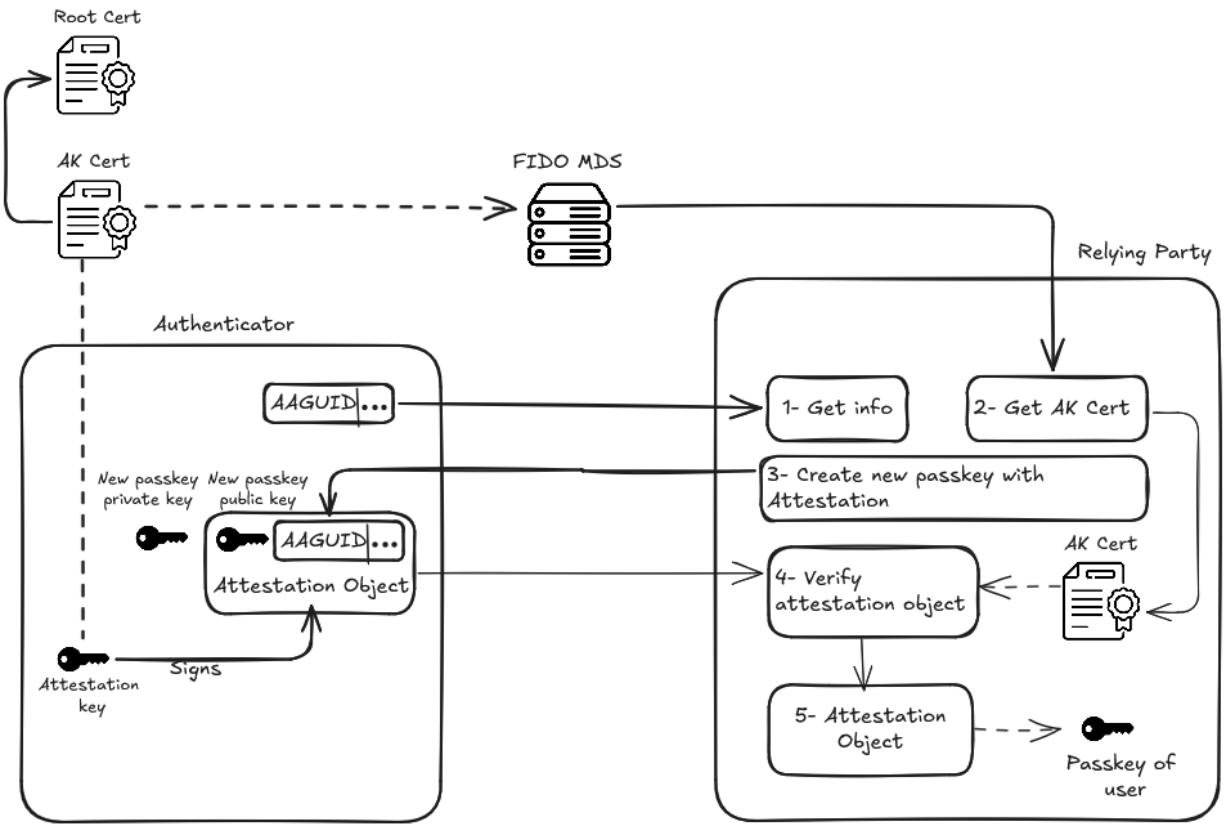

Figure 7. Registration Flow with attestation.

2.4 Authentication Flow

The user starts authentication flow they want to access the account they created during registration to the relying party. The client sends an authentication request to the relying party that starts the authentication flow. From now on, the client acts as middleman between the authenticator and the relying party. Once the relying party receives the request, it responds with a randomly generated challenge that must be processed by the authenticator. The client forwards the challenge to the authenticator. If the relying party wants the user's permission to continue the authentication flow, it puts a flag into the request, indicating the authenticator should ask the user's approval to continue the flow. After the authenticator signs the challenge with private key of the user, which is a part of the passkey, the client sends signed challenge to the relying party. The relying party validates the signed challenge with public key of the user, as well as whether the challenge was

originally created by themselves. After validation successfully completed, the user is authenticated to the relying party. Sequence diagram of the authentication flow is shown in Figure 8.

The relying party can perform additional checks on the authenticator, such as detecting whether the authenticator has been cloned or not. The relying party stores a counter for each successful authentications with a specific passkey. In parallel, the authenticator also stores a counter for each successful authentication with the passkey. During registration, the authenticator sends the counter to the relying party. If the counter of the relying party is less than the counter received by the authenticator, this may indicate more than one copy of the passkey may exist and are being used in parallel. WebAuthn specification suggests incorporating this information into a risk scoring system[8]. But the specification allows a relying party to fail the authentication attempts, if the relying party desires. For high-security scenarios, such as military applications, denying authentication attempt in such cases is useful.

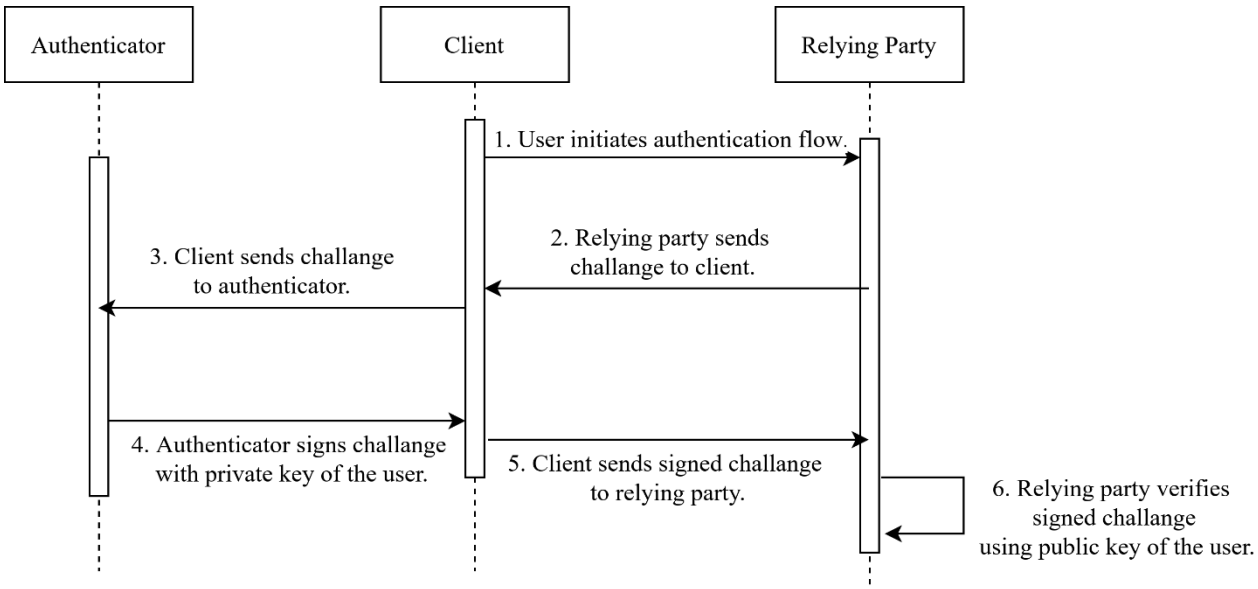

Figure 8. Sequence diagram of authentication flow.

[8] https://www.w3.org/TR/webauthn/#sctn-verifying-assertion

2.5 Client to Authenticator Protocol 2 (CTAP2)

CTAP2 is a FIDO Alliance specification that enables communication between a roaming authenticator and a client during authentication and registration flows [8]. An authenticator that implements CTAP2 is called a FIDO2 Authenticator. CTAP2 is designed to support use cases where a user wants to save or use a passkey on a device different from the device the one the client is currently working on. For example, a user registered with a relying party using an Android smartphone and later wants to authenticate to the relying party from another device, such as a Windows PC, the WebAuthn specification alone does not provide a direct method for communication between the devices. However, if the user's smartphone is a FIDO2 authenticator, the user can authenticate to the relying party with their Windows PC communicating to the smartphone. Figure 9 illustrates the relationship between clients, relying parties and different types of authenticators.

FIDO2 authenticators can communicate with clients using Bluetooth Low Energy (BLE), USB and NFC, depending on the type of authenticator. There are different types of FIDO2 authenticators, such as password managers and hardware security keys. In the case of password managers in smartphones, such as Google Password Manager, CTAP2 uses BLE to perform authentication. If the client supports CTAP2 and Bluetooth, the user can choose their smartphone as authenticator for the authentication. At the start of the authentication flow, the client displays a QR code to the user. The user scans the code in their smartphone and sees stored passkeys that are suitable for the authentication. Once the user selects the passkey, the authenticator in the smartphone sends signed authentication challenge to the relying party through the client. Alternatively, the user can use a plugged-in USB security key or NFC device as an authenticator, if supported by the client and these devices are FIDO2 authenticators.

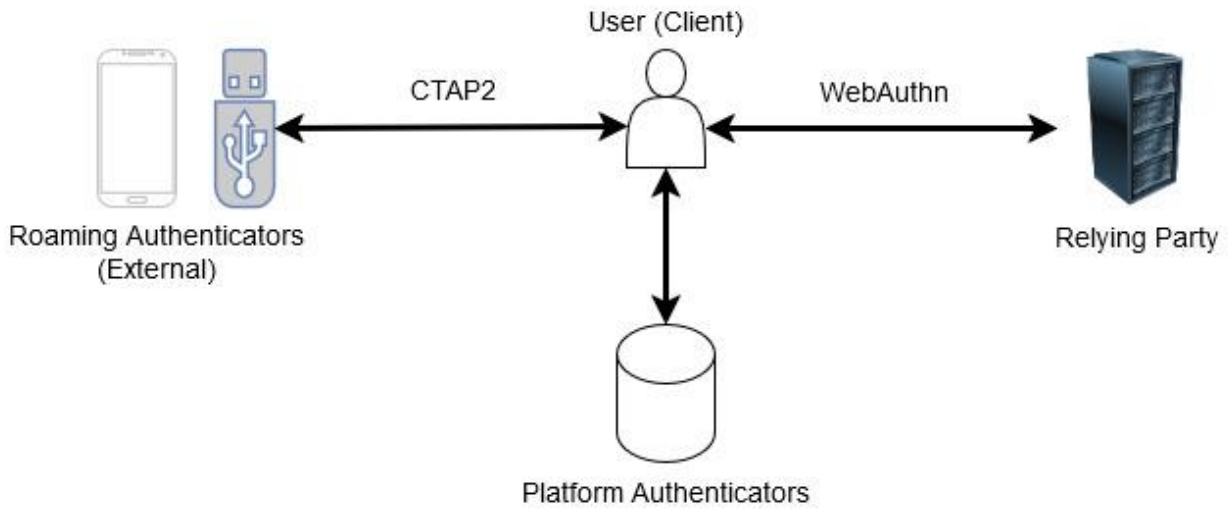

Figure 9. Communication between clients, authenticators and relying parties.

2.6 Credential Exchange Protocol

The WebAuthn specification dictates that a created passkey should never leave the device on which it is created[9]. Instead of transferring passkeys directly when users need to use them on another device, the specification suggests creating new passkeys on the device they want to transfer their passkeys. Many of the password managers, such as Google Password Manager and iCloud Keychain, store passkeys of users in their cloud services. These password managers provide support for transferring passkeys in cases like account recovery and device changes. However, there is currently no standard specification for transferring passkeys between different devices. The FIDO Alliance is working on a specification to support this scenario. The specification is called Credentials Exchange Protocol (CXP) [9]. As of 2025, existing specific password managers can transfer passkeys between different instances of the same password manager. With CXP, it will become possible for different password manager instances to securely transfer passkeys between them, significantly improving usability of passkeys for users. It is important to note that CXP is not yet part of the FIDO2 standard.

[9] https://www.w3.org/TR/webauthn/#sctn-credential-loss-key-mobility

3. Passkey Security and Usability Overview

Passkeys are designed to be an improvement on text-based passwords with MFA protection. By design, they are resistant against guessing, password-reuse, phishing attacks, replay attacks, and server password leaks. A security comparison between each authentication method is shown in Table 1. Each passkey is unique, bound to a single relying party. The part of the passkey that needs to be kept hidden from outside, it never leaves the authenticator. Relying parties only know about public part of a passkey.

Table 1. Security comparison between different authentication methods.

Protects against	Memorized password	Password manager	Password manager+2FA	Passkey
Guessing	×	✓	✓	✓
Credential reuse	×	✓	✓	✓
Device theft	×	!	!	✓
Phishing	×	!	!	✓
Server leaks	×	×	×	✓

×: Does not protect.

!: Limited protection.

✓: Protects.

In terms of security, passkeys are superior to passwords. Each passkey is unique, and they are created by using secure cryptographic signature algorithms. List of supported algorithms are defined in COSE algorithm registry[10]. Since each passkey is generated randomly, they are resistant against brute force attacks. Also, relying parties generate a unique and random challenge for each registration and authentication requests. This challenge prevents an attacker from performing replay attacks. In the case of authenticator cloning, both an authenticator and a relying party keep a counter for each authentication operation with a specific passkey. During authentication flow, the

[10] https://www.iana.org/assignments/cose/cose.xhtml#algorithms

authenticator shares its internal counter with the relying party, and the relying party checks the counter, whether the counter is same with the previously stored counter or not. If they are same, both parties increment their counters and approve authentication request.

Many authenticators rely on the Trusted Platform Module (TPM) or Trusted Execution Environment (TEE) of the device they are working on for FIDO2 implementations. These components are used for both attestation and secure storage of the passkeys. If a passkey is stored on a device without encryption and the relying party fails to verify signature of the authenticator, an attacker can extract private key of the passkey from the disk and use it to gain unauthorized access to the victim's account. When a TPM or TEE is available, authenticators leverage these modules to encrypt passkeys before storing them. However, there can be vulnerabilities in TPM and TEE, posing security risks when storing passkeys. In 2022, [10] demonstrated an attack targeting a TEE running on TrustZone hardware. This attack achieved FIDO2 WebAuthn login bypass and the attack works on some Android smartphones, such as Samsung Galaxy S8 – S9 – S10 – S20 and S21. Thus, secure implementation and design of authenticators are important for security of passkeys.

FIDO Alliance has an authenticator certification program called the FIDO Certification Program that checks design and implementation of authenticators and grants different certification levels to them. From lower security to upper, these levels are L1, L1+, L2, L3 and L3+. During passkey registration, a relying party can check certification level of authenticator through attestation. This feature is useful for certain security needs, such as banking and military. Passkey users are advised to use authenticators that have FIDO2 certification, at least L1.

3.1 Account Recovery

Passkey authentication requires an account recovery procedure for situations where a user loses access to the device storing their passkeys. In password-based authentication, if a user forgets their password, a commonly used method for account recovery is to send an email to the user's email account that contains a magic link to set new password for their account. While this method

can be applicable for passkey authentication, it introduces a significant weakness, the user's email account becomes the weakest link in the security chain. The WebAuthn specification recommends creating multiple passkeys for a single account across different devices. If access to one device is lost, users can access their accounts with other devices.

Some password managers, such as Google Password Manager and iCloud Keychain, offer the ability to transfer passkeys between devices for account recovery. However, this approach conflicts with the WebAuthn specification, which mandates that a passkey should never leave the device on which they are created. To solve this inconsistency, the FIDO Alliance is developing a new protocol called Credential Exchange Protocol (CXP). A working draft of this protocol has been published[11]. The protocol is expected to be a part of FIDO2 specification in the future, providing a standardized solution for secure passkey recovery and management.

Yubico, an authenticator manufacturer, has proposed a secure method for creating backup authenticators to simplify the recovery process for passkeys[12]. As part of this proposal, Yubico introduced a new cryptographic primitive called Asynchronous Remote Key Generation (ARKG) that allows a primary authenticator to generate unlinkable public keys for a backup authenticator that can be used for account recovery without compromising security. To ensure post-quantum security of the ARKG, this account recovery method has been further developed and refined in subsequent research, as detailed in [11].

3.2 Passkey Usability

When designing a secure authentication system, both security and usability must be considered. Finding the right balance between security and usability is challenging, as focusing too heavily on security often leads to reduced usability. And if the system becomes too difficult to use, users tend to perform unsecure practices. For example, during account registration, if the system asks users to provide a password where it has 14 characters, at least one uppercase character and a

[11] https://fidoalliance.org/specs/cx/cxf-v1.0-wd-20241003.html
[12] https://www.yubico.com/blog/yubico-proposes-webauthn-protocol-extension-to-simplify-backup-security-keys/

special character, users provide a password to the system that contains special information about them. An attacker familiar with the user could exploit this information by generating a password wordlist using tools like cewl[13] and perform dictionary attack. To prevent such unsecure practices, the design of the system should provide a balance between security and usability, which can be called “usable security”.

One of the primary design goals of passkeys is to provide users with “usable security”, ensuring both strong security and a seamless user experience. To achieve this, the FIDO Alliance has established several design objectives for passkeys to increase user satisfaction while using passkeys. These goals include:

- Decreasing login time.
- Increasing success rate of first try login attempt.
- Increasing user satisfaction while using passkeys.

To support these objectives, the FIDO Alliance has published a passkey design guideline to help companies create better user experiences when implementing passkeys into their systems[14]. Although adherence to the guideline is not mandatory, following it can promote a standardized feel of user experience while using passkeys. A standardized user experience will increase adoption of passkeys and help accomplishment of design goals of passkeys.

[13] https://github.com/digininja/CeWL
[14] https://www.passkeycentral.org/design-guidelines/

4. Conclusion

The concept of passkeys is highly promising as a future replacement for passwords. Their primary goal is to provide a more secure and user-friendly authentication method that eliminates the need for passwords entirely. By utilizing strong cryptographic algorithms and modern hardware, passkeys can address many issues associated with passwords, such as weak passwords, phishing attacks, and data breaches. As adoption increases and companies integrate passkeys into their authentication systems, passkeys have the potential to set a new standard for secure authentication.

However, several challenges remain that need to be addressed. The device-bound nature of passkeys can become problematic when access to the device is lost, and users need to recover their accounts. There are complicated account recovery methods for passkeys that are not user-friendly. Additionally, the usability of passkeys relies heavily on the interoperability of ecosystems and platforms. While major platforms such as Google and Apple offer built-in authenticators that work seamlessly within their own platforms, cross-platform compatibility remains limited.

Despite these challenges, passkeys are a significant advancement in addressing the issues associated with passwords. With further refinement and solutions to these issues, passkeys have the potential to become the primary method for secure, user-friendly authentication.

.

REFERENCES

[1] S. Wiedenbeck, J. Waters, J.-C. Birget, A. Brodskiy and N. Memon, "PassPoints: Design and longitudinal evaluation of a graphical password system," *International Journal of Human-Computer Studies,* vol. 63, no. 1-2, pp. 102-117, 2005.

[2] M. Jakobsson, "Two-factor inauthentication – the rise in SMS phishing attacks," *Computer Fraud & Security,* vol. 2018, no. 6, pp. 6-8, 2018.

[3] M. Kim, J. Suh and H. Kwon, "A Study of the Emerging Trends in SIM Swapping Crime and Effective Countermeasures," *2022 IEEE/ACIS 7th International Conference on Big Data, Cloud Computing, and Data Science (BCD),* pp. 240-245, 2022.

[4] S. Chao, Y. Tianwen, X. Haodi, Y. Gengshan and G. Xiaohong, "User practice in password security: An empirical study of real-life passwords in the wild," *Computers & Security,* 2016.

[5] NIST, "NIST SP 800-63-4 Digital Identity Guidelines: Authentication and Authenticator Management," 21 08 2024. [Online]. Available: https://csrc.nist.gov/pubs/sp/800/63/b/4/2pd. [Accessed 24 12 2024].

[6] P. Inglesant and M. A. Sasse, "The true cost of unusable password policies: password use in the wild," in *Association for Computing Machinery*, Atlanta, Georgia, USA, 2010.

[7] W3C, "Web Authentication: An API for accessing Public Key Credentials," 08 04 2021. [Online]. Available: https://www.w3.org/TR/webauthn/. [Accessed 30 12 2024].

[8] FIDO Alliance, [Online]. Available: https://fidoalliance.org/specs/fido-v2.0-ps-20190130/fido-client-to-authenticator-protocol-v2.0-ps-20190130.html. [Accessed 30 12 2024].

[9] FIDO Alliance, 03 10 2024. [Online]. Available: https://fidoalliance.org/specs/cx/cxp-v1.0-wd-20241003.html. [Accessed 30 12 2024].

[10] A. Shakevsky, E. Ronen and A. Wool, "Trust Dies in Darkness: Shedding Light on Samsung's TrustZone Keymaster Design," *Cryptology ePrint Archive,* 2022.

[11] J. Brendel, S. Clermont and M. Fischlin, "Post-Quantum Asynchronous Remote Key Generation for FIDO2 Account Recovery," *Cryptology ePrint Archive, Paper 2023/1275,* 2023.