POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea

Transition to Passwordless Technologies

A Comprehensive Analysis and Real-World Implementation

Relatore
prof. Andrea Atzeni

Davide Caria

Anno accademico 2024-2025

Alla mia famiglia, Mamma,
Babbo e Federico,
a Zia Maria Antonietta

Summary

This thesis explores the transition to passwordless technologies, a critical subject I developed during my master's studies and further refined during an internship at Microsoft. While passwordless authentication is a growing area of interest, most studies focus on the individual technologies behind it, such as security keys, biometrics, and magic links. However, little attention has been paid to the integration of these technologies within existing architectures, which is essential for a successful transition from traditional password-based systems.

To bridge this gap, the thesis presents a framework for effectively planning and executing a shift to passwordless authentication. It begins by defining the core services required to integrate various passwordless methods, followed by an approach to assess and design target architectures. The work then outlines a strategic plan that organizations can follow to facilitate this transition while understanding the increase in the security posture. Additionally, costs and risks associated with the shift in the authentication paradigm are described.

A case study involving a large enterprise that, for privacy reasons, will be referred to as "GripGotham", illustrates the application of this framework in a real-world setting. The study demonstrates the practical complexities of implementing passwordless technologies at scale and highlights the security improvements achieved through a methodical, well-planned adoption. The insights gained from this case provide valuable guidance for professionals and organizations navigating their own transition to more secure and user-friendly authentication practices.

Acknowledgements

Questa e' solo una tappa di un percorso piu' grande, un percorso che ho costruito grazie a persone straordinarie che l'hanno reso speciale. E' come aver completato il primo capitolo di un videogioco, quel momento in cui scorrono i titoli di coda e si ringraziano gli sviluppatori, gli animatori, i level designer e tutti coloro che hanno reso possibile realizzare quest'opera. Alcuni di voi erano presenti sin dalle fasi iniziali, altri si sono uniti lungo il cammino, e altri ancora parteciperanno ai capitoli successivi. Questo e' il mio ringraziamento per aver contribuito a cio' che sta dentro e fuori da queste pagine.

Alla mia famiglia, a Mamma, Babbo e Federico. Da quando volevo fare lo scienziato a quando sono diventato ingegnere sono passati anni, ma voi siete stati sempre li ad aspettare e celebrare ogni piccolo traguardo. Grazie per avermi dato gli strumenti per affrontare qualunque difficolta', risolvere i dubbi e gestire le insicurezze. Grazie per avermi lasciato libero di prendere le mie scelte e esplorare le strade che mi hanno portato qui. Grazie, perche' siete stati i coautori dei miei risultati e sempre sarate i sostenitori dei miei obbiettivi. E' stupendo vedere la gioia con cui condividiamo questi momenti e cui celebriamo la nostra famiglia. Grazie a Zio Franco, per avermi incantato con i computer e avermi fatto scoprire l'informatica. Hai contribuito a farmi entrare nel mondo in cui lavoro e che mi appassiona come nient'altro.

Ai miei amici di sempre, al gruppo dei limoni. Grazie per essere cresciuti insieme a me e per aver condiviso storie che ci racconteremo per sempre. Dai primi giorni in una scuola che si scioglieva sotto la pioggia, ai festeggiamenti per i nostri traguardi, passando per quei pomeriggi in spiaggia a scavare buche che sembravano monolocali, fino alle serate al Mill Inn trascorse a parlare di politica. Mi emoziona ritrovarci per le occasioni speciali; anche se ora siamo sparsi per l'Europa, ogni volta e' come organizzare un ritrovo di famiglia. Difficile scegliere chi menzionare, avrei tante storie da citare, in questo momento mi viene in mente Elia che passa una serata a fare fronflip con due sottobicchieri di cartone.

Al gruppo delle cenette. Grazie per avermi mostrato quanto il cibo possa unire le persone, trasformando delle semplici cene in un legame d'amicizia. Da un piatto di chili con carne siamo passati a incontrarci ogni weekend, esplorando insieme questa magnifica citta' che ormai tutti chiamiamo casa. Una mezione speciale va a Tamer e Carlo, una delle coppie meno probabili di sempre. La "s" nei vostri nomi rapresenta lo "Studio", ma "r" sta sicuramente per "Risate", "Relax" e "Ricerche strane sul tablet di Tamer". Difficile non mezionare anche Giulio e Luca, amici che avro' sempre accanto, ad esempio quando porteremo i soldi della nostra startup a Panama.

Al gruppo dell'Honduras, ai Negritos. Grazie per avermi accompagno durante l'anno che ha cambiato la mia vita. Grazie perche' nonostante le occasioni per vederci siano poche, ogni votla e' come se non avessimo mai preso quel tragetto per tornare da Utila. Come se non fossimo mai tornati e avessimo vissuto insieme gioie, emozioni e traumi; ad esempio un colpo di stato.

A mi familia hondureña. Gracias por recibirme como a un hijo y hacerme sentir querido como un miembro mas de la familia. Pensar en ustedes me recuerda una frase que aprendi' antes de partir: la distancia es como el viento al fuego, apaga los pequeños y alimenta los grandes.

A Sofi. Una persona speciale, un aiuto infinito, e un'emozione che non si puo' descrivere a parole. Grazie per essermi stata accanto nei momenti chiave di questo percorso. Hai fatto molto piu' di quanto tu possa immaginare e, senza il tuo supporto, non sarei riuscito a realizzare questo sogno.

Grazie per non esserti fermata all'apparenza: un ragazzo che si trasferisce in giro per l'Europa dopo solo pochi mesi. Grazie per aver reso la distanza qualcosa di irrilevante e per avermi fatto scoprire che la cura personale va oltre quella del corpo. Ti sono grato per tutto.

A Simo, prima un compagno a lezione, poi un compagno di studi, e ora un compagno di viaggio. Ringrazio il momento in cui hai sceglto di venirmi a fare compagnia nel corso di informatica. Raramente ho passato cosi' tante ore al telefono con qualcuno, e ora posso dire che ogni minuto e' stato ben speso. Che si tratti di risolvere un esercizio di segnali, organizzare una cena o discutere dilemmi esistenziali, so sempre chi chiamare. Mi godo questo momento, mentre immagino cosa finiremo a fare nei prossimi anni.

Ai principali sponsor del mio lavoro: il mio relatore Andrea Atzeni e al mio manager Bart Asnot. Grazie per avermi premesso di realizzare questa tesi, dandomi supporto sia dal lato universitario che aziendale. Grazie per i feedback e i commenti che hanno dato valore e hanno permesso la stesura di queste pagine, se sono soddisfatto del mio lavoro e' anche grazie a voi.

Ad Antonio. Non devo sforzarmi molto per ricordare chi ha fatto nascere in me la passione e la curiosita' verso le cose. Che fosse un generatore, le navi con cui colombo e' salpato per le americhe, o un albero di natale con bichieri riciclati, hai sempre saputo catturare la mia attenzione e incantarmi con le tue spiegazioni. Cercando nel vocabolario la parola "educatore" vorrei trovare la tua foto!

A Zia Maria Antonietta, grazie per aver alimentato e coltivato il mio entusiasmo per la conoscenza. Mi hai trasmesso tutta la tua passione per le materie che amavi e mi hai dato un metodo per interpretarle. Sono grato delle ore che hai speso a risolvere con me le espressioni; esercizi che rifacevo continuamente, mentre tu pazientemente mi spigavi in che ordine dovessi risolverle. Grazie per avermi sempre mostrato la tua fierezza e entusiasmo verso quello che stavo facendo, verso Torino, verso il Politecnico. Mi emoziona pensare di aver completato un percorso che tu hai portato a termine piu' di 50 anni fa, da sola, e come una delle prima donne laureate al Politecnico di Torino. Grazie per aver dato il "la" al mio percorso; senza, non avrei potuto iniziare.

Contents

Chapter 1

Introduction

Soon after the beginning of the computer era, a key question arose: how can we securely authenticate to these machines? This question has driven decades of development, leading to many different authentication methods. Among these, passwords have stood out and remained prevalent to this day. Despite their widespread use, security experts have long recognized the flaws in password-based systems. These flaws include vulnerability to attacks such as phishing, brute force, and password reuse across multiple sites, which expose users to significant security risks. The inherent weaknesses of passwords are combined with the human factor, as users often choose weak passwords and reuse them across multiple platforms, further compromising security.

Recently, a new trend has emerged in the field of authentication-passwordless technologies. This trend aims to address the shortcomings of traditional password systems. Passwordless methods, which will be explored in detail later, represent a major step forward in security. They offer the potential to create more secure systems by eliminating the reliance on passwords, thus reducing the risk of attacks. Passwordless authentication methods leverage advanced technologies and combine concepts known for decades such as biometrics, cryptographic keys, and hardware tokens.

Moving to passwordless technologies is not just a theoretical improvement but a practical necessity in the face of increasing cyber threats. As cybercriminals become more advanced, the need for robust, user-friendly, and secure authentication methods has become crucial. Passwordless technologies, such as biometric authentication, security keys, and single sign-on solutions, provide a multi-layered approach to security that improves both user experience and system safety. These methods also address the usability issues associated with passwords, such as the burden of remembering complex combinations and the inconvenience of frequent password changes

This thesis aims to provide a thorough analysis of passwordless technologies, looking at their development, implementation, and the real-world impact they are expected to have. By examining the historical context of password use, the weaknesses of password-based systems, and the innovative solutions offered by passwordless methods, this work highlights the importance of transitioning to these new technologies.

The following sections will explore the evolution of authentication methods, the central role of passwords, and the security challenges they present. Various passwordless technologies will then be introduced, with a discussion on their integration and categorization. This allows for a focused discussion on the most important passwordless technologies.

After analyzing the literature findings, the first contribution of this work will begin with an in-depth analysis of the architecture that supports passwordless technology. This will identify key aspects to consider when transitioning to this new authentication method. Subsequently, the focus will shift to the transition process itself, examining the key elements of a passwordless architecture, minimum requirements, and a general strategy for adoption. Additionally, the risks and costs associated with this change will be assessed, including potential impacts on legacy systems and the need for ongoing maintenance and updates.

After gaining sufficient background and theoretical knowledge, the second contribution will focus on a Proof of Concept (PoC). The PoC will be a real-life implementation for a Microsoft Belgium client, referred to as "GripGotham" for privacy reasons. This opportunity will allow us to put into practice all the concepts described earlier, ultimately leading to an implementation on the customer side. The importance of this chapter is crucial as it demonstrates how a good understanding of the technology does not lead to a successful implementation, if the supporting architecture is not in place. The PoC will also provide insights into the practical benefits and potential pitfalls of deploying passwordless technologies in a corporate environment.

The ultimate goal of this work is to advocate for passwordless technologies and show the potential of this evolving technology. By combining academic findings with real-life implementation, this thesis aims to provide a comprehensive understanding and valuable insights for approaching this topic. The successful deployment of passwordless authentication methods can lead to more secure digital environments, reduced operational costs, and enhanced user satisfaction, marking a significant milestone in the evolution of cybersecurity practices.

Soon after the beginning of the computer era, a key question arose: how can we securely authenticate to these machines? This question has driven decades of development, leading to many different authentication methods. Among these, passwords have stood out and remained prevalent to this day. Despite their widespread use, security experts have long recognized the flaws in password-based systems. These flaws include vulnerability to attacks such as phishing, brute force, and password reuse across multiple sites, which expose users to significant security risks. The inherent weaknesses of passwords are combined with the human factor, as users often choose weak passwords and reuse them across multiple platforms, further compromising security.

Recently, a new trend has emerged in the field of authentication-passwordless technologies. This trend aims to address the shortcomings of traditional password systems. Passwordless methods, which will be explored in detail later, represent a major step forward in security. They offer the potential to create more secure systems by eliminating the reliance on passwords, thus reducing the risk of attacks. Passwordless authentication methods leverage advanced technologies and combine concepts known for decades such as biometrics, cryptographic keys, and hardware tokens.

Moving to passwordless technologies is not just a theoretical improvement but a practical necessity in the face of increasing cyber threats. As cybercriminals become more advanced, the need for robust, user-friendly, and secure authentication methods has become crucial. Passwordless technologies, such as biometric authentication, security keys, and single sign-on solutions, provide a multi-layered approach to security that improves both user experience and system safety. These methods also address the usability issues associated with passwords, such as the burden of remembering complex combinations and the inconvenience of frequent password changes.

After this brief introduction, Chapter 2 introduces the topics and concepts that will trigger the need for passwordless authentication: the widespread use of passwords, their inherent flaws, and the evolution of authentication methods aimed at addressing these issues. The chapter sets the stage by highlighting the vulnerabilities associated with password-based systems, including their susceptibility to various attacks. As attackers grow more sophisticated, the need for more advanced and secure authentication mechanisms becomes clear. Passwordless technologies, leveraging methods such as biometrics, cryptographic keys, and hardware tokens, offer a promising solution to these challenges. After an extensive discussion on the technologies behind passwordless authentication, the chapter concludes by framing the thesis's core objective to explore the potential of passwordless systems in addressing the shortcomings of traditional authentication.

Chapter 3 develops the first contribution of this work: a structured analysis of architectures that support passwordless authentication. It offers an in-depth overview of one-time code authentication, biometric methods, security keys, and other alternatives. Each technology is categorized based on its functionality and use case, and the advantages of each method are discussed in detail. For example, biometric authentication eliminates the need for passwords by using unique physiological or behavioral characteristics, while security keys provide a physical component to securely store the keys. Each of them requires a specific set of services to work properly and their coherent combination is of paramount importance. This chapter lays the groundwork for future theoretical discussion and outputs a framework that can be leveraged during a passwordless transition.

Following the theoretical discussions, Chapter 4 shifts the focus to the practical implementation of passwordless systems, presenting a detailed case study involving a real-world Proof of Concept (PoC). The case study centers on the transition to passwordless authentication for a large enterprise, referred to as "GripGotham" for privacy reasons. This chapter highlights the importance of properly assessing the current architecture before undertaking such a transition. The PoC demonstrates the practical challenges of deploying passwordless technologies, and allows to apply the framework developed in Chapter 3.

Finally, the thesis concludes in Chapter 5 by emphasizing the significance of transitioning to passwordless authentication in today's cybersecurity landscape. It highlights the necessity of careful planning, risk mitigation, and strategic execution in adopting these systems. The insights gained from the GripGotham case study provide valuable lessons for organizations looking to make a similar transition, offering a framework for navigating the complexities of implementing passwordless authentication. The thesis highlights the importance of ongoing research and development in this field, suggesting that while passwordless technologies represent a major step forward, there is still much to be explored in terms of their broader application and impact on the cybersecurity ecosystem.

The ultimate goal of this work is to advocate for passwordless technologies and show the potential of this evolving technology. By combining academic findings, a novel framework and a real-life implementation, this thesis aims to provide a handbook to approach this topic. The successful deployment of passwordless authentication methods can lead to more secure digital environments, reduced operational costs, and enhanced user satisfaction, marking a significant milestone in the evolution of cybersecurity practices.

Chapter 2

Background and Related Work

This section aims to review the necessary knowledge for the discussions in the "Method" and "Proof of Concept" chapters. While some prior knowledge of cybersecurity is assumed, an effort is made to ensure that major topics are covered, providing the background needed to understand passwordless authentication. The literature covers many topics, and for further interest, the bibliography section lists all the articles, books, and websites used in this chapter. Special emphasis is placed on the reasons behind the shift away from password-based authentication, with supporting evidence and examples provided where needed.

This chapter starts by describing the central role of passwords in modern authentication. It then highlights the weaknesses of these systems, presenting them as challenges. Notable examples of failures in mitigating password vulnerabilities are discussed. After examining the best efforts to address these problems, passwordless technologies are introduced, and their fit within modern cybersecurity frameworks is explained.

To complete this chapter, a categorization of passwordless authentication methods is provided, along with an analysis of current technologies. These discussions will be particularly useful in the following chapters, as understanding the landscape of passwordless technology is crucial for discussing the architecture needed to support them.

2.1 The central role of passwords

As mentioned in the introduction, authentication methods have evolved significantly over the centuries, driven by technological innovation and changing landscapes. Ensuring proper user authentication has remained a top priority for security professionals. However, due to historical reasons, the main focus has been put on an imperfect authentication method: Passwords [1]. As the automotive industry has adopted an inefficient fuel to power its engines, modern architectures have evolved to adopt and fully operate the duet "Username and Password".

Dating back to the 60s, passwords have been introduced since the early stages of the development of operating systems, mainly to protect from jokes and misuse of devices. Over time, passwords have been established as an official login method to safeguard files, data, and resources from unauthorized access. Today, we closely link passwords with accounts and accounts with online identities. Even the most advanced identity protection tools incorporate some form of password security.

Many individuals encounter passwords early in their digital journey, whether they are setting up a user account on a personal computer or accessing vital services such as their national healthcare system. This initial interaction with passwords marks the beginning of their digital footprint, emphasizing the fundamental role of passwords in securing personal information and ensuring access to essential resources.

2.2 Is password-based authentication secure enough?

The previous section should have made clear that passwords have been rooted in our lives and systems for decades. Having acknowledged this, a trivial question may arise: are passwords secure enough? Should we rely on them to secure our identities? This section aims to answer these questions and provides real-world data to support the answers.

Nowadays, it is easy enough to state that passwords alone are fundamentally insecure for several reasons. As stated by the work of Joseph Bonneau and Cormac Herley [1], the reason why passwords are not secure enough is a combination of users' usage, systems designs, and attacker's strength. Throughout this chapter, it will become clear that this represents a deadly combination for password-based systems. Hereafter are mentioned some of the most important flaws.

2.2.1 Flaws in password-based authentication

Users often select weak passwords that are easy to guess, such as common words or simple patterns, making them highly vulnerable to attacks. Furthermore, they tend to reuse passwords across multiple sites, increasing the risk of cross-site compromises. If one site is breached, the same password can be used to access other accounts. A significant problem is the susceptibility of passwords to various forms of attacks, including phishing, where attackers trick users into divulging their passwords through deceptive emails or websites. Malware also poses a threat by directly stealing passwords from the user's device through keyloggers or other malicious software. Additionally, passwords can be intercepted during transmission, especially if encryption protocols are not properly implemented, or stolen from poorly secured servers, exposing vast amounts of user data.

Offline attacks present another challenge. If an attacker gains access to a password database, they can employ powerful computational resources to guess passwords through brute-force attacks or by using precomputed tables known as rainbow tables, which are particularly effective if the passwords are not properly hashed and salted. Hashing converts passwords into a fixed-size string of characters, while salting adds random data to ensure that identical passwords result in different hashes. Even strong passwords are not immune to these methods, highlighting the need for more sophisticated security measures. Attackers can leverage modern computing power to crack even complex passwords relatively quickly, demonstrating the inadequacy of relying solely on password strength.

The complexity of maintaining secure passwords also imposes a burden on users, who must remember multiple, often complex passwords or resort to insecure practices like writing them down. This cognitive load can lead to poor password management practices, such as creating predictable variations of the same password. Many researchers have exposed the consequences of this behavior over the course of the previous decade.

2.2.2 Remarkable examples

A notable study conducted by researchers at the University of Chicago [2] examined data breaches across hundreds of websites, focusing on university email addresses. They successfully guessed passwords for 32.0% of accounts associated with university emails found in data breaches, and for 6.5% of accounts with matching usernames by cracking hashed passwords and adjusting guesses. Many of these accounts remained vulnerable for years after the breaches, with passwords found verbatim being nearly four times more likely to be exploited than modified guesses. The study, which covered over 70 different breaches, included surveys of 40 users whose passwords were compromised. The surveys showed that many users were unaware of the risks to their accounts or that their credentials had been exposed. The findings offer practical advice for organizations on protecting accounts.

Given these vulnerabilities, the security community widely agrees that passwords alone are insufficient for protecting online identities.

2.3 Failures in solving the problems

After acknowledging that passwords are not a perfect authentication system, there is a need to strengthen them by mitigating the problems described above. The flaws cited in the previous paragraph are only some of the reasons why security professionals have been challenged to find a more robust authentication paradigm. Significant progress has been made in recent years to enhance this outdated method of user authentication. Additional layers can be included to delay, if not prevent, some of the vulnerabilities that persist in password-based authentication systems. Hereafter are some examples of our best attempts at mitigating password-related problems Specifically, the argument will be constructed around three main topics: **Password managers**, **Password strength meter**, and **Multi-Factor authentication**. Each of them is accompanied by an introduction that states the intention and scope of the mitigation, as well as the reasons that lead to an "imperfect" solution.

2.3.1 Password managers

Password managers have been around for decades and are a direct response to the increasing length trend that passwords have experienced. From a computational point of view, the time to crack a password is proportional to its length, thus, longer passwords are computationally harder to break compared to smaller ones [3]. Extending the length of the password comes with a problem, as the systems get stronger, with more robust passwords, users struggle to remember them. This task is perfectly solved with the introduction of a password manager. They eliminate the need for users to remember or write down complex strings of random letters, numbers, and symbols by securely encrypting and storing them [4]. With the expansion of online services, most password managers nowadays focus on the synchronization properties, allowing us to have all passwords shared among our devices [5]. Most importantly, nowadays password managers encourage users to adopt unique and (pseudo-)randomly generated strong passwords instead of choosing them for each account on websites, applications, or systems. Additionally, modern password managers allow the user to avoid interaction with the password itself. Auto-filling is a common function that is activated after the user's identity is verified via a master password or biometric credentials. All modern operating systems and web browsers include built-in password managers, which users are encouraged to utilize. Additionally, third-party options such as Bitwarden, NordPass, and many other 'freemium' solutions are largely available to the public.

[5] With most of the modern password managers, the functionality of strength meters is provided. Usually, they come with color-based graphics, red indicating very poor security, orange corresponding to a medium level, and green reserved for robust passwords. This functionality works both with user input and with generated strings, allowing us to modify the password that has been provided, knowing which category it will fall in, or simply construct a robust password and have an estimate of its security. This last point is particularly misleading and can be dangerous as it may give scores that are far away from the real complexity that an attacker faces. Further details will be given in the following section "Password Strength Meters".

While password managers seem a convenient way of solving the problem of remembering passwords, they introduce a single point of failure, as the compromise of a password manager can expose all stored passwords. Users must also trust that the password manager itself is secure and free from vulnerabilities. Those concerns cannot be ignored when dealing with online password managers (which constitute the majority of the market share). Remarkable examples of data breaches include *Norton LifeLock*, *LastPass* and *BitWarden*, which are ranked in the top 10 of most used online password managers [6]. These breaches have led to massive password leakage and have provided attackers with more data for future attacks. The single point of failure inherent in password managers is a significant design flaw, especially when combined with users' lack of awareness and the potential for zero-day exploits on the vendor side.

2.3.2 Password strength meters

Password strength meters are tools or algorithms that evaluate the robustness of a password by analyzing several factors. These factors typically include the length of the password, the variety

of character types used (such as uppercase and lowercase letters, numbers, and symbols), and the unpredictability or randomness of the character sequence. The evaluation results are usually presented in a visual format, often as a color-coded bar or numerical score, to indicate whether a password is weak, moderate, or strong.

[7] There are various aspects that are taken into account when evaluating a password, among them, we can find:

- **Search Space Size:** Evaluation involves measuring the size of the search space, which is the number of guesses an attacker might need to make.
- **Diminishing Returns:** Weak passwords are common and easily guessed. However, as the attack progresses, the success rate of each additional guess decreases significantly, making it increasingly costly for attackers to continue guessing.
- **Effectiveness of Attacks:** The evaluation of password strength often involves comparing different known attack techniques using datasets of known passwords.

However, despite our best efforts, password strength meters often fail to accurately predict how real attackers operate [8]. This is because they don't consider the personalized methods attackers use. In targeted attacks, hackers usually start by gathering basic information about their victims from social media and other online sources. With this knowledge, real-world attackers often use sophisticated techniques such as dynamic dictionary attacks and adaptive mangling rules. These advanced methods simulate the behavior of human beings and allow them to adjust their attack strategies based on the information gathered. For example, dynamic dictionary attacks involve continually updating the dictionary with new terms that are relevant to the target, while adaptive mangling rules use deep learning to generate customized transformations for each dictionary word in real time. These techniques make it clear that traditional password strength meters, which rely on static rules and predefined configurations, fall short of providing an accurate measure of password security. To cite a practical example, a password like **"DaV_01_IdEItA"**, which might score a high mark on standard meters, is actually easier to crack if the attacker knows basic details like the victim's name, nationality, or birth date. While password strength meters can simulate basic attacks using dictionaries or spraying passwords, they don't account for the advanced tactics used by attackers who target specific users.

Consequently, efforts have been made to design better meters that take into account real-world adversaries and their knowledge. Some remarkable examples include tools to properly model attackers with the use of Machine Learning [9] and a redesign of a password strength meter around sensitive information [10].

2.3.3 Multi-Factor authentication

Multi-factor authentication has long been our best effort to patch the inherited risk of using passwords. The core idea of MFA comes from the broader concept of authentication, which involves verifying a user's identity by proving they possess certain credentials. These credentials can be divided into three main categories [11]:

- **Something that the user knows:** Generally referred to as a PIN or a password, or any type of secret that the user possesses.
- **Something that the user has:** Usually a physical token or a second device such as a smartphone or tag.
- **Something that the user is:** Typically biometric data like fingerprints, facial traits, voice, or the iris details.

When MFA uses only two authentication factors, it is called two-factor authentication (2FA). 2FA combines two different types of credentials to verify a user's identity. For example, it might

require a user to enter their password (something they know) and then confirm their identity by entering a code sent to their mobile phone (something they have). This extra layer of security ensures that even if a malicious actor obtains the user's password, they would still need the second factor to access the account. Commonly, the approaches to implement MFA include SMS-based verification, where a one-time code is sent to the user's mobile phone; push notifications, which prompt the user to approve a login attempt on their mobile device; and authentication apps, such as Google Authenticator, which generate time-based one-time passwords (TOTPs). Biometric verification, like fingerprint or facial recognition.

Multi-factor authentication is a huge security enhancement compared to a plain username and password and this has been largely shown within the academia. In a recent study on the effectiveness of multifactor authentication (MFA) in deterring cyberattacks [12], the researchers from Microsoft focused on evaluating how well different MFA methods protect commercial accounts, particularly those with known credential leaks. They utilized a private dataset from Microsoft Entra ID(formerly known as Azure Active Directory), combining the benchmark-multiplier method with manual account reviews to assess the security performance of various MFA techniques. The study included a significant period of observation and detailed analysis of accounts showing suspicious activity. The key findings highlight that MFA provides substantial protection against unauthorized access. Over 99.99% of accounts with MFA enabled remained secure during the investigation period. The overall risk of compromise was reduced by 99.22% with MFA, and by 98.56% for accounts with leaked credentials.

In a recent security report [13], Google researchers, in collaboration with New York University and the University of California, emphasized the critical role of MFA in protecting user accounts. Their study revealed that adding a recovery phone number to a Google account can block 100% of automated bots and 99% of bulk phishing attacks. Additionally, SMS codes sent to recovery numbers stopped 100% of automated bots and 96% of bulk phishing attempts. On-device prompts, considered more secure than SMS due to the risk of SIM swapping, successfully thwarted 100% of automated bots and 99% of phishing attacks.

The numbers clearly show that multi-factor authentication (MFA) is crucial for securing our online identities. However, similar to the challenges with password strength meters mentioned earlier, we struggle to model the actions of sophisticated attackers accurately. Most research, including the two studies referenced here, focuses on automated bots and bulk phishing attacks. These are easier to test and help us understand the basic security features of a system. However, as shown by the last three annual reports of the European Union Agency for Cybersecurity (ENISA) [14][15], targeted attacks are becoming more effective and impactful, often using advanced techniques like artificial intelligence.

This analysis is aligned with recent publications from RSA Security [16], showing that in targeted cyber attacks, the effectiveness of multi-factor authentication (MFA) can drop to less than 50% under certain conditions. Specifically, attackers significantly increase the chance of a successful attack in case of misconfigurations, systems vulnerable to supply chain attacks, or fatigue attacks.

Having considered the data mentioned above, we can conclude that MFA is a significant boost in the security of modern systems. However, real-life implementations are prone to vulnerabilities and attacks that may hinder the benefits and void the economic and technical efforts.

2.4 Introduction to passwordless technologies

The previous sections should have highlighted the efforts security professionals have made to secure password-based systems. Despite these efforts, such systems remain highly vulnerable and susceptible to human errors, often leading to breaches or failures of security controls. Traditionally, we think of passwords as secrets that must be kept safe, whether they're stored in a password manager, a database, or written on a sticky note. Passwordless authentication aims to move away from this concept, enhancing security by not relying on shared secrets for authentication. Instead of using passwords, passwordless methods use different ways to verify identity, such as devices

we own, biometrics like fingerprints, or other advanced techniques. The term "passwordless authentication" encompasses a wide range of technologies and methods that change how we handle and use passwords. These methods eliminate the effort related to using a common secret for authentication. The complexity of generating, sharing, and remembering passwords is removed from the process. Users are not involved in most operations and have limited control over the authentication process. With passwordless authentication, the "username and password" combination is no longer necessary. Depending on the specific technology, it may even be irrelevant to think about a unique combination of username and secret.

Some common methods of passwordless authentication include using a smartphone to receive a one-time code, using a fingerprint scanner, or having a secure device that generates unique codes. These methods can make it much harder for attackers to gain unauthorized access since there's no single password to steal or guess [17]. While some of these technologies are still new and not yet widely used, it is important to understand them to see where the future of authentication is heading. The following sections will provide a complete view of passwordless authentication, including both popular methods and those that are still developing.

2.4.1 Passwordless technologies in the zero trust framework

Before diving into the technical details of passwordless authentication, it is important to recognize how well these technologies fit within the Zero Trust framework. This knowledge is fundamental to grasp how this necessary innovation aligns with one of the most popular cybersecurity paradigms today: the Zero Trust Architecture.

Zero Trust represents the evolution of static and perimeter-based security controls into dynamic and continuous assessments. Over the years, five main pillars have been defined: identity, device, network, application and workload, and data [18]. Each pillar demands its own set of security controls and assessments, with the key principle being that no implicit trust is carried between them. [19] This foundational concept of "never trust, always verify" has enabled organizations to advance their security postures significantly, disrupting traditional attack models and fortifying defenses against modern threats.

Within the identity pillar of Zero Trust, there has been a notable shift-left approach that emphasizes the importance of user identities and continuous verification. With the term "shift-left" we refer to the action of shifting the security controls over the user side, rather than relying on its implicit trust. This focus places passwordless technologies in a privileged position within the framework. By eliminating the need for users to remember and manage passwords, passwordless authentication reduces the "human" factor in the authentication process, which is often a significant vulnerability. Instead, methods such as biometrics, smart cards, and mobile authenticators enhance security and streamline the user experience. When users do not have to manage passwords, the attack surface for phishing and other credential-based attacks is drastically reduced. Instead, authentication methods that are harder to compromise, such as fingerprints, facial recognition, and secure hardware tokens, take their place. Since adherence to the Zero Trust Framework is important to increase the security posture of an organization, a further discussion will be carried out in one of the following chapters (3.5).

2.4.2 Overview of the main passwordless technologies

This section provides an overview of different passwordless technologies. As part of the introductory chapter on this topic, the focus is on describing the main features and uses of each technology and categorizing them accordingly. More detailed architectural aspects are treated in the section "Architecture of Passwordless Technology". The classification has been made by combining a variety of studies and researchers that focused on standalone passwordless technology. After a careful analysis of available literature and sector-specific journals, a spectrum of passwordless authentication method arises. The final categorization is the following:

- **One-Time code authentication**

- **Long-Term code authentication**
- **Biometric authentication**
- **Security keys**
- **Session property authentication**

What follows is a description of each category and the main solutions that belong to them. To keep explanations straightforward and to offer clear, consistent examples across the various technologies, the perspective taken is one of a client authenticating to a server.

Category 1 - One-Time code authentication

One-time codes are an effective and common solution adopted to enhance the security of password-based systems. They are typically used as a second-factor authentication method, configured after a primary password for a system or service. Notably, one-time codes, if used alone, can be transformed into a passwordless authentication method by slightly modifying their basic specifications [20]. As the users are already aware of this technology, the usability gap is extremely slim and One-Time code can be an effective first step towards passwordless authentication. According to their initial design, One-Time code work by generating a single, disposable token that is used to answare a simple challenge-response protocol. Modern implementations of stand alone One-Time code leverage additional server-side mechanisms to eliminate the need for this pre-shared secret. The basic principle of operation is as follows:

- **Initialization**: The user's secret passphrase is combined with a seed provided by the server. This initial combination does not need to be transmitted over the network, thus preventing exposure to eavesdropping.
- **Hashing**: The combined passphrase and seed are processed through a secure hash function multiple times. This creates a sequence of one-time passwords (OTPs), where each subsequent OTP is generated by reducing the number of hash iterations by one.
- **Challenge and Response**: During authentication, the server issues a challenge to the user. This challenge includes a sequence number and a seed, which the user's client device uses to compute the corresponding OTP.
- **Verification**: The server verifies the OTP received from the user by hashing it once and comparing the result to the stored previous OTP. If they match, the user is authenticated, and the server updates its records with the new OTP.

This schema has proven to be extremely effective in providing a robust second-factor authentication method. However, as mentioned earlier, it can be adjusted to avoid the need to initialize secrets. Specifically, the schema can be modified to eliminate the Initialization and Hashing phases. Similar to the evolution of Time-based OTP, the server can generate a one-time code that is closely linked to a time constraint and is delivered to the user, who can then authenticate. These codes are generally valid for a single session and login attempt, failing if used multiple times, thereby protecting the user from replay attacks. Users have the flexibility to choose the medium for transmission, which could be Email, SMS, or Push Notification. Depending on the medium, we can refer to them as OTPvE, OTPvSMS, and OTPvN. Real implementations include services like Adobe and Oracle Cloud that leverage One-Time Code to access their services [21] [22]. The architectural details will be discussed further in a subsequent section.

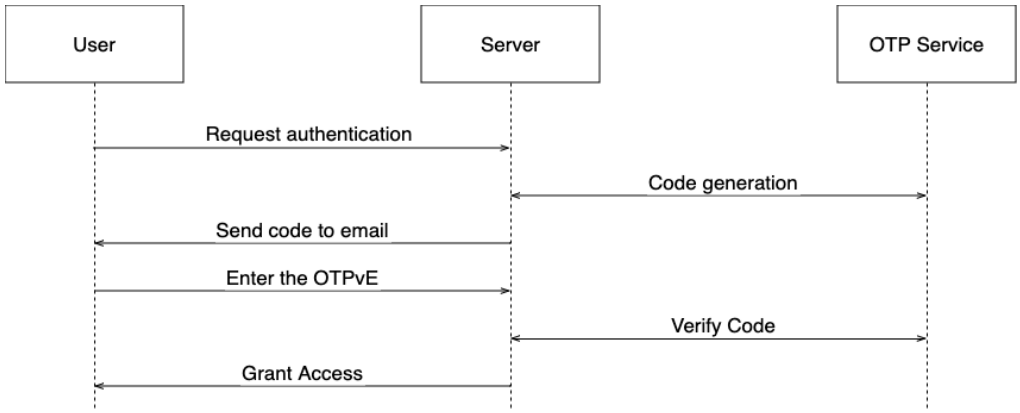

Figure 2.1. One-Time code via Email, basic authentication schema.

Category 2 - Long-Term code authentication

Long-term code authentication represents a category of authentication methods aimed at extending user access from One-Time code single use, to multiple uses and sessions. While the time property remains a crucial factor in validating a request, much like in One-Time codes, literature, and existing resources do not specify an exact threshold for what constitutes a "long-term code" in terms of duration. However, practical implementations and real-life examples demonstrate a variety of choices depending on the implementation. Two primary technologies fall into this category:

- **Token-based authentication**
- **Magic link**

Token-based authentication enables users to request a token that serves as a "passepartout", allowing them to access various services and applications until the token expires. Tokens, typically possessing longer lifespans than other codes, can be tailored to specific scenarios or constraints, such as particular service types or geographical locations like a company office [23]. This flexibility ensures that tokens can be precisely scoped to limit their use to intended contexts, enhancing security and usability. Once a user obtains a token, a validation component is responsible for ensuring that access is granted if the token's requirements are fulfilled. This process involves checking the token's validity, expiration, and scope to confirm that it meets the necessary conditions for access. Token-based authentication has proven particularly beneficial in securing machine-to-machine (M2M) communications. In M2M platforms, tokens facilitate secure interactions between devices and servers [24], allowing developers, administrators, and users to use the token as a single interface for authentication. This approach eliminates the need for other secrets or passwords in the architecture, simplifying security management and enhancing overall system integrity.

Magic links operate in a manner similar to token-based authentication. They are a relatively new technology that is gaining traction for its low cost and low effort characteristics in the implementation of passwordless technologies. When a user requests a magic link, the system generates the link with specific parameters tailored to the user's request and sends it to their registered email or phone number. [25] This link allows the user to access the application seamlessly without needing to enter a password. The generation of magic links typically occurs on the fly and can include properties such as geolocation, browser type, and device information. These properties ensure that the link is uniquely tied to the context in which it was requested. [26] On the server side, a dedicated component handles the creation and validation of these magic links. This component ensures that the generated link contains the necessary parameters and properties for secure authentication. When the user clicks the link, it directs them to a specific endpoint within the application. The embedded properties within the link are then validated against the

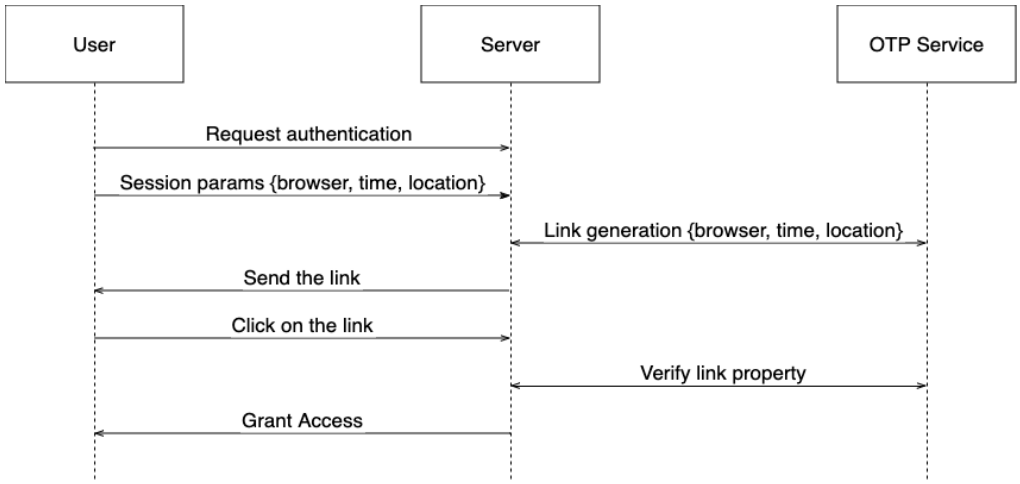

Figure 2.2. Magic Link via Email, basic authentication schema.

original request parameters, confirming the user's identity and granting access if the link meets all security requirements.

Category 3 - Biometric authentication

Biometric authentication has become a key technology in our authentication systems, as for the One-Time code, it is often chosen as the second factor of authentication in an MFA architecture. Biometric authentication is by default a passwordless authentication method and can play a vital role in the removal of passwords. Biometric authentication leverages measurable characteristics of the individual to prove its identity to the counterpart [27]. Specifically, they can be categorized into two types:

- **Physiological:** related to the physiognomy of the body and leverages the fact that each one of us has unique characteristics including fingerprints, iris and retina scans, facial recognition, and hand geometry.
- **Behavioral:** related to how each of us behaves under certain conditions and include signature recognition, voice recognition, keystroke dynamics, and gait analysis.

A key aspect of biometric authentication is the enrollment and matching process. The enrollment process involves capturing a reliable biometric sample and associating it with an individual's identity. This process must be rigorous, especially in high-security environments, to ensure the accuracy and integrity of the biometric data. For instance, fingerprint enrollment may require multiple scans of the same finger to average out any anomalies. The matching process compares a new biometric sample (the bid sample) with the stored template using probabilistic methods. [28] This comparison is based on the Hamming distance, which measures the degree of difference between the samples. The system must balance the false acceptance rate (FAR) and false rejection rate (FRR) to ensure both security and usability. A novel aspect, which is shared with Security Keys, is that with Biometric authentication, we rely on the quality of the sensors that are performing the reading. A faulty sensor can mismatch the trait that is under measure and allow an intruder to impersonate the real user. On the other end, due to the ever-changing property of some of our characteristic traits, the sensors should allow for some flexibility during the reading and matching phase.

Furthermore, with this type of passwordless authentication, there is a significant concern regarding the security standards that must be in place to ensure information remains private and secure. [29] Biometrics, unlike the other technologies discussed in this work, are not replaceable and offer little to no ability to change in the event of a compromise. Additionally, biometric data

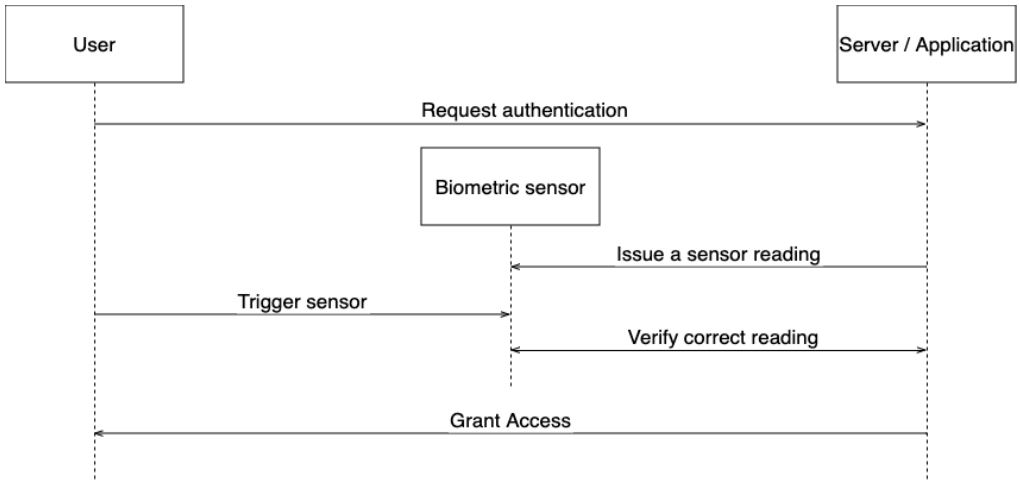

Figure 2.3. Biometric reading, basic authentication schema.

represent a valuable asset capable of correlating individuals, which may lead to misuse if it is not properly managed and protected.

Category 4 - Security keys

Security keys represent a distinct category of authentication methods, characterized by the use of physical devices to facilitate the authentication process. Like other advanced authentication methods, security keys have undergone significant technological development, positioning them in a privileged position in the passwordless scenario [30]. Security keys are engineered to securely store secrets by isolating them within a physical component. The integration of hardware and software simplifies the complexity of secure hardware enclaves, making them more user-friendly. Notably, security keys uniquely link authentication to a physical device under the user's control. All security keys share the same working principles:

- **Generation phase:** The hardware is involved in the generation of they key by leveraging small manufacturing deviations to create unique keys. Alternatively, user input can be the source to generate the key for a service.
- **Storage phase:** The hardware is responsible for storing the secrets in a secure environment, resistant to hardware tampering.
- **Retrieval phase:** The user may need to authenticate to unlock the key and allow for it to be used, either by streamlining it or by answering a challenge.

The properties of security keys are inherently tied to the presence of the hardware token. Specifically, security keys are considered "isolated", meaning the operating system (OS) or browser interacts with them solely through a software interface [31]. This isolation ensures that secrets typically remain non-portable, and most implementations prevent the key from leaving the device. Consequently, losing the physical device often necessitates a system reset or recovery of the system protected by that key.

These unique characteristics have led to the emergence of certain standards in recent years. Specifically, the FIDO (Fast Identity Online) and Passkey standards are the most widely used and up-to-date protocols for implementing passwordless authentication with security keys. The primary distinction between these standards lies in the portability of the keys [32]. FIDO2-compliant keys are designed to prevent key export, ensuring that the key remains within the device. Conversely, Passkey-compliant keys can be recovered and backed up. Both standards support software vaults and hardware keys, and they ensure multi-factor authentication (MFA) by pairing authentication with biometric verification.

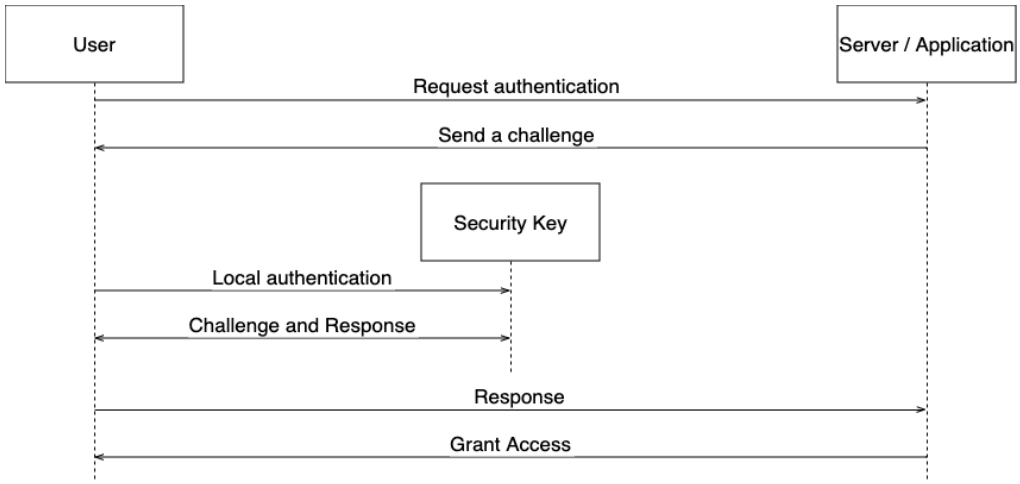

Figure 2.4. Security Key challenge response, basic authentication schema.

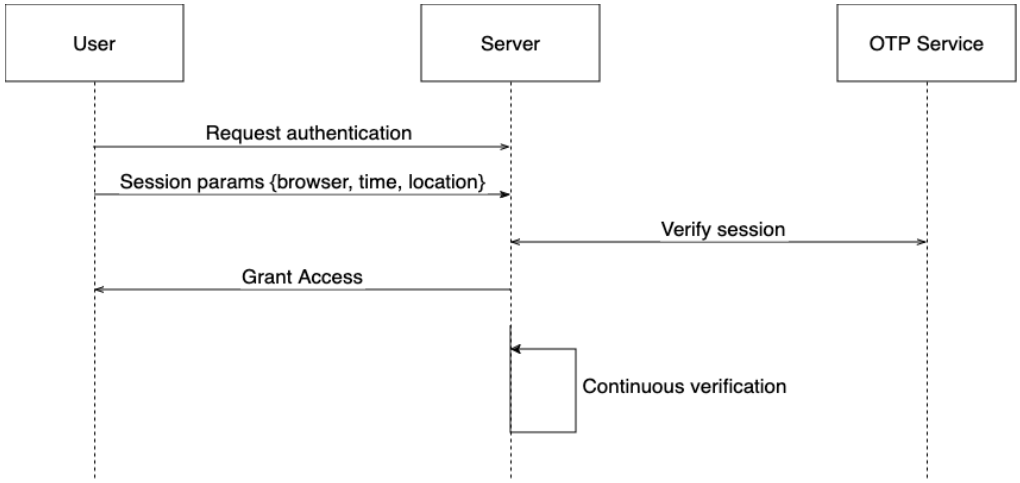

Figure 2.5. Session property evaluation, basic authentication schema.

Category 5 - Session property authentication

Session property authentication focuses on verifying a user's identity by examining a range of properties related to their session. These properties include network attributes, device characteristics, and other contextual factors that remain constant during a session. By leveraging these stable properties, additional verification layers can be implemented, significantly enhancing security. This method integrates various data points such as IP address, geolocation, time properties, and the specific application or client from which the requests originate. [33] These techniques are similar to "device fingerprinting", allowing the server to accurately verify the user's identity by creating a unique profile based on the session's attributes. For example, the server might recognize a particular combination of device type, browser version, and IP address as a unique identifier for a session.

Session property authentication is entirely transparent to the user, providing a seamless experience without requiring additional input or actions from them. This authentication method is entirely transparent to the user and can be paired with other strategies to provide a tailored experience, such as recognizing trusted locations or specific clients. While not yet widespread, session property authentication is mentioned for the sake of completeness.

2.4.3 State of the art for passwordless technologies

After categorizing each passwordless solution, it becomes evident that many can be combined to enhance authentication strength. Conversely, some technologies have limited use and scope, confining their market share to specific cases. Evaluating the current market and technical maturity of these solutions reveals three standout combinations [34][35]:

- **Security keys + Biometrics:** Many security keys incorporate biometric authentication, such as fingerprint recognition, transforming them into standalone multi-factor authentication (MFA) devices. This combination leverages the inherent security of physical devices and the convenience and security of biometric verification. These keys provide a robust authentication method that is both secure and user-friendly.
- **One-time code + Biometrics:** Authenticator apps commonly merge one-time code generation with an additional biometric authentication layer. This integration ensures that even if the one-time code is intercepted, unauthorized access is prevented without biometric verification. The combination enhances security while maintaining ease of use for the end-user.
- **Long-term code:** Magic links and certificate-based authentication excel in specific environments, such as those with existing Public Key Infrastructure (PKI) or minimal authentication development. Magic links provide a seamless user experience by sending authentication links to the user's email or phone, which, when clicked, grants access without the need for a password. Certificate-based authentication, on the other hand, leverages digital certificates to authenticate users, providing a high level of security for applications with stringent security requirements.

Following this, the next chapter will focus on these three combinations-Security keys with Biometrics, One-time code with Biometrics, and Long-term code authentication-as they represent the best-in-class passwordless technologies, including all the architectural components that have to be taken into account in the passwordless transition (3.1).

2.4.4 Passwordless does not imply MFA

Before diving into the architectural analysis, it's important to address a common misconception: Passwordless authentication is not the same as Multi-Factor Authentication (MFA). While it may seem intuitive to equate the two, especially since many popular passwordless solutions, like FIDO2 security keys, are inherently MFA-compliant, this assumption is not accurate. The compliance of FIDO2 devices with MFA standards is a result of the FIDO Alliance's specifications, rather than an intrinsic property of the security keys themselves.

Another aspect that may contribute to this confusion is that certain authentication methods are often reserved as the second factor within an MFA configuration. For instance, biometric authentication, such as facial recognition or fingerprint scanning, is frequently used as a second factor in multi-factor systems, especially for local device unlocking. However, biometrics can also serve as a single factor of authentication in a passwordless setup. A common example is the use of facial recognition to unlock smartphones-here, no secondary factor is needed, making it a standalone passwordless method.

Similarly, SMS codes, often used as a second factor or as a recovery PIN for account access, are another form of passwordless authentication. Though widely associated with MFA, SMS-based authentication stands on its own as a valid method of identity verification without requiring an additional password.

As will become clear in the next chapter, these methods are perfectly valid examples of passwordless authentication, each with its own strengths, limitations, and use cases. They all share a common characteristic: the ability to remove traditional passwords from the equation, enhancing both security and user convenience. Yet, they remain distinct from MFA by design, illustrating that passwordless systems and multi-factor setups, while often working in tandem, serve different purposes and should not be conflated.

Chapter 3

Method

With a solid understanding of passwordless authentication established, we can now explore the practical aspects of implementing this technology. This chapter presents the first significant contribution of this thesis: a focused examination of the infrastructure, the strategy, the requirements, and the architectural components necessary for a successful passwordless transition. We will outline the key elements that will be used to build the Proof of Concept (PoC), including the minimum requirements for initiating the shift to passwordless systems.

It is important to note that the sections are constructed to go further in the level of detail and complexity. This allows a reader with knowledge from Chapter 2 to stop and be satisfied with the details provided by the first paragraphs. If the reader is only interested in the logical level of the architecture, it is best to stop at the end of the next section. Conversely, the subsequent section is advised from a more technical reading. The chapter's core is the discussion carried out in the first section, as it explores all the logical components to implement a passwordless authentication system. Furthermore, the chapter develops some implementation details, a strategy for passwordless implementation, an evaluation of design elements, and a consideration of risks. Additionally, we will address the costs and overhead involved in the transition, as well as the expected benefits in terms of security maturity, using frameworks like the CISA maturity model.

The academic literature on authentication methods is extensive, and with the rapid rise of passwordless technologies, more research is now focusing on these new approaches. However, there has been little work on how to transition to these methods in a unified way. While the technology is well documented, and its pros and cons are discussed in many papers, few have addressed how to actually implement and integrate these technologies into our complex systems. Our modern architectures are made of active directories, hybrid environments, multi-cloud setups, and heterogeneous operating systems that are not always passwordless-ready.

3.1 Architectures for passwordless authentication

As outlined previously, from this section on, the analysis will consider only the best-in-class passwordless authentication method:

- **Security Key + Biometrics**
- **One-Time Code + Biometrics**
- **Long-Term Code**

These three technologies allow us to describe the most complete set of components needed for a smooth passwordless transition. Other technologies mentioned in Chapter 2 can be seen either as simpler versions of these three or slight variations that can be derived from the three major ones. As a reference, the three passwordless architectures that have been chosen will be called

"Major Architectures". The goal of this section is to understand the logical components behind the three major architectures. The key question that a reader should be able to answer at the end of this part is: What are the elements of passwordless authentication architecture?

Some of the elements are common across all of the architectures. Therefore, they are referred to as "Base Services". Other components are specific to one of the three major architecture and are called "Characteristic Services". This distinction will become clear at the end of the section and is properly explained later on (3.1.4). The word "service" in this context refers to a logical or physical component with a specific job (e.g., a server that performs authentication, a database for the backup, biometric software interface to interact with a biometric reader). Separating these services helps us identify the core elements needed for the transition and makes it easier to describe risks and vulnerabilities by addressing each component individually.

Unlike Chapter 2, which focuses on the technical aspects of each authentication method, this section will concentrate on the architectural requirements for deploying these methods. Chapter 2 considered passwordless technologies as stand-alone, on the other hand, the framework of this chapter will be centered around the server-side architecture needed to implement and support them. With the term "architecture" we refer not to the details of encryption or code-generation algorithms but to the functions and infrastructure needed for onboarding users, performing authentication, and managing passwordless authentication. This section is closely linked with the next one (3.2), which describes the characteristics of each service, including their duties and functions.

Settings

To correctly analyze each of the aforementioned methods, it is necessary to take some considerations that will become useful in the following discussions. The first consideration is that any passwordless method will be treated as the primary and only authentication for a user. This allows the removal of pieces of the backend structure that are not involved in the passwordless authentication. This simplification will be later discussed when talking about the strategy to migrate from a password-based system to a passwordless system (3.4). Additionally, machine-to-machine authentication is not treated, hence, the user will be always present in the diagrams. The second consideration is that the users are already part of the "user directory", meaning that they already have an entry that can be updated with the new authentication method. The last consideration is useful for future diagrams: users interact and authenticate to the counterpart with a single point of contact that can be viewed as a proxy. All the logic that sits behind the scenes, which allows for passwordless authentication, is transparent for the user.

Services as Lego Bricks

Before exploring the characteristics of each major architecture, it's helpful to mention that the approach is similar to assembling Lego bricks. The services act as building blocks that come together to form the overall architecture. Some of these blocks are used universally, perhaps in different colors but with the same basic structure (Base Services). Others are unique, resembling other shapes but customized specifically for a particular purpose (Specific Services). It will come in handy to always have a look at the picture provided and find the right "brick" in the image. With this comparison in mind, we can explore the services and their use in passwordless architectures. At the end of this part, it should be clear if a Backup service is needed for a specific design or if a Distribution service must be implemented in order to complete the back-end.

3.1.1 Security Key + Biometrics

When implementing passwordless authentication using Security Key and Biometrics the first discussion includes device compatibility. This is the only authentication method that poses some limits to the device itself, yet, this is crucial to proceed with functionalities like enrollment and authentication. The device must support Security Key authentication and be physically compatible with the key. Biometric capabilities are not directly involved in this phase, as they are

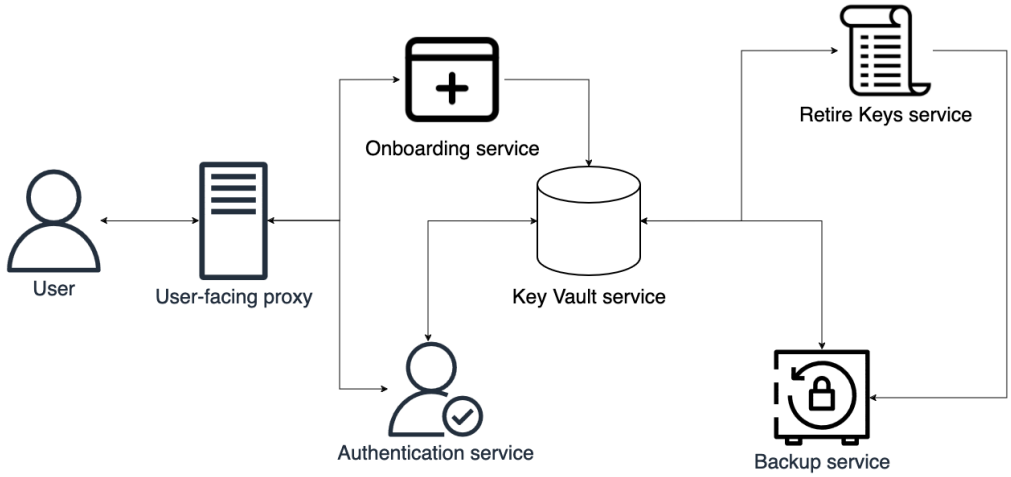

Figure 3.1. Passwordless architecture with Security Keys.

primarily used to unlock the key for authentication. The crucial standards for Security Keys are FIDO2 and Passkeys, so hardware compatibility with these standards is essential for successful integration. The specifics of them have been mentioned in the second chapter (2.4.2).

From an architectural point of view, to properly employ Security Keys, five blocks are needed:

- **Onboarding:** To guarantee key registration both on the device and on the server side.
- **Key Vault:** To properly store and handle keys.
- **Authentication:** To perform the challenges required to authenticate a user.
- **Retire Keys:** To allow for key management and handle lost tokens.
- **Backup:** To allow for portability of the keys and backup among devices (the portability duty is only for passkey-compliant designs).

Sample Architecture

The sample architecture for Security Keys, including the common services, is shown in Figure 3.1.

From the diagram, it's clear that the Key Vault service is a central component of the architecture. As it connects the user's identity with the associated keys, it is used by most of the other services. This makes it the core element and, if we look at the units as nodes, it has the highest number of connections in the graph. The diagram models a direct line between the user and the proxy, not taking into account the specific interaction between the user and the device. As mentioned in section 2.4.2, the properties and usage of security keys are closely tied to the hardware token and, more importantly, to the Operating System or Browser used in the authentication phase. For this reason, the only part of the Authentication service that needs to be implemented is the one that lies on the server side, as the local authentication is delegated to OS-specific methods.

The Retire Keys service takes care of retired keys that can fall into this category in case of compromise or loss. This service directly updates the Key Vault and the Backup by communicating with the correct service. Conversely to the other two designs, the Authentication service only interacts with the Key Vault service to verify the upcoming user. Another distinction between the other two architectures is the presence of the Onboarding service connection. This is mandatory for the first interaction with the user's key, as it is not logically equal to a normal authentication. The two services may be implemented as a single one, yet, the logical distinction holds true.

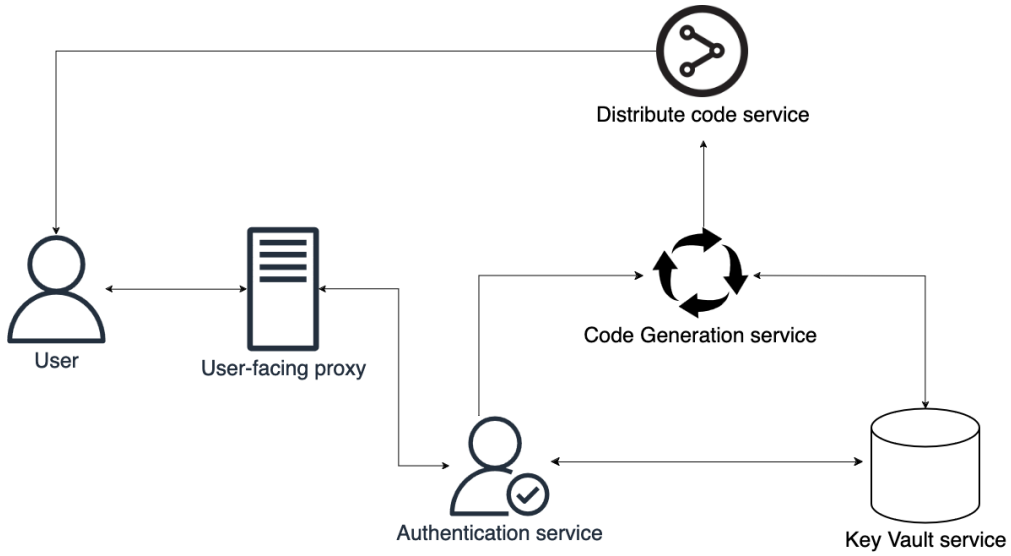

Figure 3.2. Passwordless architecture with One-time codes.

3.1.2 One-Time Code + Biometrics

As mentioned in the previous sections (2.4.2), One-Time Codes are widely used and require minimal adjustments to serve as a primary passwordless authentication method. On the server side, the following blocks must be implemented to use One-Time Codes within the architecture:

- **Code Generation:** Responsible for creating the unique code that is valid for a single session.
- **Key Vault:** Handles the secure storage and management of codes.
- **Authentication:** To issue and orchestrate the code generation and verification to authenticate the user.
- **Code distribution:** Ensures the code is delivered through the appropriate medium.

Sample Architecture

The complete architecture for One-Time codes, including the common services, is shown in Figure 3.2. The Code Generation service is involved in the initial phase of authentication, when a new request is made, in distributing the generated code, and in storing the code in the Key Vault to finalize the authentication process. Therefore, it's clear that the Code Generation service is a key element of this architecture. The Code distribution service also plays a significant role, as any delay or disruption in this process could prevent the user from receiving the code. The distribution service can leverage any medium to deliver the generated code and it is the only service that interacts with the user directly.

Moreover, the authentication service is also responsible for issuing the request to the Code Generation service. As there is no Onboarding, the codes are generated each time the Authentication service is triggered. The Key Vault service is a much lighter version of the one present in the previous design. The only real duty is to store the link between the user and the code. Since the validity of the codes is often very small, the Key Vault service can also provide logging to detect replay attacks. As we will discuss in the following sections (3.3, 3.6), this is the easiest and most economical design for passwordless architecture. It includes the least amount of components and most of them are present in the lightest form compared to the other architectures.

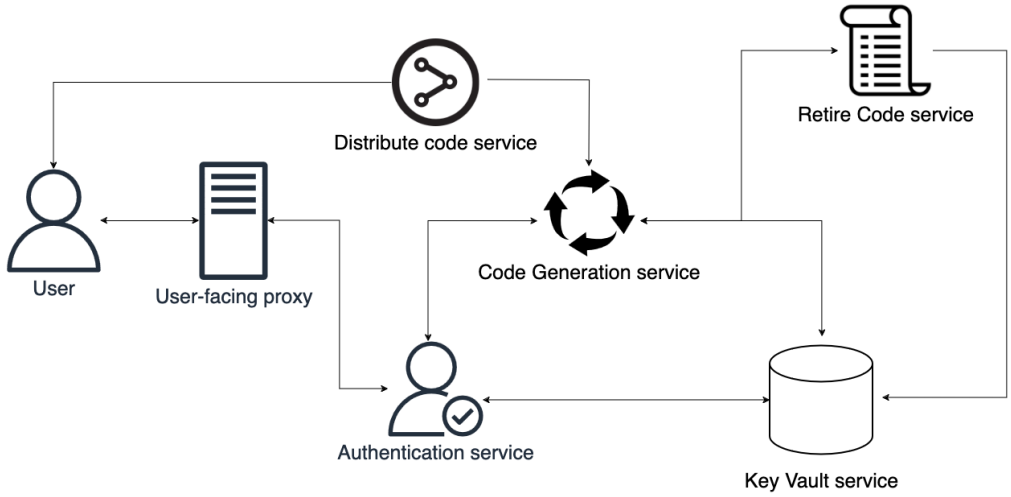

Figure 3.3. Passwordless architecture with Long-term codes.

3.1.3 Long-Term Code

As it may be predictable, Long-Term codes share some components and characteristics with One-Time code architectures. Hence, some of the services may sound familiar, yet, their implementation and interaction changes due to the longer lifespan of codes. On the server side, the following blocks are needed to use Long-Term Codes within the architecture:

- **Code Generation:** Generate the unique link or issue the certificate based on some input properties.
- **Key Vault:** To store certificates or handle links hashes.
- **Authentication:** To perform the verification of the link or certificate validation.
- **Code distribution:** Ensures the code is delivered through the appropriate medium.
- **Retire code:** To allow for certificates to be revoked and blocked from authentication.

Sample Architecture

The complete architecture for Long-Term codes, including the common services, is shown in Figure 3.3. Here the two key services are the Code Generation service and the Key Vault service. Depending on the implementation, the importance of one or the other may vary significantly. In a Token-based authentication, both Generation and Storage are fundamental to verify tokens properly. On the other hand, in the case of a magic link architecture, the Code Generation is dominant.

The complexity of this design sits in between the previous two. From the point of view of the number of nodes, this solution is closer to the first architecture. However, the technical implementation of the services is much more simple. Additionally, depending on the starting architecture, this may be the easiest solution to implement in the event that a Certification Authority is already set up. The services like Retire Code and Code Generation may be already in place if the organization has its own CA.

3.1.4 Common and Characteristic services

So far, it has become clear that every passwordless technology requires a well-defined architecture to operate and be implemented effectively. Without the necessary supporting services, transitioning to passwordless authentication is not feasible. After presenting the three potential designs

for the selected technologies, it is crucial to recognize that not all services carry the same level of importance. Hence, we will use this section as a wrapper for the following discussions. It will be useful to come back to this section when dealing with implementation details (next chapter, 3.2), minimum requirements (3.3) and costs and overhead (3.6). The following table summarizes the services associated with each architecture, categorizing them into common and characteristic services. The architectures are named as follows: **SecKeyBio**, **OneTimeBio**, and **LongTerm**.

Architecture	Common Services	Characteristic Services
SecKeyBio	Key Vault, Authentication	Onboarding, Retire Keys, Backup
OneTimeBio	Key Vault, Authentication	Code Generation, Code Distribution
LongTerm	Key Vault, Authentication	Code Generation, Code Distribution, Retire code

Table 3.1. Architectures and Their Services

Looking at Table 3.1 and at each of the illustrative pictures, the division of services and the complexity of each architecture become evident. The simplest architecture is **OneTimeBio**, followed by **LongTerm**. The most complex is **SecKeyBio**. However, this consideration is a general deduction that we draw from the previous sections. It may not always hold true, as it does not account for factors such as the initial state of the architecture (prior to the transition) and technical aspects like the strong support for FIDO2 Keys (which is part of **SecKeyBio**). The scope of one of the following chapters (3.3) is to properly address those decisions.

3.2 Implementation of the services

As mentioned before, this section is paired with the previous and is aimed at providing an implementation view of the services described above. This section introduces some of the details of the logical components and may be skipped if the reader wishes to keep a higher-level view of the work. Here is a more in-depth analysis of the services that can be used to construct a passwordless authentication system. First, the common services are discussed, followed by the characteristic services. Each of the services will come with mentions of the actual technology stack that can be used to realize the service. The tables are not an exhaustive list of the technologies that can be employed, yet, they try to cover cloud-based solutions, on-premises solutions, and hybrid solutions.

3.2.1 Key Vault service

This component comes in different flavors depending on the implementation chosen. Its logical function remains constant: provide secure storage for secrets involved in the authentication phase. It is the first of the two services that are common to every passwordless environment. The following functions are required:

1. Secure Storage

The primary pillar of the Key Vault Service is to securely store cryptographic keys and secrets. This involves encrypting private keys and other sensitive data at rest, ensuring they are only accessible by authorized entities. The service must implement robust encryption algorithms and key management protocols to maintain the confidentiality and integrity of the stored data.

2. Secret Access Control

Access control is crucial in the Key Vault Service to ensure that only authorized modules or services can access or use the stored keys. The service may implement fine-grained access policies, such as role-based access control (RBAC) or attribute-based access control (ABAC). These policies define who can perform actions such as key retrieval, signing, or decryption.

3. Backup and Recovery

To ensure data availability and resilience, the Key Vault Service includes backup and recovery mechanisms. These ensure that keys and secrets can be restored in the event of data corruption, accidental deletion, or system failures. Backups are encrypted and stored securely, and recovery processes should be extensively tested to ensure they function correctly when needed.

Service Component	Technologies/Strategies	Category
Secure Storage	AWS KMS, Azure Key Vault, GCP KMS	Cloud
	Thales Luna, SafeNet HSM	On-Prem HSM
	HashiCorp Vault, OpenSSL, LUKS	Server
	Encrypted PostgreSQL, TDE, MongoDB	Database
Access Control	AWS IAM, Azure RBAC, GCP IAM	Cloud RBAC
	LDAP, Active Directory	On-Prem RBAC
	Vault Policies, Custom Policies	Policy Engine
Backup & Recovery	AWS Backup, Azure Backup, GCP Storage	Cloud Backup
	Bacula, rsync, Tape Backup	On-Prem Backup
	pg_dump, mongodump	Database Backup

Table 3.2. Technologies for Implementing the Key Vault Service

3.2.2 Authenticaiton service

This component is crucial for the correct implementation of a passwordless authentication system. This is the second building block that is common to all passwordless architecture. It is responsible for handling the authentication process between the user and the system. It may act on different stages of the process, from the first registration to the normal login of a user. It glues the internal structure of the architecture to the user requests and correctly satisfies the following pillars:

1. Evaluation Mechanism

The core of the authentication service is the evaluation mechanism that allows authentication requests to be approved or denied. This involves the verification of the code, response of the challenge, string, or certification parameters.

2. User verification

User verification is a crucial aspect of the Authentication Service, ensuring that the authentication request is being made by the legitimate user. This may involve a query to the directory of users to ensure its presence in the system. The service verifies that the user has successfully completed this step before proceeding with authentication.

Service Component	Technologies/Strategies	Category
Evaluation Mechanism	OAuth 2.0, OpenID Connect	Protocol
	FIDO2, WebAuthn	Standard
	JWT, SAML	Token-Based
	Custom Auth Logic	Custom Implementation
User Verification	LDAP, Active Directory	Directory Service
	Azure AD, AWS Cognito	Cloud Directory

Table 3.3. Technologies for Implementing the Authentication Service

3.2.3 Onboarding service

This functionality is peculiar to security keys, as the first initialization phase is required to register the physical device. This service should take care of acquiring the public key of the hardware token and sending a challenge to prove that the user actually possesses the relative private key. This phase is successful when the server has acquired the necessary key and associated it with the user for the login phase. For the Onboarding service, the following functions are required:

1. Key-Pair setup

The first pillar of the onboarding service is the Key-Pair setup. As hardware tokens heavily rely on asymmetric cryptography, this phase is crucial to generate the pair needed in the authentication phase. The keys are generated on the user side and from the server side, the only duty is to issue this generation and present the user with an interface to register the newly created key.

2. Attestation

This pillar is crucial to verify that the key has been correctly generated by the hardware token. The attestation object that has to be analyzed is constructed as follows:

- **authData** (Authenticator Data)
- **fmt** (Attestation Format)
- **attStmt** (Attestation Statement)

3. Registration request

This last pillar is responsible for generating the correct registration request that will then trigger the client-side interactions with the hardware token and with the browser. The request should contain the following fields:

- **rp** (Relying Party Information)
- **user** (User Information)
- **challenge**
- **attestation**
- **excludeCredentials** (To prevent double registration)

Service Component	Technologies/Strategies	Category
Key-Pair Setup	FIDO2, WebAuthn	Protocol
	YubiKey, Feitian Token	Hardware Token
	OpenSSL, GnuPG	Cryptographic Tools
	Custom Registration UI	User Interface
Attestation	TPM, Secure Enclave	Hardware Security
	FIDO2, U2F	Standard
	Custom Attestation Logic	Custom Implementation
Registration Request	WebAuthn API, FIDO2 API	API Integration
	Custom Backend Service	Custom Implementation
	HTTPS, TLS	Secure Communication

Table 3.4. Technologies for Implementing the Onboarding Service

3.2.4 Retire Keys/Codes service

This functionality is designed to manage situations where security keys need to be updated, such as when a key is lost, broken, or compromised. In these cases, it's important to disconnect the user from the affected hardware. The system also needs to handle situations where providers are switched or vulnerabilities are found in the hardware, leading to certain keys being blocked. A dedicated component manages these tasks and integrates with other parts of the system to ensure authentication continues smoothly, even when there are issues with security keys. This service is especially useful when using Tokens and Certificates. Similar to a Certificate Revocation List, it allows tokens and security links to be invalidated before they expire, particularly in response to breaches or tampering. This is crucial for protecting the system from potential attacks, like those that could occur if the Code Generation service is compromised. For this service, the following pillars are required:

1. Key/Code Deactivation

The first function of the Retire Keys/Codes Service is the deactivation of keys/codes that are ready for retirement. This involves marking the key/code inactive in the system, ensuring that it can no longer be used for authentication or any cryptographic operations. The service must also update the associated access control policies to reflect this change, preventing any further usage of the deactivated key/code.

2. Secure Key/Code Deletion

Following deactivation, the Retire Keys/Code Service securely deletes the key/code from all storage locations. This process must ensure that the element is irrecoverable, involving methods such as cryptographic erasure or overwriting the data multiple times. The service must also ensure that any backups or replicas are similarly deleted, maintaining the confidentiality of the retired key/code.

Service Component	Technologies/Strategies	Category
Key/Code Deactivation	CRL, OCSP	Revocation Protocols
	AWS KMS, Azure Key Vault	Cloud Key Management
	LDAP, Active Directory	Directory Service
	Custom Revocation Service	Custom Implementation
Secure Key/Code Deletion	Shredding, Cryptographic Erasure	Secure Deletion Methods
	AWS KMS, Azure Key Vault	Cloud Key Management
	LUKS, BitLocker	Disk Encryption
	Secure Wipe Tools (srm, shred)	Software Tools

Table 3.5. Technologies for Implementing the Retire Keys/Codes Service

3.2.5 Backup service

This component needs to be considered only for Security Keys that allow exportable secrets. Some Security Keys, like those compliant with FIDO2 standards, do not support recoverability and therefore do not require a backup service. However, other standards, such as Passkey, offer users backup and restore functionality. When implementing such features, the security properties of the backup must be even stronger than those of the Key Vault. This is because the Backup service must enable the user to recover their full identity and transfer it to another token securely. The following are the key pillars:

1. Key restoration and portability

The Backup Service facilitates the secure restoration of hardware token keys on new or restored devices. This process involves decrypting the backed-up keys and importing them onto the new hardware token or into the new architecture.

2. Cross-Device synchronization

This feature enables users to securely sync their hardware token keys across multiple authorized devices, ensuring consistent authentication experiences across all platforms. The service manages the secure distribution of the keys to the other services in the architecture.

Service Component	Technologies/Strategies	Category
Key Restoration	Passkey, PGP	Backup Standards
	AWS S3, Azure Blob Storage	Cloud Storage
	LUKS, BitLocker	Disk Encryption
Cross-Device Sync	Secure Transfer Protocols (SFTP, TLS)	Secure Transfer
	AWS KMS, Azure Key Vault	Key Management
	Custom Sync Service	Custom Implementation

Table 3.6. Technologies for Implementing the Backup Service

3.2.6 Code Generation service

This component is responsible for generating the random codes used during authentication. The codes are unique to each user, valid for a single use, and only for a limited time. The Authentication Service is in charge of verifying the code, while this service focuses on generating an unpredictable and secure code. The fundamental pillars are:

1. Secure Code Generation

The primary function of the Code Generation Service is to generate secure codes that can be used for authentication. This typically involves creating time-based or event-based codes, property-based links, or newly generated certificates. The main details for generation are reported in industry standards such as RFC 6238 and RFC 4226. The service must use cryptographic algorithms to ensure that the codes are unpredictable and resistant to attacks.

2. Audit and Monitoring

The Code Generation Service maintains logs of all code generation and delivery activities, enabling audit and monitoring of the service. These logs provide a record of when and where codes were generated, who initiated the generation, and whether the codes were successfully used in an authentication attempt. This information is crucial for security audits, compliance checks, and forensic investigations in case of security incidents.

3.2.7 Code Distribution service

This function is crucial for delivering the code generated by the previous service. As discussed earlier (2.4.2), there are various mediums for sending the code to the user for authentication. The distribution service should consider that users may have multiple delivery channels, and if time is a critical factor, it should use the fastest option available. This service is a bridge between the internal structure that generates the code and the end user that receives it. The following are the key pillars:

Service Component	Technologies/Strategies	Category
Secure Code Generation	TOTP (RFC 6238), HOTP (RFC 4226)	Industry Standards
	OpenSSL, PyOTP	Cryptographic Libraries
	HSM, TPM	Hardware Security
	AWS Secrets Manager, Azure Key Vault	Cloud Key Management
Audit and Monitoring	Splunk, ELK Stack	Log Management
	AWS CloudWatch, Azure Monitor	Cloud Monitoring
	Syslog, Fluentd	Logging Tools
	Custom Audit Service	Custom Implementation

Table 3.7. Technologies for Implementing the Code Generation Service

1. Real-Time delivery

Timeliness is crucial in the Code Distribution Service, as authentication codes typically have a short validity period. The service is designed to deliver codes in real-time, minimizing delays that could prevent users from completing the authentication process. The service must monitor delivery performance and optimize routing to ensure that codes reach users as quickly as possible.

2. Integration with delivery systems

The second pillar is the integration with multiple delivery systems at once. The code distribution service should integrate with communication mechanisms such as mail exchange service, SMS, and notification service.

Service Component	Technologies/Strategies	Category
Real-Time Delivery	WebSockets, HTTP/2	Real-Time Communication
	Firebase Cloud Messaging (FCM)	Push Notifications
	SMS Gateways (Twilio, Nexmo)	SMS Delivery
	Email Services (SendGrid, Amazon SES)	Email Delivery
Delivery Systems	API Integration	General Integration
	Message Queues (RabbitMQ, Kafka)	Messaging Systems
	Email Servers (Postfix, Exchange)	Email Servers
	SMS Aggregators (Plivo, Bandwidth)	SMS Aggregators

Table 3.8. Technologies for Implementing the Code Distribution Service

3.3 Passwordless Fit and Minimum Requirements

After detailing the key components needed for a passwordless authentication system, it is crucial to understand how our architecture is positioned and what are the minimum requirements to start the transition. So far, we have detailed the architecture, behaviors, and interactions of the different construction bricks in three state-of-the-art configurations. This section acts as a practical checklist, similar to pre-flight checks, aimed at evaluating how our current architecture measures up against the necessary conditions for transitioning to passwordless authentication.

From a conceptual point of view, the minimum requirements have already been stated in the previous section (3.1). It is sufficient to have a look at the services that need to be implemented for one of the three architectures. Those services are the minimum requirements to go passwordless with one of the solutions described above. While the three presented architectures cover a wide range of scenarios, they can be adjusted for simpler designs if needed. The key here is to assess whether our current system is ready for passwordless authentication or if adjustments are necessary to meet these requirements. We will use the previous knowledge to define what is missing and what needs to be modified, in order to plan for the transition.

3.3.1 Checklist for the transition

As organizations can begin the transition to passwordless authentication at various stages in their security journey, it is fundamental to understand the current state of the architecture. First, we will discuss the 5 steps checklist to go through before proceeding with the next steps.

1. Evaluate Current Infrastructure

The infrastructure must be ready to adapt to the new authentication method and to add or modify elements in order to create the necessary services. The key questions to be answered are:

- Am I creating a new infrastructure?
- Can I perform changes to the current infrastructure (network, server, backups, etc.)?
- Can I add elements to the current infrastructure?
- Does our current infrastructure include legacy systems?
- Does our infrastructure already leverage some type of passwordless authentication?

2. Evaluate the User Directory

The user directory is the core element of every authentication system. As it holds the bond between classic credentials and user information, it is fundamental to guarantee strict compatibility with the new passwordless architecture. The key questions to be answered are:

- What is the current authentication method?
- Is MFA already in place?
- What information about the user is present in the directory?
- Do we manage the user identity "in-house"?
- Does the directory issue challenge / generate code to be sent to the user?
- Do we have a CA infrastructure?
- Where are the secrets stored?

3. Evaluate Users and Devices

Although it may not seem a dealbreaker, end users will have to adopt passwordless authentication and they have to be taken into account when evaluating the current level of our system. As a perfect migration may fail due to a low level of adoption on the user side, it is important to take into account the Users and their Devices. The key questions to be answered are:

- What is the current level of awareness towards passwordless authentication?
- What is the typical persona that will use the solution?
- Do any of the users or groups of users already leverage passwordless authentication?
- Can I add new devices (i.e., security keys or phones) to the current pool of a user?
- Do users have limitations regarding login systems?
- Do their current devices support any of the passwordless authentication methods?
- How diverse are the devices in terms of operating systems?

4. Evaluate Security Policies and Compliance

When planning a pivotal change, such as moving from password-based authentication to passwordless authentication, compliance and policies may limit or prevent some choices. In order to ensure compliance, it is necessary that the new elements and design adhere to the company policy. The key questions to be answered are:

- Is there any company policy limiting the authentication types?
- Is MFA strongly enforced?
- Do I need to adhere to specific industry standards (i.e., financial sector, medical sector, etc.)?
- How do our policies address the management and protection of cryptographic keys?
- Does our current logging and auditing infrastructure provide sufficient coverage?

5. Evaluate Platform Compatibility

The last group of questions are referred to the platform itself. This aspect deals with all the systems already in place such as external providers or services that directly integrate with our infrastructure. The key questions to be answered are:

- Do we work with on-premise, hybrid, or cloud-only environments?
- Do we have a multi-cloud integration?
- Is there any legacy system integrated with our infrastructure?
- Do we have custom integration with external providers (i.e., identity providers, intelligence providers)?
- Do we have monitoring systems or agents to be configured in the new environment?

3.3.2 Target for the transition

Once the key questions have been addressed, we should have a clear picture of the current state of our architecture. With this understanding, we can now map the minimum requirements discussed in the previous chapter to our existing setup. The goal is to judge how far we are from meeting these minimum requirements and to identify whether we already meet some or all of them. The insights gained from these questions will help us picture a course for the passwordless transition. As previously mentioned, the transition can start from any level of maturity or state of infrastructure, but it may not always be clear which passwordless architecture to aim for. The answers to our questions should clarify our current position and guide us in bridging the gap between our current design and the requirements.

If a specific passwordless architecture has been determined in advance, the next steps become more straightforward: we need to implement the necessary requirements or components to align our system with the target architecture. For instance, this might involve setting up a Code Generation Service or integrating our existing Key Vault with a Retire Code service. However, if the target architecture isn't well-defined, we need to determine which of the available architectures is closest to our current setup. This "closeness" could be measured in terms of infrastructure components (such as software, servers, or Active Directory) or by evaluating the cost and effort required to make the necessary changes. Additionally, we should consider whether any passwordless authentication methods meet our policy requirements and assess the investment needed to achieve the desired security level. Deciding on the best approach will ultimately fall to the architects overseeing the transition.

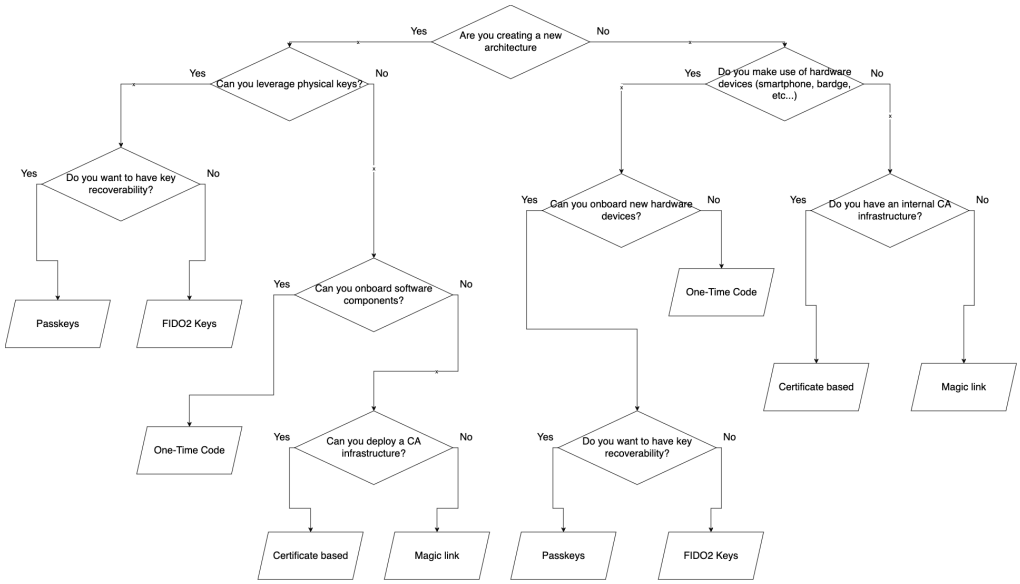

Figure 3.4. Flowchart for passowordless architecture target.

The questions answered above allow us to easily go through a flowchart of decisions. Figure 3.4 is an example of a diagram that leverages part of the answers to give a practical architectural target. The main objective is to minimize architectural changes, hence, costs. Similar diagrams may be created with a different focus in mind. To see a more extensive list of flowcharts, refer to the Github repository of the project at the following link: Passwordless transition PoC - GitHub repository

As mentioned above, this flowchart is more useful in case of doubt about the target architecture. If the desired passwordless technology has been already chosen, the diagram offers little benefit. A practical example may be the following:

Flowchart walkthrough

We have previously answered the checklist and got a basic understanding of the current state of architecture. Starting the flowchart we know that the architecture is already in place **(Question Groups 1 and 2)**. We know that our users already have devices in their personal pool (smartphones, tablets, PC, etc.) **(Question Group 3)**. Hence, we also know if we can add a new piece of hardware and more to either Passkeys or FIDO2 Keys **(Question Group 3)**. Ultimately, we fall into the FIDO 2 Keys as company policy has to adhere to financial service regulations and secrets cannot leave the devices **(Question Group 5)**.

With this example, it should be clear that the questions are directly used to navigate the flowchart and that they are needed to correctly select the target passwordless technology.

3.4 Strategy for the transition

Defining a clear target for the passwordless transition, grounded in the minimum requirements and current architecture, is essential for the project's success. This step is critical as it lays the foundation for an organized and trackable project. It's important to recognize that the insights from the previous chapter (understanding the services, assessing the architecture, and defining a target), are part of a broader objective: achieving full passwordless adoption across the organization.

Specifically, every deployment may follow a series of steps that lead to the desired passwordless coverage. A further development that we treat in this first contribution is the definition of a

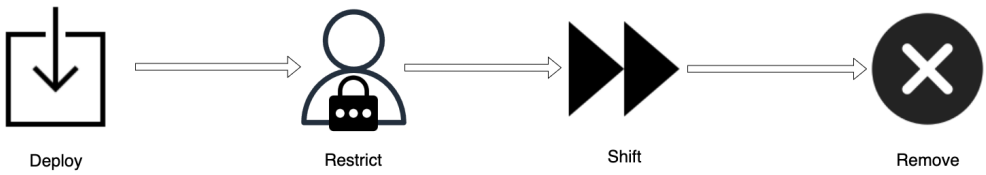

Figure 3.5. Strategy for passowordless transition.

"Strategy" to correctly execute the passwordless transition. As it needs to accommodate room for the different technologies, the plan focuses on the steps to be taken and to be considered when planning the time and coverage of the deployment.

By combining the target definition from the previous section with the roadmap outlined here, it becomes possible to plan strategically, present a clear path to stakeholders, and allocate resources effectively. This approach provides a holistic view of the transition, making sure that all aspects are covered from the first deployment of a passwordless authentication method to the removal of passwords across the organization.

3.4.1 Roadmap definition

To help in visually identifying the elements of the roadmap, Figure 3.5 is provided. This straightforward map is composed of four main steps. With the letter "M", we indicate "Milestone".

- **M1: Deploy a passwordless architecture.**
- **M2: Restric the use of passwords.**
- **M3: Shift from password-based to passwordless authentication.**
- **M4: Remove passwords from the system.**

Each of these steps represents a strategic milestone on the path to achieving password freedom within the organization. The sequence is deliberate, drawing inspiration from general technology-adoption-lifecycle strategies. In such strategies, a new component is first developed and integrated, followed by a gentle push to initiate the shift, finally leading to the complete adoption of the new component and the removal of the old one. Having a look at the picture, the comparison becomes clear.

It's easy to see that the roadmap can be simplified depending on the current state of the architecture. This is where the earlier framework of questions and the flowchart come in handy. For instance, if the existing system already uses magic links, and the goal is to make minimal changes, the flowchart might suggest moving to a One-Time Code system as the target architecture. In this case, following the strategy would mean skipping milestone **M2** (Restrict), since the use of passwords has already been limited with magic links. As mentioned before, the plan is designed to cover a general scenario but should be adjusted to fit the specific needs of the project.

3.4.2 Step-by-Step analysis

Now that the outline of the plan is clear, it is worth exploring the different stages. This section, along with the next ones, will be of paramount importance when re-imagining our current authentication strategy to fit passwordless authentication. Hereafter are the details of each of the milestones proposed in the plan.

M1: Deploy a passwordless architecture

The first step is to establish a robust passwordless system within the organization. This involves implementing the right architecture to fit the needs of the chosen technology. Here, all the work developed so far is put in place. The definition of the architecture, the identification of the requirements, and the development of a flowchart are used during this phase. The goal of this phase is to deploy a reliable architecture that is integrated with existing infrastructure. This phase may include testing against functional requirements to ensure compliance. This milestone is achieved once the implementation is completed. As a reference, this phase may also include the **Pilot Phase** and the **Metrics Phase**, in which the demo of the project is deployed and the metrics to be taken into account are evaluated.

M2: Restric the user of passwords

With a passwordless architecture in place, the next step is to reduce the reliance on passwords across the organization. This involves gradually phasing out password prompts in favor of the newly deployed authentication methods. By doing so, users are encouraged to rely more on passwordless options, making passwords increasingly obsolete. This phase also includes updating workflows and applications to eliminate the need for passwords, ensuring that the password surface area within the organization is minimized. The objective is to create an environment where users know they have passwords but rarely, if ever, need to use them, thereby reducing the risks associated with password-based authentication, such as phishing. As a reference, this phase may also include the **User Training** and the **Communication Plan**, in which the users (unaware of security standards) receive training and the communication of the new adoption is carried on.

M3: Shift from password-based to passwordless authentication

Once passwords are no longer a visible part of the user experience, the organization can begin the full transition to a passwordless state. At this stage, users stop using passwords altogether. Authentication is handled entirely through the new passwordless methods that have been deployed. This shift requires complete user education and support to ensure a smooth transition. It also involves making necessary changes to user accounts and access policies to enforce passwordless authentication, ensuring that all systems and applications are fully compatible with the new methods. The focus here is on making passwordless authentication the default and only method of access across the organization. As a reference, this phase may also include the **Compatibility Check** and the **Incident Response Plan**, in which the compatibility of the newly integrated system is checked and a plan for disaster recovery is ready to be used.

M4: Remove passwords from the system

The final step in the transition is to completely remove passwords from the organization's identity directories. At this point, passwords are no longer stored or used in any form. This means that even if a user were to try, they wouldn't be able to log in with a password because it simply doesn't exist in the system. This step is the final commitment of the organization to a passwordless environment. It also involves continuous monitoring and updates to maintain the integrity of the passwordless system. As a reference, this phase may also include the **Backup Plan** and the **Monitoring phase**, in which the backups or fallback architecture is kept in case of rollback and a monitoring structure is used to ensure that the system is functioning.

3.4.3 Need for a plan

This section offers a preview of what will be covered in Chapter 4. The second major contribution of this work involves the analysis of a case study, including the creation and deployment of a Proof of Concept. This initial mention underscores the importance of a strategy and roadmap for the passwordless transition. Chapter 4 will dive into a case study focused on a Microsoft client,

referred to as "GripGotham" for privacy reasons. In the first attempt to deploy a passwordless authentication method, the architect underestimated the complexity of implementing FIDO2 Keys in a large-scale environment, assuming it would be as straightforward as in a standalone setup. Testing FIDO2 Keys on a personal device limits visibility into the necessary components. This makes the identification of required elements challenging for the architects who are initiating the change. As previously discussed in the Background chapter (2.4.2), the authentication architecture is often transparent to the user, complicating the identification of essential components. Without a clear plan, this deployment failed at the initial stage (**M1**), highlighting a lack of a defined target and no checklist prior to deployment. These issues will be extensively explored in a dedicated section.

3.5 Expected increase in the security posture

A fundamental aspect to be considered when implementing passwordless authentication is the increase in the security posture. Security posture refers to the overall state of an organization's security measures and practices. It includes how well its systems, processes, and policies are designed and implemented to protect against threats and vulnerabilities. Essentially, it's a snapshot of how secure an organization is at any given time, based on its current defenses, risk management strategies, and response capabilities. A strong security posture means the organization has effective controls in place to prevent, detect, and respond to security incidents, while a weak security posture indicates that improvements are needed to better safeguard against potential risks.

This section is paired with the following one (3.6) in which we analyze the transition from the opposite point of view: Risks introduced by altering the architecture and migrating to passwordless technology. To evaluate the increase in security posture there are plenty of frameworks or practical aspects that can help us. The first discussion will deal with security frameworks as a formal assessment while the second will map the formal improvement to more practical benefits.

3.5.1 Evaluation with frameworks

To evaluate the expected increase, a useful approach is to apply a well-established security framework, like the NIST Cybersecurity Framework [36], CIS Controls [37] or CISA Framework [38]. These frameworks give a structured way to assess improvements by providing specific criteria and benchmarks for different security domains. However, not all the frameworks include a specific section for passwordless technologies. This is due to the fact that some of the standards may be lagging behind with respect to the latest implementation. Nevertheless, all passwordless authentication is framed as the phishing-resistant authentication method.

Passwordless authentication contributes to various pillars of the following frameworks, not only in the identity domain but also in the access management and incident pillars.

NIST Cybersecurity Framework

Hereafter are the interesting security pillars of the NIST Cybersecurity Framework:

- **Identify** - Passwordless systems enhance the inventory of users and devices by linking identities to stronger authentication methods.
- **Protect** - Passwordless authentication directly improves this category by replacing weak password-based controls with stronger authentication methods.

CIS Controls

Hereafter are the interested security pillars of the CIS Controls:

- **Control 5** - With passwordless authentication, the process of provisioning and de-provisioning accounts becomes more secure. The elimination of passwords reduces the risk associated with account creation and removal.
- **Control 6** - Replacing passwords with stronger authentication methods directly impacts account security by reducing the likelihood of account compromise through stolen or weak passwords. This control is strengthened as passwordless methods make it harder for attackers to gain unauthorized access.

CISA Framework

Hereafter are the interesting security pillars of the CISA Framework:

- **Strengthening Identity and Access Management** - Passwordless authentication is a key component in enhancing IAM by reducing the risks associated with password reuse, phishing, and brute force attacks.
- **Reducing Risk from Phishing and Credential Theft** - Passwordless authentication, especially with methods like FIDO2 keys or biometrics, eliminates the vector of phishing attacks that target passwords, significantly reducing this risk.

3.5.2 Practical improvements

The formal frameworks for evaluating the impact of deploying passwordless authentication are particularly useful when dealing with other security professionals or when checking against the compliance of the organization. They must be present as a formal reference in every document but they leave little room for their interpretation. The following are some of the practical examples of the increase in the security posture of an organization.

Reduction in Attack Surface

The shift to passwordless authentication directly reduces the attack surface by eliminating common vulnerabilities associated with password-based systems. Passwordless technologies mitigate these risks by nullifying the most common attack vectors.

- **Phishing and Social Engineering Mitigation**: With the elimination of passwords, attackers can no longer employ phishing tactics to steal credentials. The reliance on biometrics or hardware tokens makes it significantly harder for attackers to impersonate legitimate users, as these authentication factors are not easily transferable or replaceable.
- **Brute Force and Credential Stuffing Prevention**: Since there are no passwords to guess or reuse, brute-force attacks and credential stuffing become ineffective. The security posture is enhanced as attackers must now contend with far more complex and secure authentication methods, such as biometric data or cryptographic keys.

Improved User Behavior and Compliance

Passwordless technologies not only enhance security but also improve user behavior by reducing the cognitive load associated with managing passwords. Users are no longer required to remember complex passwords or frequently update them, which often leads to poor security practices such as password reuse or storing passwords in insecure locations.

- **Reduction in Human Error**: By eliminating the need for passwords, the potential for human error is significantly reduced. Users are less likely to fall victim to phishing or accidentally expose their credentials, leading to a more secure overall environment.

- **Better Compliance with Security Policies**: Passwordless authentication simplifies the enforcement of security policies. Users are more likely to comply with security requirements when the process is seamless and does not involve cumbersome password management practices.

Alignment with Zero Trust Architecture

As mentioned in section 2.4.1, passwordless authentication aligns closely with the principles of Zero Trust architecture, which emphasizes continuous verification of user identity, device security, and network integrity. The adoption of passwordless technologies enables organizations to implement a zero-trust model more effectively

- **Continuous Authentication**: In a Zero Trust environment, passwordless authentication methods can be used for continuous verification, ensuring that access rights are maintained only as long as the user and device remain authenticated and secure.
- **Granular Access Control**: Passwordless systems allow for more granular access control, as authentication can be tied to specific devices and contexts, reducing the risk of unauthorized access even further.

3.6 Costs and overhead

As every technological evolution, the transition towards passwordless authentication does not come with costs. Implementing a new authentication method and all the components to support it may be costly for already established architecture. As seen in one of the previous chapter (3.3), we can construct a flowchart with any parameter in mind, such as the costs. Intuition may suggest that for an already established architecture costs associated with the transition are higher. However, thinking about the case of an already established CA, this is not true. Moreover, the monetary cost is not the only one to be considered. Time costs, workload, and operational overhead have to be taken into account.

3.6.1 Monetary Costs

Monetary costs are the most straightforward and often the most immediately impactful consideration when transitioning to a passwordless architecture. These costs include the purchase of hardware, licensing fees for software, and potential increases in infrastructure requirements.

Security Key + Biometrics

This architecture generally incurs the highest upfront costs. The purchase of hardware security keys, especially in a large organization, can be significant. For instance, FIDO2-compliant keys are widely used but can range from $20 to $50 per key. Additionally, the already-in-place hardware may require some update or adaptation to the keys. This architecture also demands robust backup systems and an onboarding service, further increasing costs.

One-Time Code + Biometrics

This architecture has lower upfront costs compared to Security Key + Biometrics. The primary costs are associated with the infrastructure needed to generate and distribute one-time codes securely. Services such as SMS aggregators or email servers must be reliable and secure, and this might require additional investments if the current infrastructure is inadequate. However, the need for biometrics adds a layer of cost similar to that in the Security Key architecture, though potentially less extensive depending on the implementation.

Long-Term Code Architecture

The Long-Term Code architecture typically has low monetary costs compared to Security Keys. This architecture can leverage existing infrastructure, such as a Certificate Authority (CA), for code generation and storage. If a CA is already in place, the additional costs may be minimal. The architecture does require a Key Vault service, but this can be implemented with relatively low overhead if existing systems are adapted.

3.6.2 Time Costs

Time costs are another critical factor, particularly in the implementation phase. These include the time required for planning, deployment, user training, and the transition process.

Security Key + Biometrics

The implementation of this architecture is the most time-consuming. Extensive planning is required to ensure compatibility across the organization, especially when deploying physical security keys and integrating them with biometric systems. The onboarding process is also complex, requiring significant time to enroll users into the new system, distribute keys, and provide necessary training. Moreover, the time required for ongoing management, such as key revocation and replacement, adds to the operational overhead.

One-Time Code + Biometrics

This architecture demands significant time for initial setup, particularly in configuring the code generation and distribution systems. However, once established, the ongoing time costs are generally lower than those for Security Key architecture. The user training process is streamlined since many users may already be familiar with one-time codes from existing two-factor authentication systems, reducing the time needed for adoption.

Long-Term Code

The Long-Term Code architecture offers similar Time costs compared to One-Time code. Nevertheless, in case the CA is already in place it becomes an efficient implementation in terms of time. If the organization already has a CA or similar infrastructure, the transition can be relatively quick. However, ensuring that all systems are compatible with long-term codes can still require a detailed and time-consuming assessment phase.

3.6.3 Workload and Operational Overhead

The workload and operational overhead associated with each architecture include the ongoing management, support requirements, and potential disruptions to normal operations during the transition period.

Security Key + Biometrics

This architecture introduces significant operational overhead. The management of physical security keys, including their issuance, replacement, and retirement, requires continuous administrative support. Additionally, the integration of biometrics may necessitate ongoing maintenance and support, particularly if hardware issues arise. The operational workload can also increase due to the complexity of managing backups and ensuring continuous availability.

One-Time Code + Biometrics

The operational overhead for this architecture is moderate. While the ongoing management of code generation and distribution systems requires attention, these tasks can often be automated to some extent, reducing manual intervention. The biometric component still adds some overhead, particularly in environments where user devices are not uniformly equipped with biometric capabilities. However, this architecture generally demands less continuous support than the Security Key architecture and Long-Term Code.

Long-Term Code

The operational overhead for this architecture is the lowest among the three. Once implemented, the system requires minimal intervention, especially if integrated with an existing CA. The primary workload involves ensuring the ongoing validity of long-term codes and managing their revocation when necessary. As a result, this architecture is particularly appealing for organizations with limited IT resources or those looking to minimize ongoing support requirements

3.6.4 Comparative Analysis

Each of the three designs has its own strengths and capabilities that have been extensively discussed in the previous chapters. Those characteristics are associated with the costs of their deployment. After having broken down the sources of costs for the three designs the following table provides a comparison among them. The architectures are named as follows: **SecKeyBio**, **OneTimeBio**, and **LongTerm**.

Table 3.9. Comparison of Costs and Overhead for Passwordless Architectures

Architecture	Monetary	Time	Operational	Best Use Case
SecKeyBio	High	High	High	High-security environments
OneTimeBio	Moderate	Low	Moderate	Medium to large organizations
LongTerm	Low	Moderate	Moderate	Organizations with existing CAs

3.7 Risks introduced

Risks are a significant concern in security designs and technologies, and passwordless systems are no exception. Just like in the discussion about Costs and Overheads, risks are inherent in the design of a passwordless system. Understanding and mitigating these risks is essential to fully harness the potential of passwordless technologies. Nevertheless, after having read this work and its references, the hope for passwordless technologies should overcome the associated risks and costs.

Just like costs, each architecture comes with its own risks and sources of attacks that can be broken down by looking at the figures describing the architectures in chapter 3.1.

3.7.1 Highly sensitive services

With the diagrams, we can identify which are the core services and link them to the risks they introduce in the architecture. As previously stated, each element of the architecture is a source of risk, yet, for the sake of brevity, we will limit the discussion to the fundamental elements of each of the three architectures. As a reference, always keep Figure 3.1, 3.2 and 3.3 at reach.

Security Keys + Biometrics

For the first architecture, the node with the highest number of connections is the Key Vault service. The two main attacks are **Unauthorized Access to the Key Vault** and **Insider Threats**.

For **Unauthorized Access to the Key Vault** we have the following:

Attack Vector
Exploiting weak access controls or unpatched vulnerabilities in the Key Vault software.

Attack Flow
The attacker identifies an unpatched vulnerability or misconfiguration in Key Vault Service. The attacker gains unauthorized access to the Key Vault. The attacker disrupts the bond between users and keys.

For **Insider Threats** we have the following:

Attack Vector
Malicious insiders with legitimate access to the Key Vault.

Attack Flow
A malicious insider uses their access to retrieve sensitive cryptographic keys from the Key Vault. The insider uses the keys for unauthorized actions or exfiltrates them to an external party.

One-Time Code + Biometrics

Regarding the second architecture, the highly connected node is the Code Generation Service. The two main types of attacks are **Service Compromise** and **Code Tampering**.

For **Service Compromise** we have the following:

Attack Vector
Exploiting a vulnerability in the code distribution infrastructure.

Attack Flow
The attacker identifies a vulnerability in the code distribution service (e.g., an unpatched server). The attacker exploits the vulnerability to take control of the service. The attacker reroutes or intercepts all outgoing codes, enabling unauthorized access.

For **Code Tampering** we have the following:

Attack Vector
Tampering with the code before it reaches the user.

Attack Flow
The attacker gains access to the code distribution service and alters the outgoing one-time codes. Users receive tampered codes which lead to denial of service.

Long-Term code

Regarding the third architecture, the highly connected node is again the Code Generation Service. However, a distinction can be made between Magic Links and token-based architectures. For magic links, the attacks are similar to the one described above (One-Time code architecture). On the other hand, for Token-based architecture, the attacks and risks are associated with the presence of the CA. The main type of attack is **CA Exploitation**.

For **CA Exploitation** we have the following:

Attack Vector
Exploiting a software vulnerability or misconfiguration in the CA.

Attack Flow
The attacker discovers a flaw in the CA flow. The attacker exploits the flaw to gain administrative control over the CA. The attacker issues or revokes certificates at will, compromising the trust model.

3.7.2 Complementary risks

With the previous sections, the main risks associated to each of the design have been detailed. They have been identified as key risks to be aware of and act to mitigate. Among the plethora of risks, there are some that are somehow independent of the design. Therefore we may have limited control over them and mitigation may not be always possible. Those risks are referred to as "Complementary" and are detailed in this section.

The two main complementary risks are: risks associated to Hardware keys and risks associated with Code Distribution service.

Risks Associated with Hardware Keys

Security keys, such as those used in the Security Key + Biometrics architecture, introduce a significant risk if the physical key is lost or stolen. Unlike passwords, which can be changed relatively easily, a lost security key poses immediate risks. An attacker gaining possession of the key could potentially use it to authenticate as the legitimate user, especially if the biometric component is not strictly enforced or if the biometric data is compromised. The risk is higher in environments where the recovery and deactivation mechanisms for lost or stolen keys are not robust.

Moreover, Security keys can be subject to various physical and side-channel attacks, especially in cases where the key is temporarily left unattended or in environments where an attacker might gain physical access to the key. Attacks such as differential power analysis or electromagnetic analysis could be used to extract cryptographic secrets from the device. Additionally, if the key's firmware is not adequately protected or is updatable without proper security checks, an attacker could potentially inject malicious firmware to compromise the key's security. It is essential to deploy hardware keys with tamper-evident and tamper-resistant features and to use keys that comply with the latest standards, such as FIDO2, which offer protection against such attacks.

Risks Associated with Code Distribution Services

For the One-Time Code + Biometrics and Long-Term Code architectures, the distribution of authentication codes via SMS, email, or push notifications poses inherent risks. Interception of these codes during transmission could allow attackers to authenticate themselves fraudulently. SMS, in particular, is susceptible to SIM-swapping attacks where the attacker takes control of the victim's phone number. Email-based codes can be intercepted if the user's email account is compromised, and even push notifications could be susceptible to attacks if the attacker has managed to compromise the user's device or notification system.

Additionally, codes used in these architectures are often vulnerable to replay attacks, where an attacker intercepts a valid code and uses it within its valid timeframe. Furthermore, if there are delays or synchronization issues between the code generation and the server's acceptance of the code, expired codes could be inadvertently accepted, providing a window of opportunity for attackers. The risks associated with these factors are higher in environments where network latency is a significant concern or where the code expiration policies are extremely stringent.

Chapter 4

Proof of Concept

As previously introduced, this thesis includes a Proof of Concept (PoC) developed during an internship at Microsoft. This chapter marks the second major contribution of the thesis: a real-world implementation of the concepts and strategies discussed in earlier chapters. This allows us to put the methodology into practice and to identify weaknesses in the original design. The Background and Related Works, as well as the Method chapters, serve as essential groundwork to comprehend the sections that follow fully.

The structure of this chapter mirrors that of the previous ones: the technical intensity grows as we proceed in the discussion. This allows readers to understand the high-level details of the project from the initial sections, while a more technical chapter treats the practical aspects of the PoC. The first three sections directly build upon the methodology discussed in the Method chapter, covering the project's requirements, the target architecture, and the decision-making process illustrated by the flowchart. The chapter does not follow a chronological order, this is because dealing with real-life scenarios may imply delays that promote the development of future parts and vice versa. For a more coherent discussion, the chapter follows what we believe to be the best logical order to grasp every detail of the Proof of Concept.

It is also important to note that the entire Proof of Concept can be reproduced using the associated GitHub repository: Passwordless transition PoC - GitHub repository.

Disclaimer

All data provided in this section is for demonstration purposes only and is not connected to the specific Microsoft client who initiated the project. For the sake of confidentiality, this client will be referred to by the alias "GripGotham". The GitHub repository does not link to any meaningful data and does not describe the original architecture and its implementation. Finally, all IP addresses, User IDs, Service IDs, etc, are meaningless.

4.1 Background for the PoC

The focus of the project is to create an architecture that supports the passwordless transition seamlessly, without disrupting the current state of the infrastructure. This shift is expected to enhance the overall security posture, though there are limitations in terms of technology and cost. Being tied to a real-world scenario brings both benefits and challenges, which will be explored further in the chapter. Some decisions are constrained by the target company's requirements, but it is crucial to test the framework's impact and adaptability in a concrete setting.

Out of the company's numerous manufacturing lines, the scope of this project will focus on a single line located in Spain. While there is potential to expand and replicate the implementation in other locations, a single plant has been chosen to avoid unnecessary complexity. The users

of this specific manufacturing line will be the first to experience the transition and will serve as testers and references for future scenarios.

In the manufacturing process, workers report data to the business through tablets and shared devices. Specifically, for a baseline of 5,000 users, there are 1,000 tablets available. It's worth noting that these tablets are not equally distributed. The service desk assigns one tablet per employee, while blue-collar workers share one tablet among an average of 20 users. This detail will become relevant later.

Currently, users authenticate on the tablets with a basic username and password, with no multi-factor authentication (MFA) in place. This applies to both Windows 10 and internal application logins. These internal applications range from SAP for Business to the Microsoft 365 suite. Employees are not permitted to use company phones, and for ethical and personal reasons, they do not use personal devices for work-related tasks. The need for change was identified during a brief assessment of one of the manufacturing lines in the Spanish sub-unit. Due to a complex and lengthy password policy, users tend to write down their passwords on sticky notes attached to the benches or tablets, creating a major security risk. This issue prompted the push for a shift in the authentication model.

Furthermore, the current password-based system is causing significant time losses in the manufacturing process. An internal audit revealed that the average login time per worker is 3 minutes. This means that each time a worker needs to use a tablet, 3 minutes are spent on authentication, which is especially problematic in time-sensitive pipeline and manufacturing operations. In addition, due to a lack of awareness about privacy, many workers leave their sessions open as they step away from the tablets, increasing the risk of data leaks and identity theft.

It is important to understand the baseline and the security posture that we currently get after the previous considerations. Specifically, it is clear that the authentication systems are not up-to-date and leave a huge window of opportunities for attacks. Every information reported previously will be presented in a more structured way in the following sections (4.3).

4.2 Initial requirements and motivation for the PoC

Following the brief introduction in the previous section, we can now explore the requirements and motivations that led to the final Proof of Concept. As mentioned earlier, some of the requirements discussed here are not directly related to passwordless authentication. While they influenced the design and development process, they stem from a broader security assessment conducted on the manufacturing line. Transitioning to passwordless authentication is the primary response to the low-security standards previously outlined, and thus, the implementation of a passwordless system is included in the requirements. However, the remaining requirements are not directly tied to this transition. Passwordless authentication addresses most of the core needs with a single, comprehensive solution. Everything else is considered secondary and will receive less focus in the discussion.

4.2.1 Security requirements

The main security-related goal is to increase the robustness of the login process and bring it to up-to-date standards. Setting up a passwordless authentication flow is a natural requirement after the original interest of the company. Reporting and alerts for misusage are also incorporated into the security requirements. Hereafter is a table that summarizes both Functional and Non-Functional requirements.

4.2.2 Business requirements

The main business-related goal is to lower the login time and to ensure a faster experience for users. This is to reduce wasted time during the log-in and avoid monetary overhead. Furthermore, there should be the least number of changes in the architecture to contain costs. Hereafter is a table that incorporates both Functional and Non-Functional requirements.

Table 4.1. Security Requirements

Requirement Type	Details
Functional Requirements	1. Enhance login security. 2. Implement passwordless authentication. 3. Set up reporting and alerts for misusage.
Non-Functional Requirements	1. Achieve "Advanced" identity status (CISA). 2. Ensure automatic logout. 3. Minimize architectural changes.

Table 4.2. Business Requirements

Requirement Type	Details
Functional Requirements	1. Reduce login time by 50%. 2. Implement a data lake for SIEM and dashboards. 3. Minimize human intervention in the transition.
Non-Functional Requirements	1. Keep costs within the current Azure plan. 2. Utilize existing security keys. 3. Maintain business continuity.

4.2.3 Naming convention

As we continue discussing the Proof of Concept, it will be important to refer back to the original requirements to understand how specific components contribute to the final goal. Within the PoC, six requirements are categorized under "Security" and the remaining six under "Business". When referencing them, the following notation will be used: {B/S}_{F/NF}_{number}. For example, the second non-functional requirement in the Security group will be referred to as S_NF_2.

4.3 Initial assessments

The following section is the natural continuation of Chapter 4.1. It will dive into the details of the first assessment and will serve as a technical background for all the following discussions. Technologies and tools will be covered here, along with most of the limitations and constraints that will shape the development of the Proof of Concept. Specifically, three key pillars have been analyzed:

- **Architecture** - An assessment of the technology stack and the infrastructure that powers the ecosystem of GripGotham.
- **Security** - An assessment of the security practices and posture from the point of view of the authentication, hence, considering only the identity space.
- **Customer Personas** - An assessment of the target user that will be impacted by the transition, including roles and affinity with the technology.

For simplicity, not all details of the assessment outcomes will be presented. Only the relevant information needed to understand how the Proof of Concept is built will be mentioned, allowing the reader to grasp why certain choices were made. Each of the three pillars is equally important, as they will inform discussions about potential alternatives and modifications to the Proof of Concept. It's important to highlight that the **Customer Personas** analysis is crucial when dealing with passwordless technologies. Any change to the authentication paradigm (such as introducing password strength meters, enforcing MFA, or transitioning to passwordless) must be considered from the end users' perspective. If users are not properly trained, aware, and comfortable with the technology, they may bypass, misuse, or ignore the new methods altogether.

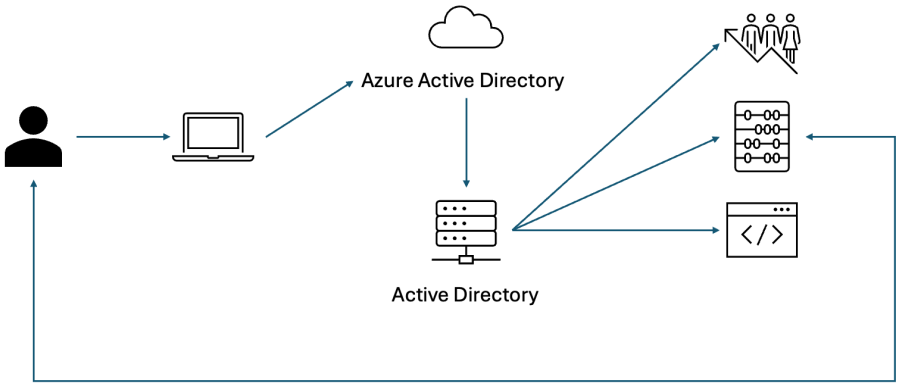

Figure 4.1. Authentication architecture.

4.3.1 Architectural assessment

As previously noted, GripGotham has a broad global presence. However, most of the infrastructure is consistent across its production lines, with slight variations due to different update speeds at individual plants. For the specific line chosen for the Proof of Concept, the following architecture is in place.

The environment is hybrid, combining a cloud tenant in Azure with an on-premises Active Directory forest. This mix of cloud and on-premises services enables the production line to operate smoothly. Additionally, some servers and network components are managed both on-premises and in the Azure Cloud. A wide range of devices is connected to the network, managed through the cloud identity provider Entra ID (more details in the following section, 4.5.2). These devices include laptops for helpdesk and managerial tasks, as well as endpoint devices shared among blue-collar workers. GripGotham oversees the management of these devices, serving as the access point for both cloud and on-premises services. Most of the user-facing equipment (laptops, tablets, etc.) runs Windows 10 with the latest service pack updates, while servers and virtual machines operate on Windows Server 2016 and REHL as the primary operating systems. Power line redundancy is in place, though its details are outside the scope of this work.

4.3.2 Security assessment

This pillar focuses on assessing the current standards for users and authentication flows. Other areas, such as physical or network security, are beyond the scope of this project. Users are managed within the company's hybrid environment, using a Pass-Through Authentication (PTA) model, where users authenticate through the cloud, and the authentication is then forwarded to the on-premises Active Directory. This setup means users interface with the cloud for authentication and federation, facilitated by Entra ID (Azure Active Directory) agents connecting with the on-premises Active Directory. As for the devices used by blue-collar workers, Intune and Defender for Endpoint are properly configured and operational, allowing full control over tablet behavior. Additionally, users are already active within the directory and are regularly monitored and audited. Figure 4.1 is a high-level representation of the original authentication flow.

4.3.3 Customer Personas

The focus here is on the end users most affected by the migration, particularly the blue-collar workers on the manufacturing line. While these users are highly skilled in their production duties, they have limited expertise in information technology, especially regarding privacy and security. As mentioned earlier, this has led to poor security practices, such as writing passwords on sticky

notes or leaving sessions open for extended periods. Given their profile, it is critical to ensure a smooth, frictionless transition to the new authentication system. We will explore this aspect in greater detail later.

4.3.4 Implementation tentative by GripGotham

It is worth noticing that GripGotham has already initiated a preliminary implementation of passwordless authentication. This initial attempt demonstrates a forward-thinking approach and a genuine interest in adopting cutting-edge security technologies. It reflects the organization's vision and commitment to advancing its cybersecurity posture. However, without a comprehensive transition plan and the thorough analysis conducted in Chapter 2 and Chapter 3, this initial effort ultimately faced significant challenges, leading to its failure.

The primary issue lies in the complexity of transitioning an entire organization to passwordless authentication, especially one with a vast and intricate global infrastructure like GripGotham's. While implementing passwordless solutions can be relatively straightforward for individual users, organizations must contend with existing legacy systems, deeply structured processes, and an architecture that was not originally designed to support this modern authentication method.

This experience highlights the necessity of a well-structured and carefully considered plan. The challenges faced by GripGotham highlight the importance of conducting a complete analysis of the existing architecture, identifying potential obstacles, and developing a tailored strategy that addresses the specific needs and complexities of the organization. Without such a plan, even the most promising technological initiatives can falter, as demonstrated by GripGotham's initial steps into passwordless authentication.

4.4 Application of the Framework

With the background established and the requirements outlined, the next focus is the framework itself. The work developed in the first contribution (3) will prove useful in the following analysis. Specifically, the proposed framework will be applied to the GripGotham scenario. To avoid redundancy, the following structure will guide the discussion:

- **Checklist** - 3.3.1
- **Target** - 3.3.2
- **Roadmap** - 3.4.1

For privacy reasons, not all outcomes and applications can be disclosed. However, a general overview of the three steps will be provided to allow the reader to follow the technical dissertation that follows. Insights into the challenges faced during the application of the framework will also be shared, helping the reader understand the complexity or simplicity of each phase, depending on the project.

4.4.1 PoC Checklist

As discussed during the framework's development, the checklist ensures a clear understanding of the architecture, identifying which capabilities can be leveraged, modified, or constrained. These discussions often take longer than expected, as was the case with GripGotham. No single person could answer all the questions, causing delays. For this Proof of Concept, it was necessary to involve Security Architects, Infrastructure Architects, and the Service Manager for the production line. Some key findings were mentioned earlier, while others remain unaddressed. The following key elements will guide future decisions:

- Currently, authentication is carried thorough username and password.

- MFA is not configured nor enforced.
- The Certification Authority is present and running (it issues certificates and revokes them).
- Personal devices are not allowed for work related tasks.
- Elements can be added to the cloud infrastructure, not to the on-premises one.
- There is no awareness on passwordless authentication on the production line.
- Users already make use of laptops and badges.
- Devices support all passwordless authentication method (at least the one cited in this work).
- We have no multi-cloud, yet, the environment is hybrid.
- There is no monitoring system for the login.

4.4.2 PoC Target

Once the assessment is completed and the checklist is finalized (with only some answers mentioned above), the next step is defining a target. For GripGotham, two key factors are crucial: minimal cost changes and leveraging the already purchased security keys. These two aspects are somewhat conflicting, but a balance can be found between them. The limiting factor is that the company has already invested in Security Keys and wants to maximize their use. Once this Proof of Concept proves successful, the intention is to purchase more.

Given these constraints, constructing a flowchart based on the requirements becomes unnecessary. Security Keys are the preferred and most valued option for the company, though this may not be the optimal choice. Examining one possible flowchart 3.4, which prioritizes minimal architectural changes and cost reduction, it becomes clear that Security Keys are not the ideal choice. If the goal is to minimize changes and reduce costs, certificate-based authentication would be more suitable. GripGotham's CA could issue the necessary certificates with minimal modifications, and they could be stored in existing badges. This approach offers two advantages: it maintains the original scope and goal, and more importantly, users would not need to adopt new devices. Familiarity with existing technology is always a plus when introducing new methods.

That said, the commitment remains to Security Keys. Although an architecture based on certificate-based authentication would have been preferable (3.2), the choice is limited to the Security Keys architecture 3.1.

4.4.3 PoC Roadmap

Regarding the roadmap, no prior passwordless authentication has been deployed. This means the company will need to follow each of the four steps without shortcuts. The deployment phase is the primary focus of this work. Following deployment, the Restrict and Shift phases must be initiated from a policy perspective. The overall strategy aligns with the plan described above, ultimately leading to the removal of passwords from the authentication process. This strategy has been implemented for a single manufacturing line in one country, and other locations may require adjustments to complete the transition. The Restrict, Shift, and Remove phases are beyond the scope of this work and may be addressed in future research.

4.4.4 PoC Design

We now have a solid understanding of the current architecture, along with the details of what can be modified, added, or must remain unchanged. Furthermore, we have an outline of the transition roadmap and the technologies to be used. What remains is the design of the Proof of Concept (PoC). This section is intentionally placed before the technical discussion to give non-technical readers an overview of the PoC design, without delving into the technical complexities. Specifically, we know that we have to refer to Figure 3.1 for the design of the solution. By examining the "Lego Brick" blocks, we can visualize how the solution will be constructed:

- **Onboarding Service** - This service has to be designed partially from scratch and will be the core of the Proof of Concept as will be discussed in the next chapter.
- **Key Vault Service** - This service will be implemented within the Azure space and connections and automation will have to be created.
- **Authentication Service** - As will be discussed shortly, this service will make great use of Entra ID to authenticate the user with the keys. It will have to be tuned to allow for passwordless authentication and to respond to challenges.
- **Backup service** - As we are dealing with FIDO2 Keys, no backup is involved.
- **Retire Key Service** - This service will be deployed as a custom service as there is no cloud automation that can serve this purpose.

Additionally, a small integration from the One-Time Code design will be needed. While details will be discussed later, a code distribution service will be developed to address specific requirements. Although this service isn't included in the original architecture, it's intuitive that building blocks like this can be integrated into the overall design, just like adding new Lego bricks to a completed structure.

From an architectural standpoint, nothing changes compared to the previous chapters. All prior considerations about the services still apply to the Proof of Concept.

4.5 Technical implementation

The following section provides a technical breakdown of the PoC, highlighting key details. As previously mentioned, all information that could identify the original company or its users has been removed and replaced. In fact, this PoC can be replicated in any test environment with minimal familiarity with Azure and ARM templates. From a high-level overview to the specifics of each component, the goal of this hands-on discussion is to demonstrate the practicality of the transition and to clarify the reasoning behind the designs proposed earlier. As intuition may suggest, this chapter can be entirely skipped if the technical details are not of interest. On the contrary, what follows from this chapter may be of great interest to a designer or architect who is focused on outcomes, results, and issues.

4.5.1 High level overview

To fulfill the requirements there exists no pre-built solution that can do it all-in-one, therefore a combination of components has to be designed in order to achieve 100% coverage of the requirements. From a high-level view, the proposed solution consists of a series of automation that will be responsible for enabling passwordless login, onboarding a new FIDO2 user key, monitoring user's behavior, and guaranteeing an automatic logoff. The purpose of achieving the requirements through automation and logic is that on a large set of users (5000 for a single line), human intervention should be minimal and targeted to some corner cases.

The final PoC consists of 4 logical components, this division is just a matter of explanation, as there are no boundaries set in the environment and every component can be further connected to each other.

- **Component 1 - User transition management:** This component is responsible for all the operations that are carried out on the user object of the directory. It is responsible for enabling passwordless authentication, onboarding new keys, and login information for the other components.
- **Component 2 - Device management:** This component is responsible for the controls and automation on the user device, it provisions the scripts and updates automatically and guarantees a smooth logoff.

- **Component 3 - SIEM Integration:** This component gets data from the other sources and raises alerts when certain scenarios occur or runs playbooks to remediate issues.
- **Component 4 - Reporting:** This component is responsible for processing data and presenting it through the use of reporting tools. The statistics and visualizations can be used by managers to track the status of the deployment.

It is important to note that to start the transition towards passwordless authentication with Hardware Tokens, a Security key must be possessed. As mentioned previously, the GripGotham company bought the first round of 100 keys, thus a batch of 100 users can be transitioned. In the following sections, the two main components (Component 1 and Component 2) will be discussed in more detail as they represent the core of the solution. Component 3 and Component 4, will also be treated but more details can be found at the project repository: Passwordless transition PoC - GitHub repository

4.5.2 Technology stack

To build the Proof of Concept, a huge set of assets and technologies has been leveraged. As the majority of the infrastructure runs on Azure, the components have been designed to fit into the current technology stack of the company. This is aligned with what has been discussed previously about consumption and the possibility of operating only on the cloud side of the infrastructure. To highlight the most important:

Entra ID

Entra ID (formerly Azure Active Directory) is Microsoft's cloud-based identity service that manages user access to apps and resources. It enhances security with features like single sign-on and multi-factor authentication, making it easier to control who can access what in your organization.

Azure Logic Apps

Azure Logic Apps is a tool for automating workflows between apps and services without managing the underlying infrastructure. It allows developers to write code integrating with cloud components naively. Common functionalities are syncing data or sending alerts, streamlining repetitive tasks, and integrating processes seamlessly.

Azure SQL Database

Azure SQL Database is a fully managed, cloud-based database service that handles everything from backups to scaling. It is suitable to build and run data-heavy applications without worrying about the underlying infrastructure, offering the possibility of automatic scaling and modular throughput.

Microsoft Graph

Microsoft Graph is an API that lets you tap into data and services across Microsoft 365, like user info, calendars, and files. It's the backbone for integrating apps with the Microsoft ecosystem, enabling deep connections and richer data-driven experiences.

Microsoft Power BI

Microsoft Power BI turns raw data into interactive visuals and dashboards that make insights easy to understand. It connects to various data sources, helping teams analyze and share key business metrics quickly and visually.

Intune

Microsoft Intune helps manage and secure mobile devices and apps across an organization. It ensures that devices meet security standards, protecting corporate data while giving employees flexibility in how they work.

Microsoft Sentinel

Microsoft Sentinel is a cloud-based SIEM/SOAR tool for spotting and responding to security threats in real time. It uses automation and playbooks to sift through data, helping organizations detect issues early and respond swiftly to protect their systems.

4.5.3 Preliminary considerations

Before diving into the specifics of each component it is worth discussing some concepts that will be helpful in the description of the components:

1. Users in Microsoft Entra ID

In Microsoft Entra ID, users are represented by their profiles, which include various attributes and properties. On a high-level view, the properties of a user can be grouped into three categories:

- **Identity Information:** such as name, email address, user principal name (UPN), and others.
- **Authentication Information:** a list of available authentication methods for that user (eg. Password, MFA, biometric, passwordless, etc).
- **Roles and permissions:** a list of assigned or eligible roles for that user, used for role-based conditional access.

What is interesting for the purpose of the PoC, is that Hybrid environments such as the one of GripGotham can be managed by a large part by Entra ID, without the need for interfacing with the Active Directory on-premise. As a reference, Figure 4.2 is a sample user created within the PoC Entra ID directory. Given the fact that every user is authenticating towards the cloud (Pass-Through Authentication), and considering this capability of Entra ID, Component 1 and Component 2 will only have to operate on the cloud side.

2. Temporary Access Pass (TAP)

A Temporary Access Pass (TAP) in Microsoft Entra ID is a time-limited passcode that can be configured for either single or multiple uses. Its primary function is to help recover accounts when users lose access to strong authentication factors due to loss or technical issues. TAP serves as the strongest authentication method within Entra ID, allowing users to perform actions that typically require a robust authentication factor. For reference, Figure 4.3 illustrates the TAP creation process.

3. Microsoft Graph API

The Microsoft Graph Application Program Interface is a RESTful API that allows access and interaction with a variety of range of Microsoft Cloud services and resources. The goal is to provide a unified endpoint to manage data and services from Microsoft 365 to Security to Power Automate. Microsoft Graph API comes with its own set of permissions and can be aligned with Entra ID.

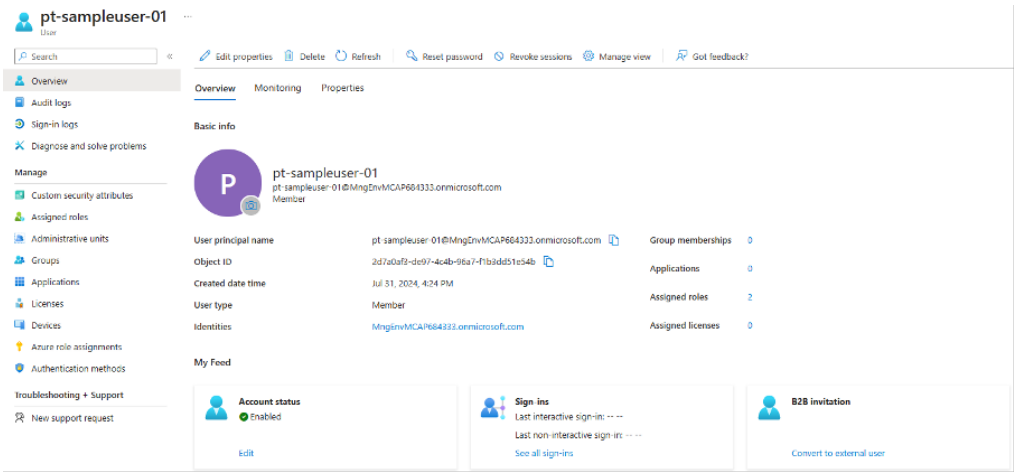

Figure 4.2. Entra ID user characteristics.

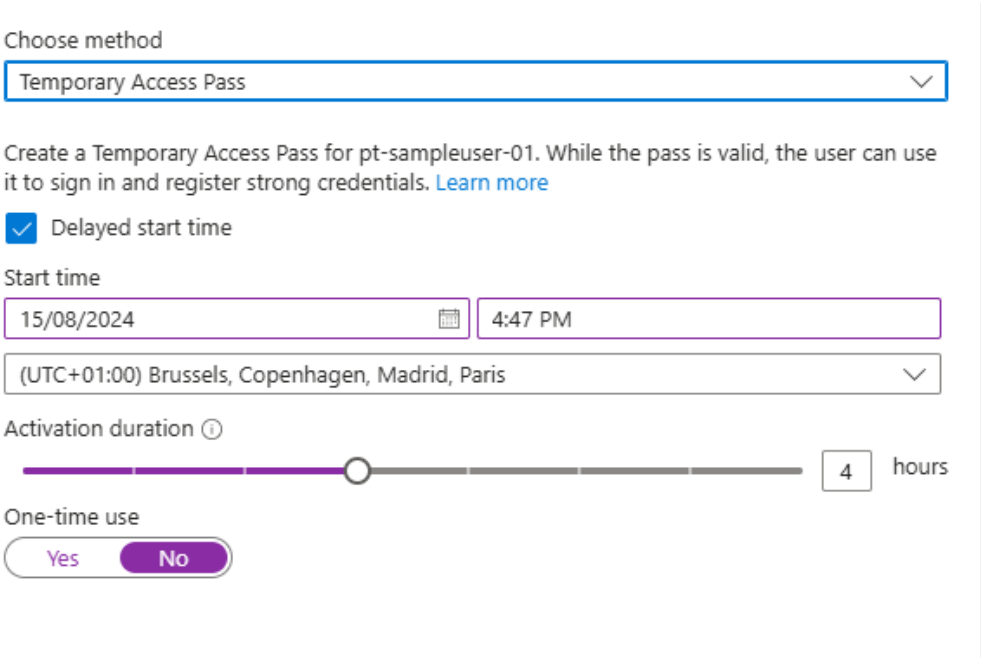

Figure 4.3. Temporary Access Pass creation.

4.5.4 Component 1 - User Transition Management

As previously mentioned, this component is responsible for managing the users in the directory and handling all passwordless related operations. This component is made of 3 logic apps that handle separate phases of the transition as well as 3 directory groups that support the automation. Interesting aspects to be treated are: **Groups and Users**, **Logic apps and automation** and **Permissions**.

1. Groups and Users

The proposed solution centers around the creation of three custom groups to manage users as they progress through the transition stages. Each group represents a different phase of the transition.

- **pt-uncommitted-group** - This group hosts users who are ready to start using the keys.

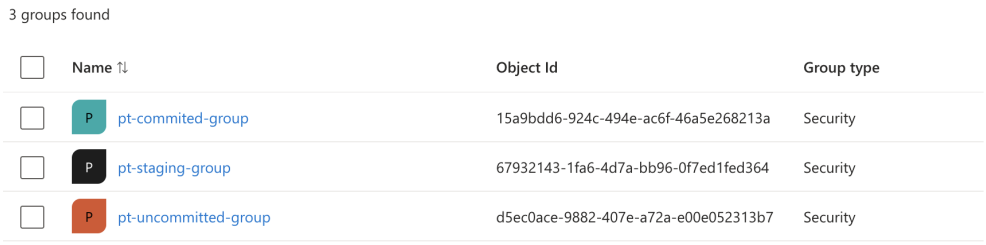

Figure 4.4. Group division in Entra ID.

- **pt-staging-group** - This group hosts users who are in the process of onboarding their keys and finalizing their first passwordless login.
- **pt-committed-group** - This group hosts users who have already transitioned to passwordless authentication.

Figure 4.4 provides a reference in Entra ID for the mentioned group division. These groups act as containers in the directory, and no user can join them unless assigned by the automation. This setup provides a clear view of each user's status. For example, if a blue-collar worker is in the **pt-staging-group**, it indicates that their Security Key is ready to be enrolled, and they are about to activate it and link it to the directory. Similarly, if a user is in the **pt-committed-group**, it means their key has been registered and they have completed their first passwordless login.

This structure is particularly valuable when viewed in the context of the overall strategy. Knowing which users have onboarded their keys and transitioned to passwordless authentication is critical for the Restrict phase. It allows targeted actions based on whether users are ready to eliminate password-based authentication or are nearing that point.

2. Logic apps and automation

The entities that perform operations on users and groups are automated within the logic apps. Other than interacting with the users and the groups, the cloud automation interacts with Entra ID and with a series of SQL databases for logging. As mentioned above, three logic apps are necessary to complete the process:

- **pt-la-fromUncommittedToStaged**
- **pt-la-TAPdelivery**
- **pt-la-fromStagedToCommitted**

The abbreviations in the name stand for: "passwordless-transition" (pt) and "logic-app" (la). As the naming suggests, each of the automation is dedicated to operating at a specific stage of the transition. As a high-level overview, the diagram in Figure 4.5 describes the interaction of groups and logic apps.

Hereafter are discussed the details of each logic app, for the sake of brevity, only some code snippets are reported for each automation. For the full code base of the logic app, refer to the Github repository.

LogicApp 1 "pt-la-fromUncommittedToStaged"

This logic app is responsible for starting the entire process and handling some parsing of the users. It is important to note that this automation will run in the cloud in a scheduled manner, meaning that with a recurrence of X minutes, the automation will be run again. This is because the group can be filled with a user as soon as the physical key is ready, and the automation should take care of that. The flow is described as follows:

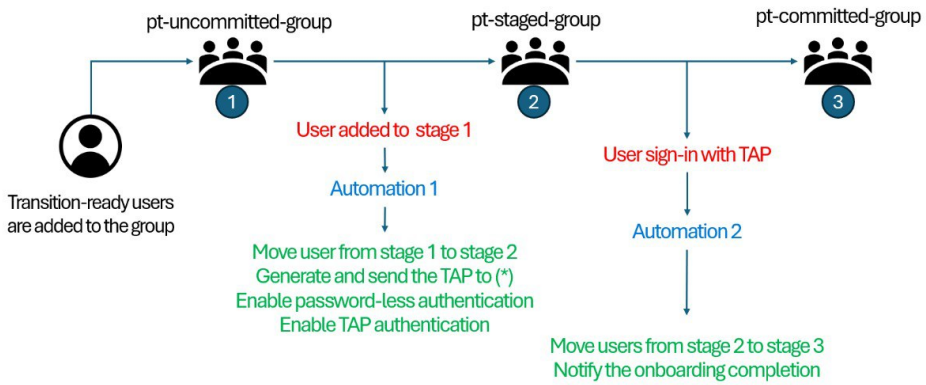

Figure 4.5. Logic apps and groups interactions.

- Query the directory (Entra ID) and retrieve the full list of users belonging to the "pt-uncommitted-group".
- If the query returns a non-empty array of users, the automation will loop through the user principal names and send a POST request to Entra ID to enable passwordless authentication for that specific user.
- Each user is then moved by the following group (pt-staging-group) and removed from the current group (pt-uncommitted-group).
- The final step is to log the information, and the changes performed by the logic app. An insert statement is performed to an SQL database.

LogicApp 2 "pt-la-TAPdelivery"

The main purpose of this logic app is to issue and monitor the use of the temporary access pass (TAP). As discussed above, the temporary access pass is the strongest authentication method that can be used in Azure and will allow users to onboard the FIDO2 Keys without prompting for MFA. The flow works as follows:

- Query the directory (Entra ID) and retrieve the full list of users belonging to the "pt-staged-group".
- After receiving the list of users, check if the user already has a TAP assigned. If it does, then loop to the next user.
- If it does not, generate a TAP and append it into a string variable.
- Send the list of the TAP to the desired helpdesk mail.
- The final step is to log the information, and the changes performed by the logic app. An insert statement is performed to an SQL database.

The following is a code snippet that is responsible for issuing the Temporary Access Pass for the automation.

```
"HTTP_-_Generate_TAP": {
"inputs": {
    "authentication": {
        "audience": "https://graph.microsoft.com",
        "type": "ManagedServiceIdentity"
```

```
        },
        "body": {
            "isUsableOnce": false,
            "lifetimeInMinutes": 60,
            "startDateTime": "@{utcNow()}"
        },
        "method": "POST",
        "uri": "https://graph.microsoft.com/v1.0/users/
        @{body('Parse_JSON_-_Parse_single_user')?
        ['id']}/authentication/temporaryAccessPassMethods"
    },
    "runtimeConfiguration": {
        "contentTransfer": {
            "transferMode": "Chunked"
        }
},
```

LogicApp 3 "pt-la-fromStagedToCommitted"

This automation takes care of checking whether the user has successfully onboarded the FIDO2 Key and performed the first login. If those conditions are met then the user is moved to the last group that hosts successfully transitioned users. The flow works as follows:

- Query the directory (Entra ID) and retrieve the full list of users belonging to the "pt-staged-group".
- After receiving the list of users, query, for each specific user, whether the key was onboarded or not. If this is the case, then move the user to the "pt-committed-group" stage and notify the corresponding administrator.
- If the user has not onboarded the key, the logic app checks if the previously issued TAP is still valid. If so, the automation proceeds with the next user.
- If the TAP is expired, then the user is removed from the "pt-staging-group", and it is moved back to the "pt-uncommitted-group".
- The final step is to log the information, and the changes performed by the logic app. An insert statement is performed to an SQL database.

The following is a code snippet that is responsible for notifying the successful transition of a user.

```
    "type": "Http",
    "inputs": {
      "uri": "https://graph.microsoft.com/v1.0/users/<sending_user>/sendMail",
      "method": "POST",
      "body": {
        "message": {
          "subject": "Successful transition for user
          @{body('Parse_JSON_-_Parse_single_user')?['id']} ",
          "body": {
            "contentType": "Text",
            "content": "The user @{body('Parse_JSON_-_Parse_single_user')?['id']}
            has successfully setup the FIDO2 Key.
            Moving the user forward to committed group."
          },
          "toRecipients": [
            {
```

```
            "emailAddress": {
              "address": "<receiving_user>"
            }
          }
        ],
        "ccRecipients": []
      },
      "saveToSentItems": "false"
    },
    "authentication": {
      "type": "ManagedServiceIdentity",
      "audience": "https://graph.microsoft.com"
    }
  },
```

3. Permissions

As the automation will act on the directory level by changing properties of the users such as authentication methods and groups they belong to, it needs specific permissions and roles. Other than the directory permissions, appropriate rights over the logging database should be granted. Moreover, it is fundamental that automation has the least number of permissions required. This will ensure adherence to the least privilege principle and limit the scope of action in case of compromise. Entra ID treats logic apps as users when it comes to roles and permissions. This means that each logic app can receive a set of permissions that can be revoked and audited if needed.

For what concerns the directory-level permissions, to carry out the work required by each automation, the following permissions are required:

Logic App	Permissions
`pt-la-fromUncommitedToStaged`	- GroupMember.ReadWrite.All - Mail.Send - Directory.Read.All
`pt-la-TAPdelivery`	- GroupMember.ReadWrite.All - UserAuthenticationMethod.ReadWrite.All
`pt-la-fromStagedToCommitted`	- GroupMember.ReadWrite.All - Mail.Send - UserAuthenticationMethod.ReadWrite.All

Table 4.3. Permission sets for different logic apps

As a reference Figure 4.6 shows the different permissions assigned to the logic app.

4.5.5 Component 2 - Device Transition Manager

This second component is responsible for managing operations that happen on the device itself. It allows to deploy configuration and to enforce policies defined by the administrators. It integrates into the overall solution and specifically with "Component 1" to guarantee a smooth logoff operation and to allow for passwordless login.

The two main contributions of this component are the following:

- Allow passwordless login and registration at the device level.
- Ensure that the session is logged off when the user finishes.

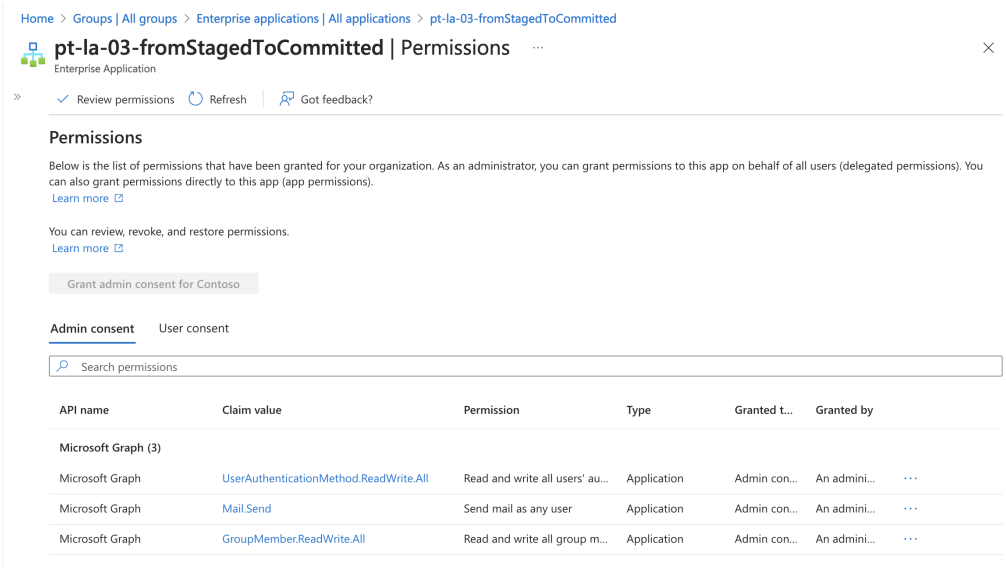

Figure 4.6. Permissions for pt-la-fromStagedToCommitted

To carry out the previous tasks Microsoft Intune will be of paramount importance for mainly two reasons: it allows to deployment of the right policies and custom scripts on the endpoints, then it will be helpful when massively testing the solution to the entire manufacturing line as it can provide a single interface for multiple device types.

Enable passwordless registration

This specific task is what Microsoft Intune was originally built for, deploy configuration on multiple devices without interacting directly with them. This feature allows us to set up what is known as a "Policy" that will be shared among all devices in the scope of the policy. The specific action that we want to take is "Enable Passwordless Experience" for all the devices that are Joined with Entra ID. This setting will unlock the possibility of having passwordless authentication for all devices that have Windows 10 or above. In the specific scenario considered by the PoC, all the tables are Windows 10 devices that are Entra ID joined, hence, they can receive the policy and align with the desired configuration. To roll out the changes, Microsoft Intune will have to synchronize with all the devices, and it may take different times depending on network and resource availability.

The policy configuration is as follows:

Property	**Value**
Name	pt-authentication-policy
Platform	Windows 10 and later
Assignments	All Devices
Configuration Settings	Authentication, Enable Passwordless Experience

Table 4.4. Policy Configuration

With this set of parameters, the policy will enable passwordless authentication for every device in the testing environment. This step may be redundant with respect to what has been done by the first logic app of Component 1. It is actually not overlapping and the two operations are not redundant. The log app ensures that the user can register the passwordless authentication method in the user directory, this means that in the active directory, the user object will have a new field with the new authentication method. On the other end, with this policy, we are enabling passwordless login and registration at the device level. If a user has the possibility of registering

a FIDO2 but the device is not supporting or allowing this technology, then there will be no way of providing passwordless access to that device.

Ensure proper logoff

This second part is meant to ensure that the session is properly logged off when the user is not using the device anymore. This aspect is directly linked to the privacy issues discussed in the initial requirement and motivation for the PoC. Blue collars of the manufacturing line are not fully aware of the privacy issues related to an open session on the tables. Therefore, they expect that by removing the key, the tablet automatically locks the session, and they can walk away with the key in their pockets.

This behavior is not typical of Security Keys, specifically, they serve as the primary method for authentication and as soon as they are done with the cryptographic operations they are not used anymore. This means that after the authentication phase, the keys are just a dormant piece of hardware waiting for another authentication round or to be removed. Hence, a removal of the key will not trigger any action on the device itself, nor on the authentication token.

The best way to track events that happen in Windows is through the event viewer. Here, events and logs are grouped by type and can be queried to find specific occurrences in time. Specifically, among Windows logs, Service logs, and Application logs, the USB insertion and removal are tracked in the Windows logs section. Moreover, this section has multiple subsections containing "Application", "Security", "Setup" and "System". Among them, the same event can have multiple logs and can be found in more than one subsection. This is because the event may be registered and trigger other tasks depending on the subsection that it is found in.

Events are mapped into event IDs and by finding the right event id a task can be created on top of the occurrence of the event. For this specific case, the event of a USB removal is mapped to event ID 6416. This means that by tracking the occurrences of this event, it is possible to trigger an action and to act on that. This duty is carried out by the "Task Scheduler" which is responsible for setting up recurring tasks triggered by events found in the Windows logs (event viewer). Each task has the following sections: General, Triggers, Actions, Conditions, Settings, and History. For the specific application in this PoC, Triggers and Actions are the ones that need to be modified. Under the Trigger section, the correct event id is represented by the following string:

```
"Log: Security; Source: Microsoft Windows Security Auditing; Event ID: 6416"

Moreover, the follow-up action to this trigger is the following:

C:/Windows/Temp/key-presence-detector.exe
```

The task is ready, but it has to be first packaged before shipping it to all the relevant machines. The reason for this "packaging" is that it would not be appropriate to send a standalone script to all the tablets without proper wrapping. The industry-leading solution when it comes to deploying scripts with Microsoft Intune is to use a wrapper to include support for installation, uninstallation, detection, and logging. The script alone would be enough for a manual installation, yet the number of devices considered here makes this method timely unfeasible. The correct application package to distribute is a Win32-compliant package. Win32 is a set of libraries and tools from Windows programming that, among other things, define the way that applications should be deployed. As previously mentioned, the program

```
C:\\Windows\\Temp\\key-presence-detector.exe
```

would require also the following pieces of code:

- **Install** - install.ps is a custom PowerShell script that is responsible for the creation of the task, installation of the dependencies, and import of the key-presence-detectof.exe program.
- **Uninstall** - uninstall.ps is responsible for deleting all the dependencies and files that have been created by the install.ps script. It allows us to un-do all the actions done by the previous program.
- **Detection** - detection.ps is a custom script responsible for detecting whether or not a good installation was performed, and it is used by Microsoft Intune to manage the installation phases

As a reference, the following code snippet handles the installation process and logging aspects of the Device Management Component.

```
    # Register a new Scheduled Task using the XML
try {
    Register-ScheduledTask -xml
    (Get-Content C:\Windows\Temp\pt-session-logoff-task.xml | Out-String)
    -TaskName "pt-session-logoff-task" -TaskPath "\" -User $principal
    Log-Message "Registered Scheduled Task pt-session-logoff-task" 0
} catch {
    Log-Message "Failed to register Scheduled Task pt-session-logoff-task"
    $_.Exception.HResult
}

# Additional logging for debugging
try {
    $taskXmlContent = Get-Content C:\Windows\Temp\pt-session-logoff-task.xml |
    Out-String
    Log-Message "Task XML content: $taskXmlContent" 0
} catch {
    Log-Message "Failed to read Task XML content" $_.Exception.HResult
}
```

4.5.6 Component 3 and 4 - SIEM Integration and Reporting

The final two components of the Proof of Concept, SIEM Integration and Reporting, work hand-in-hand, and are discussed together here as they share a common foundation. Both components are crucial for fulfilling the project's requirements, particularly when it comes to monitoring and analyzing the transition process from password-based to passwordless authentication. However, due to privacy concerns, not all specifics can be disclosed.

These components are responsible for tracking and assessing the events that occur during the transition, as well as evaluating the data once the process is completed. While they leverage data from the previously discussed components, each serves a distinct purpose. The SIEM Integration component focuses on ingesting and correlating security-related data, ensuring real-time monitoring and incident response. Meanwhile, the Reporting component visualizes these data points, presenting them to various stakeholders, such as security architects, service managers, and operational teams. Since both components already existed within GripGotham's infrastructure, the task at hand involved integrating and customizing them to meet the needs of the passwordless authentication transition project. The technical logic behind each component is explained below, and a modified configuration set can be found in the project repository.

SIEM Integration Component

A Security Information and Event Management (SIEM) system is integral to modern cybersecurity frameworks, as it collects, processes, and analyzes security data to detect potential threats. In

GripGotham's case, their Azure cloud infrastructure includes a fully operational SIEM system, Microsoft Sentinel, which provides real-time monitoring, threat hunting, and automated responses to security incidents. Sentinel ingests data from various sources such as virtual machines, Active Directory logs, and individual user devices.

The process of shifting from traditional passwords to passwordless methods introduces potential attack vectors that must be monitored closely. For instance, when a Temporary Access Pass (TAP) is issued to a user during the transition, attackers might attempt to intercept it and use it to gain unauthorized access. Sentinel has been configured to track these security-critical events by correlating data from various sources. If, for example, an unauthorized login attempt is detected from an unusual location or network, the SIEM can immediately flag the incident for review or trigger an automated response through the SOAR (Security Orchestration, Automation, and Response) capabilities of Sentinel. This ensures that potential security threats are mitigated in real-time.

A key part of this monitoring involves the use of Hunting Queries, which search for specific events in the data and trigger responses based on the results. For example, a query may check if a TAP was issued and subsequently used from an unexpected location. If any anomalies are detected, alerts are raised, and further actions, such as escalating the case to the security team, are initiated automatically. Here is an example of a hunting query that has been integrated:

```
SigninLogs
| where Identity == "user@example.com"
| project TimeGenerated, Identity, Location, AuthenticationDetails,
SigninStatus, UserPrincipalName, AppDisplayName, ClientAppUsed, IPAddress
| extend City = tostring(Location["city"]),
     State = tostring(Location["state"]),
     Country = tostring(Location["countryOrRegion"]),
     AuthMethod = AuthenticationDetails[0].authenticationMethod
| summarize Logins = count() by bin(TimeGenerated, 1d), Identity, City,
State, Country, AuthMethod, ClientAppUsed, IPAddress
| order by TimeGenerated desc
```

This query specifically tracks user sign-ins following the issuance of a TAP, cross-referencing login attempts with geolocation data to detect any suspicious activity. If any anomalies arise, actions can be triggered within the SOAR framework, ensuring security breaches are swiftly dealt with.

Reporting Component

While the SIEM focuses on real-time security monitoring and automated incident response, the Reporting component is concerned with providing clear, actionable insights to the company's stakeholders through visual data presentations. This component integrates directly with Power BI, a business intelligence tool that GripGotham already has in place. By visualizing data stored in Azure SQL databases, this component helps management and operations teams understand how the passwordless transition is progressing and identifies areas that might need further attention or improvement.

Power BI dashboards pull data from the same data pool as Sentinel, but the way the data is presented differs significantly. These dashboards are designed to track several key metrics related to the transition process. For example, management might want to know how long it takes for users to transition from password-based logins to passwordless authentication, or how long a Temporary Access Pass remains active and valid. The insights provided by these dashboards offer a high-level view of the transition process, enabling stakeholders to make informed decisions. This is a small set of the visualized metrics:

- **Time to Transition** - This tracks the total time that a user needs to complete the transition. Starting from the moment that the user joins the pt-uncommitted-group to the time of the first login with passwordless authentication.

- **Effective Time to Transition** - This tracks the total time that a user needs to complete the transition. Starting from the moment that the user receives the Key and Temporary Access Pass to the time of the first login with passwordless authentication.
- **Effective Time to Live TAP** - This tracks the time in which the Temporary Access Pass is alive and usable, not the default time.
- **Time to Setup** - This tracks the time between the use of the Temporary Access Pass and the moment that the Active Directory registers the public key of the Security Key.
- **Time to First Login** - This tracks the time between the use of the Temporary Access Pass and the moment that the user logs in with passwordless authentication.
- **Effective Time to Reset** - This tracks the time between the notification of a lost key and the new login with the replacement Security Key.

4.6 Testing

Once the four components of the solution were deployed, the next phase was testing, particularly to evaluate how the system handled various edge cases. Initially, the Proof of Concept (PoC) was set up and tested in a development environment. However, this preliminary deployment did not cover all potential scenarios, so more extensive testing was necessary. The following step involved deploying the solution in a dedicated testing environment on the client's side, specifically for GripGotham.

This allowed for more comprehensive testing and an assessment of the solution's performance in an environment that closely resembled its intended real-world use. It was anticipated that not all tests would succeed on the first attempt. While the tests were designed to address core requirements, these were slightly modified to align with the specific conditions of the testing environment. Given the complexity and scale of GripGotham's infrastructure, it was not feasible to fully replicate their entire setup for testing. As a result, some tests conducted on the Microsoft side used a simplified version of the actual environment.

Due to confidentiality agreements, detailed test results cannot be shared. However, a high-level overview of the process is provided to illustrate the approach and the types of evaluations conducted. This overview highlights the scope of this phase and the adjustments made to ensure the solution meets the necessary requirements in a practical setting. These tests can be reproduced using the repository, though full access to the user directory is required to execute them correctly.

4.6.1 First Batch of Tests

The set of tests selected for this report focuses on the functional aspects of the Proof of Concept (PoC). Each test is designed to assess a specific functionality aligned with the defined requirements. The following tests, along with their outcomes, are detailed:

- **Test 1:** User passwordless onboarding - A new user tries to register a valid key with no issue during the process.
- **Test 2:** User passwordless onboarding 2 - New user tries to register a valid key with network problems (high delays).
- **Test 3:** User passwordless onboarding 3 - User tries to register an already bounded key.
- **Test 4:** User passwordless reset - User tries to reset their passwordless authentication method due to a loss or compromise.
- **Test 5:** Policy and script deployment - New device is enrolled into the pool and enabled for passwordless login.

- **Test 6:** User logoff automation - User walks away with the key.

As noted, these tests are focused solely on functional aspects. Unit tests have also been conducted and are available in the projectâs repository, along with detailed instructions to reproduce them. Below, we present the original outcomes of these functional tests:

Test Number and name	Outcome
Test 1 User passwordless onboarding	Success
Test 2 User passwordless onboarding 2	Failure
Test 3 User passwordless onboarding 3	Success
Test 4 User passwordless reset	Success
Test 5 Policy and script deployment	Failure
Test 6 User logoff automation	Success

Table 4.5. Sample of the original tests

4.6.2 Improvements and followups

As can be seen from the previous table, not all tests were successful, moreover, many more were conducted and some of them failed as the one reported above. This forced a change in the code base or on the integration of the components to make sure that 100% coverage was hit. Specifically, to cover the two reported failures, the first set of logic apps has been changed and the automation shipped to the devices had to be modified. For reference purposes, the following are the snippets of the changed code:

Code added after Test 2:

```
"HTTP_-_Remove_previous_TAP": {
    "inputs": {
        "authentication": {
            "audience": "https://graph.microsoft.com",
            "type": "ManagedServiceIdentity"
        },
        "method": "DELETE",
        "uri": "https://graph.microsoft.com/v1.0/users/
        @{body('Parse_JSON_-_Parse_single_user')?['id']}/
        authentication/temporaryAccessPassMethods/@{item()['id']}"
    },
    "runAfter": {
        "HTTP_-_Remove_user_from_Staged_group_2": [
            "SUCCEEDED"
        ]
    },
    "runtimeConfiguration": {
        "contentTransfer": {
            "transferMode": "Chunked"
        }
    },
"type": "Http"
```

Code added after Test 5:

```
try {
$quserOutput = quser
Log-Message "quser command executed" 0
```

```
Log-Message "quser output: $quserOutput" 0

$principal = $quserOutput | Select-String -Pattern "Active" |
ForEach-Object {
    $_.ToString().Split(' ', [System.StringSplitOptions]::RemoveEmptyEntries)[0]
}
Log-Message "Parsed principal from quser output: $principal" 0

$principal = $principal -replace '[><]', ''
Log-Message "Cleaned principal: $principal" 0
} catch {
    Log-Message "Failed to retrieve user account using quser"
    $_.Exception.HResult
    Log-Message "quser output: $quserOutput" 1
}
```

The full list of changes related to the Tests can be found by looking at the history of the GitHub comments. Moreover, in the testing space, a document keeps track of modifications due to failed tests.

4.7 Blockers and issues

As is often the case with technical deployments, challenges arose due to the real-world conditions that could not be fully replicated in the test environment. Typically, the pre-production environment is intended to closely mimic the production setup, but achieving an exact replica is not always possible. In the case of GripGotham, although the test environment was designed to be as close as possible to production, it was not perfect, and this led to some issues during deployment.

This section is not intended to serve as a comprehensive troubleshooting guide, nor does it aim to cover every possible problem that can occur in deployment. Rather, its purpose is to highlight some of the technical blockers encountered and the solutions that were found.

4.7.1 Logic app permissions

The first issue involved permissions for the Logic App. As outlined earlier, the code within the Logic App does not inherently handle permissions. During development, it is assumed that the necessary permissions will be applied externally to allow the app to function properly. However, if the correct permissions are not granted, the behavior can become unpredictable, some parts of the code may execute successfully, while others may fail.

In this case, the second Logic App, which was responsible for generating Temporary Access Passes (TAPs), encountered a permissions issue. Although TAPs were generated successfully, they were not linked to the user directory, resulting in a collection of authentication codes with no associated users. Once the correct permissions were applied, a second issue emerged. The connection to the email distribution system, in this case, the internal Exchange server, was not functioning as expected.

To be more specific, every email within the internal system needs to be sent by a recognized sender. While this is straightforward with human-to-human emails, where the sender has a valid email address in the domain, cloud-based workloads require proper integration with the Exchange server, as no human is directly involved in sending the emails. In this case, an email address with an associated inbox and key for sending and receiving messages had not been set up, causing TAP codes to be sent to the Exchange server but never delivered.

4.7.2 Intune privileges

The second issue worth mentioning is related to Intune privileges. This problem arose from a combination of permission settings and discrepancies in the versioning of the operating system on the target devices. The devices under focus for the Proof of Concept were tablets used by blue-collar workers in the production line. During the process of replicating the production environment in Azure, efforts were made to mirror the setup of the actual devices, including the operating system, policies, and hardware, as well as the software and apps present on the devices.

However, the version of Intune in use was a few minor releases behind, and the service packs installed on the test devices differed from those on the target devices. These inconsistencies led to constant failures in the installation and detection scripts of Component 2. After further investigation, it was determined that a PowerShell module being used was not permitted within the scope of the Intune user on the target devices. Like other software, Intune operates with its own set of permissions, and in this case, the PowerShell module required additional permissions that Intune was not granted. This issue was later resolved with updated versions of Intune, which allowed for broader use of PowerShell modules on the devices.

Chapter 5

Conclusions

This thesis marks the conclusion of an extensive journey into the world of passwordless authentication. As emphasized in the introduction and explored in depth in subsequent chapters, this topic represents a significant shift in the cybersecurity landscape. It is not merely introducing a new authentication method, but a fundamental rethinking of how authentication systems are designed. Despite technological advances, passwords have long been the dominant form of authentication and will likely remain common for years to come. Passwordless technologies are challenging this paradigm and will eventually overrule it.

In this work, we developed a comprehensive framework to facilitate the transition to passwordless technologies, alongside a detailed analysis of a real-world application. Both elements are crucial and contribute equally to the overall objective. The framework provides the necessary guidance and resources to address the challenges of implementing passwordless authentication, while the GripGotham case study offers valuable insights into the practical challenges that arise when applying new technologies. This dual approach benefits both theoretical understanding and practical application, verifying the framework on one side while facilitating a smooth transition away from password-based authentication on the other.

One key takeaway for readers of this thesis is the realization that transitioning to passwordless authentication is not a simple plug-and-play process. It requires comprehensive planning, resources, and strategic alignment with an organization's goals. The transition is not a one-size-fits-all solution, but it offers substantial advantages in terms of security and usability when approached methodically. However, this shift also introduces new risks and potential vulnerabilities that must be carefully considered. This thesis aims to serve as a practical guide for security professionals seeking to navigate these complexities and implement passwordless systems effectively.

Looking ahead, there is significant potential for further research and development in the field of passwordless authentication. Each section of this work could be expanded, and numerous topics within the scope of passwordless technology deserve deeper exploration. The human factors involved in adopting these systems, as well as the new attack vectors they may introduce, are areas for future investigation. This thesis represents only the beginning of a broader conversation, with many questions yet to be answered and more real-world cases needed to test and refine the framework.

From a personal perspective, working on this topic as both a student and an emerging security professional has been a rewarding experience. What began as a university project evolved into an internship, and ultimately into this thesis. The shift towards passwordless authentication is more than just a technical evolution; it is an important development in the future of cybersecurity. Being able to contribute to this growing field has been an invaluable opportunity, and I look forward to seeing how it continues to evolve in the years to come.

Bibliography

[1] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano, "Passwords and the Evolution of Imperfect Authentication", Communications of the ACM, vol. 58, July 2015, pp. 78–87, DOI https://doi.org/10.1145/2699390

[2] A. Nisenoff, M. Golla, M. Wei, J. Hainline, H. Szymanek, A. Braun, A. Hildebrandt, B. Christensen, and D. Langenberg, "A two-decade retrospective analysis of a universityâs vulnerability to attacks exploiting reused passwords", 32rd USENIX Security Symposium, Anaheim, CA, USA, August 9â11, 2023, pp. 5127 – 5144

[3] M. Abadi, T. M. A. Lomas, and R. Needham, "Strengthening Passwords", IEEE Computer Society Symposium on Research in Security and Privacy, December 1997

[4] S. Chaudhary, T. Schafeitel-TÃ¤htinen, M. Helenius, and E. Berki, "Usability, security and trust in password managers: A quest for user-centric properties and features", Computer science review, vol. 33, 2019, pp. 69–90, DOI https://doi.org/10.1016/j.cosrev.2019.03.002

[5] A.Karole, N.Saxena, and N.Christin, " A Comparative Usability Evaluation of Traditional Password Managers", Information Security and Cryptology, vol. 6829, December 2011, p. 233â251, DOI https://doi.org/10.1007/978-3-642-24209-0_16

[6] Lionel Sujay Vailshery, https://www.statista.com/statistics/1331322/password-management-market-share/

[7] M. Dell'Amico, P. Michiardi, and Y. Roudier, "Password strength: An empirical analysis", 2010 Proceedings IEEE INFOCOM, March 2010, DOI https://doi.org/10.1109/INFCOM.2010.5461951

[8] D. Pasquini, M. Cianfriglia, G. Ateniese, and M. Bernaschi, "Reducing bias in modeling real-world password strength via deep learning and dynamic dictionaries", 32th USENIX Security Symposium, Anaheim, CA, USA, August 11â13, 2021

[9] pasquini-dario, https://github.com/TheAdamProject/adams

[10] X. Cui, C. Li, Y. Qin, and Y. Ding, "A password strength evaluation algorithm based on sensitive personal information", 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), February 2020, pp. 1542–1545, DOI https://doi.org/10.1109/TrustCom50675.2020.00211

[11] J. Williamson and K. Curran, "The Role of Multi-factor Authentication for Modern Day Security", Semiconductor Science and Information Devices, vol. 3, April 2021, DOI https://doi.org/10.30564/ssid.v3i1.3152

[12] L. A. Meyer, S. Romero, G. Bertoli, T. Burt, A. Weinert, and J. L. Ferres, "How effective is multifactor authentication at deterring cyberattacks?", Conrnel University, May 2023, DOI https://doi.org/10.48550/arXiv.2305.00945

[13] Kurt Thomas and Angelika Moscicki, https://security.googleblog.com/2019/05/new-research-how-effective-is-basic.html

[14] European Union Agency for Cybersecurity, https://www.enisa.europa.eu/publications/enisa-threat-landscape-2022

[15] European Union Agency for Cybersecurity, https://www.enisa.europa.eu/publications/enisa-threat-landscape-2023

[16] Neil J. Rubenking, https://www.pcmag.com/news/has-multi-factor-authentication-failed-us

[17] V. Parmar, H. A. Sanghvi, R. H. Patel, and A. S. Pandya, "A comprehensive study on passwordless authentication", International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), Erode (India), April 07-09, 2022, DOI https://doi.org/10.1109/ICSCDS53736.2022.9760934

[18] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, " Zero Trust Architecture ", NIST Special Publication, vol. 800-207, August 2020, DOI https://doi.org/10.6028/NIST.SP.800-207
[19] PaloAltoNetworks, https://www.paloaltonetworks.com/cyberpedia/what-is-a-zero-trust-architecture
[20] N. Haller and C. Metz, "A One-Time Password System." RFC-2289, February 1998, DOI https://www.rfc-editor.org/rfc/rfc2289.html
[21] NHS, https://nhs-digital.zendesk.com/hc/en-gb/articles/4420385199761-One-Time-Passcodes-SMS-security-codes
[22] Adobe, https://helpx.adobe.com/sign/config/send-settings/auth-methods/one-time-password-via-email.html
[23] J. Kubovy, C. Huber, and M. JÃ¤ger, " A Secure Token-Based Communication for Authentication and Authorization Servers ", Future Data and Security Engineering, vol. 10018, October 2016, DOI https://doi.org/10.1007/978-3-319-48057-2_17
[24] H. POLAT and S. OYUCU, " Token-based authentication method for M2M platforms ", Turkish Journal of Electrical Engineering and Computer Sciences, vol. 25, January 2017, DOI https://doi.org/10.3906/elk-1608-6
[25] Ashley Stevenson, https://www.pingidentity.com/en/resources/blog/post/what-is-magic-link-login.html#What-is-Magic-Link-Authentication
[26] I. Matiushin and V. Korkhov, " Passwordless authentication using magic link technology ", Distributed Computing and Grid Technologies in Science and Education", vol. 3031, July 2021
[27] A. C. Weaver, "Biometric authentication", IEEE Computer, vol. 39, February 2006, pp. 96–97, DOI https://doi.org/10.1109/MC.2006.47
[28] J. Wayman, A. Jain, D. Maltoni, and D. Maio, "An Introduction to Biometric Authentication Systems", Biometric Systems, vol. 14, no. 370, 2005, DOI https://doi.org/10.1007/1-84628-064-8_1
[29] D. Bhattacharyya1, R. Ranjan, F. Alisherov, and M. Choi, "Biometric Authentication: A Review", International Journal of u- and e- Service, Science and Technology, vol. 2, September 2009
[30] D. M. Reddy, K. P. Akshay, R. Giridhar, S. D. Karan, and N. Mohankumar, "Bharks: Built-in hardware authentication using random key sequence", 2017 4th International Conference on Signal Processing, Computing and Control (ISPCC), September 2017, DOI https://doi.org/10.1109/ISPCC.2017.8269675
[31] J. Lang, A. Czeskis, D. Balfanz, M. Schilder, and S. Srinivas, "Security keys: Practical cryptographic second factors for the modern web", Financial Cryptography and Data Security), vol. 9603, September 2017, DOI https://doi.org/10.1007/978-3-662-54970-4_25
[32] FIDO Alliance , https://fidoalliance.org/specifications/
[33] Q. Xu, R. Zheng, W. Saad, and Z. Han, "Device fingerprinting in wireless networks: Challenges and opportunities", IEEE Communications Surveys Tutorials, vol. 18, no. 1, 2016, pp. 94–104, DOI https://doi.org/10.1109/COMST.2015.2476338
[34] Polaris market research, https://www.polarismarketresearch.com/industry-analysis/passwordless-authentication-market
[35] Fortune Business Insights , https://www.fortunebusinessinsights.com/passwordless-authentication-market-109838
[36] NIST Cybersecurity Framework, https://www.nist.gov/cyberframework/identify
[37] CIS Center for Internet Security, https://www.cisecurity.org/controls/access-control-management
[38] CISA, https://www.cisa.gov/