Raising security level in Cloud by using 2FA TOTP (Time-based One-time Password) and SSO (Single Sign On)

Submitted By

Raval Abhishek A

20MCEI15

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2022

Raising security level in Cloud by using 2FA TOTP (Time-based One-time Password) and SSO (Single Sign On)

Major Project

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology in Computer Science and Engineering

(Information and Network Security)

Submitted By

Raval Abhishek A

(20MCEI15)

Guided By

Dr Madhuri Bhavsar

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2022

Certificate

This is to certify that the major project entitled **"Raising security level in Cloud by using 2FA TOTP (Time-based One-time Password) and SSO (Single Sign On)"** submitted by **Raval Abhishek A (20MCEI15)**, towards the partial fulfillment of the requirements for the award of degree of Master of Technology in Computer Science and Engineering of Nirma University, Ahmedabad, is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this Major Project Part-II, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

[illegible]

Dr Madhuri Bhavsar
Internal Guide & Professor & Head
CSE Department
Institute of Technology
Nirma University, Ahmedabad

[illegible]

Dr Sharada Valiveti
Professor & PG Coordinator (M.Tech-INS)
CSE Department
Institute of Technology
Nirma University, Ahmedabad

[illegible]

Dr. Rajesh N Patel
Director
Institute of Technology
Nirma University, Ahmedabad

Statement of Originality

I, **Raval Abhishek A**, **20MCEI15**, give undertaking that the Major Project entitled **"Raising security level in Cloud by using 2FA TOTP (Time-based One-time Password) and SSO (Single Sign On)"** submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in **Computer Science & Engineering (INS)** of Institute of Technology, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made.It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

[illegible]

Signature of Student

Date:

Place:

[illegible]

Endorsed by

Dr Madhuri Bhavsar

(Signature of Guide)

Acknowledgements

It gives me immense pleasure in expressing thanks and profound gratitude to **Dr Madhuri Bhavsar**, Professor & **Dr Vivek Kumar Prasad**, Assistant Professor at Computer Science and Engineering Department, Institute of Technology, Nirma University, Ahmedabad for his valuable guidance and continual encouragement throughout this work. The appreciation and continual support he has imparted has been a great motivation to me in reaching a higher goal. His guidance has triggered and nourished my intellectual maturity that I will benefit from, for a long time to come.

It gives me an immense pleasure to thank **Dr Madhuri Bhavsar**, Hon'ble Head of Computer Science And Engineering Department, Institute of Technology, Nirma University, Ahmedabad for her kind support and providing basic infrastructure and healthy research environment.

A special thank you is expressed wholeheartedly to **Dr.Rajesh N Patel**, Hon'ble Director, Institute of Technology, Nirma University, Ahmedabad for the unmentionable motivation she has extended throughout course of this work.

I would also thank the Institution, all faculty members of Computer Science and Engineering Department, Nirma University, Ahmedabad for their special attention and suggestions towards the project work.

- Raval Abhishek A

20MCEI15

Abstract

In recent times, there are rising concerns over cloud security principles. Applications upload massive data on cloud through provided service sets, platforms, and infrastructures. The private and sensitive data needs to be protected. Recent studies have suggested multi-factor authentication techniques to secure against possible security attacks on cloud systems. Most techniques involve a single factor authentication, but recent attack perimeters have intensified. Thus, two factor authentication (2TP) becomes important in cloud security. In this paper, we propose a 2TP based time based one time password (TOTP), that provides high degree of secrecy. A single sign on (SSO) scheme is integrated with 2TP TOTP as a dual layer of security, and provides users with the flexibility to authenticate multiple websites and credentials with a single sign. This improves the overall scalability of end-user applications, that mitigates the overhead of 2TP TOTP. Overall, the scheme provides strong security, while maintaining the flexibility of user authentication over heterogeneous cloud applications.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

In this chapter we discuss about the basics of Cloud computing and how it plays roles in todays world. The chapter discusses in brief about Cloud computing, Single Sign-on and 2FA using TOTP.

1.1 Cloud Computing

Cloud computing is a type of computing that uses the It's a software framework that allows one or more data centers to pools large amount of storage, networking resources and computation to manage. In this environment, users have access to shared computer System resources through an Internet connection from anywhere, at any time. Many CSP such as such as Amazon Web Services S3, Google Cloud Storage, and Microsoft Azure Storage, and many more. A company that provides cloud computing services is referred as a cloud service provider(CSP).

Cloud computing refers to a sophisticated infrastructure, hardware, computing, and memory that is made accessible as a pay-as-you-go service over the internet. Cloud security is extremely significant since it is one of the most cutting-edge technologies and security techniques for safeguarding our data. [6] We can allocate resources to multiple levels of use, such as public, private, or hybrid.

1. Public - The files which we want to share among all users can be placed as public which is cost effective too.
2. Private - The files which are highly confidential in the cloud platform can be placed

as private on the secure servers.

3. Hybrid - The files which are confidential as well as shareable among several user can be placed in hybrid category.

1.2 Single Sign-On

In the present digital world single, sign-on is the technology that is widely used among service providers.[21] This technology facilitates users with more flexible authentication. This technique provides benefits to both sides: the user side service provider side. [28] From the user perspective user does not need to memorize different usernames or password combinations. Users can access the different applications with a single username and password. While service provider needs to manage a single credential per user instead of multiple credentials per user. So, administration becomes simple.

SSO refers to the ability to regulate access to many related but separate software systems. [9] This attribute allows users to log in once and have access to every system without having to login again.[23]

1.2.1 Types of SSO

1. **F-SSO (Federated SSO)**
   SSO is a part of federated identity manegement. So, sso also referred as federated SSO. One feature is often available with FIM architecture is SSO.

2. **OAuth** -
   FIM architecture provides the OAuth 2.0 framework. It focuses on the trusted connection that allows user identification data to be transferred between domains.[17].

3. **OIDC (OpenID Connect)** -
   OIDC is one of OAuth 2.0's layers. SSO is allowed as a result of this.

4. **Security access markupp language (SAML)**
   SMAL[30] also providing SSO functionality but it's an open standard under FIM.

5. **Same Sign-On (SSO)** -
   Same Sign-on is also abbreviated like SSO. However, it is different from SSO. There

is no trust connection between entities while using same sign-on. It is more depended on credential being duplicated. While comparing security between single sign-on [11] and same sign-on, same sign on is less secure.

1.2.2 Advantages of SSO

1. **It boosts both employee and IT output.**

   A single point of access will save resources and time. With single sign-on, you may perform the following: Reduce the number of service calls: Customers who have a single password for all applications will require less assistance. Boost the customer's experience: Since they don't have to navigate between several logon Pages or reset credentials, users save 5 to 10 seconds while logging in. Users may securely utilize their SSO credentials on any system and in any browser, reducing the risk of data breaches. One of its most important benefits of SSO is the ability to increase end-user productivity. [34]

2. **It enhances security capacities.**

   One of its most popular misconceptions about SSO affects security. The assumption is that when a master password is stolen, it puts all related accounts in jeopardy. However, because they only have to memorize a single password for multiple applications, users tend to create a tougher (extremely hard) passcode and are less willing to write everything down. If you execute these best practices, password theft will be less likely.

3. **It makes it easier to remember passwords.**

   To avoid cybercrime, security experts advise using different and strong passwords for each program. As a reason, the average user must memorize numbers of credentials for individuals as well as organizations. Unfortunately, as a result of this, "password fatigue" is frequent. What are the commercial ramifications of password fatigue? To put it another way, the more credentials you have, more the problems you'll encounter. Customers being unable to log in will leave your website or app until you can engage them. This is supported by a recent Baymard Institute usability research. Baymard evaluated that 2 e-commerce sites (Amazon and ASOS) discovered that 18.75 percent of existing customers abandon their baskets due to missed passwords or password reset troubles in this study.[3]

4. **It improves the user's experience.** User experience is one of the most significant advantages of SSO. Customers can now enjoy an advanced technological experience without logging in multiple times in multiple applications, thanks to [29]. Customers will become more committed, and engagement levels will be higher, resulting in increased profits for enterprises.

5. **It keeps Shadow IT at bay.**
   Shadow IT is nothing new in the world of cybersecurity. It refers to illegal downloading in the workplace. When it comes to Shadow IT, employees were formerly limited to purchasing the software through office supply stores. However, as cloud-based downloads grow more common, the risk of a data breach rises. To address this issue, IT administrators can utilize SSO to track what applications employees use. As a result, identity theft is no longer a threat. To ensure that domestic and global compliance regulations are met, a company's IT or compliance team may use a single platform.

1.2.3 Disadvantages of SSO

1. **At scale, it's the most expensive/best option**
   Simply put, SSO may quickly become prohibitively costly. SSO can bring significant benefits for small businesses, but it can also be a significant financial burden. Many SSO suppliers charge per feature, and because the majority of the basic capabilities are add-ons, the fees can soon pile up.

2. **It's necessary to have an IdP**
   An firm's IdP/directory service is the foundation of any SSO system. SSO services are often built on top of a directories, requiring enterprises to pay for multiple solutions in order to obtain the desired result. Of course, like SSO, this may be costly for businesses, both in terms of the initial setup and installation expenses as well as the ongoing costs of utilising it.

3. **SSO support majority of web-based app**

   An firm's IdP/directory service is the foundation of any SSO system. SSO services are often built on top of a directories, requiring enterprises to pay for multiple solutions in order to obtain the desired result. Of course, like SSO, this may be costly for businesses, both in terms of the initial setup and installation expenses as well as the ongoing costs of utilising it.

4. **Extra-strong credentials are required**

   Though end users only had to memorize one password for SSO, it's better if it's long, difficult, and secure. Although this is typically beneficial to identity security, it also increases the risk of a person losing or compromising their password, negating SSO advantages negating the benefits of SSO.

5. **If SSO service provider is compromised, all resources tied to it are vulnerable to assaults**

   Because SSO is tied to so many key services, if an SSO service provider is attacked, whole user communities are at risk. If an individual user's SSO portal is hacked, their accessibility to those apps is likewise jeopardised unless MFA is used.

6. **SSO necessitates setup and configuration.**

   SSO, like so many IT products, is seldom "plug-and-play," which means IT administrators must devote the extra effort which is necessary to integrate and customize their SSO services to their company. Not only do apps have to be deployed, but integrating them with a third-party IdP may be difficult and time-consuming.

1.3 Time Based One-time Password

In today's world, the majority of transactions are conducted digitally. Banking, retail orders, transportation, and other transactions taking place over digital media have become a natural reality, posing significant authentication and authorization security problems to the enterprises that provide these services. [25] Consumers want operations and transactions to be more secure and their data to be kept private. Since then, OTP has entered the picture. It has been utilized in operations for authentication. OTP has grown rapidly to the point that it is now used as an extra authentication level in practically all applications.

A one-time password is just a pair of keys and values generated each time a user logs in or requests a transaction that requires proper user authentication to complete. Only digital technology uses OTPs, which are only valid for a single transaction or a single login session. OTPs are utilized in a variety of applications, as well as in two-factor authentication. It can guarantee that the machine's resources aren't acquired illegally by any unauthorized users.

The benefit of OTP over static passwords is that it generates dynamic passwords that cannot be utilized in replay attacks. Even if hackers get an OTP that has already been used for one transaction, which is no longer usable for further transactions. They won't be able to use it again because it is no longer a valid OTP. OTPs have become a requirement for practically all transactions conducted over the internet or through digital media.

1.3.1 Types of OTPs

1. **HOTP**

   HOTP means Hash-based Message Authentication Code, The HMAC-based One-time Password algorithm (HOTP) is an event-based OTP with a counter as the moving factor in each code, in layman's terms. The movement factor is increased every time the HOTP is requested and validated, depending on a counter. Generated code is valid till you actively demand another one and the authentication server validates it. Every time the code is validated and the user is given permission, the OTP generator and the server are synchronized. An example of a HOTP is Yubiko's Yubikey.

2. **TOTP**

   TOTP is a time-based OTP. TOTP uses a static seed, similar to HOTP, but the movement factor is time-based instead of counter-based. A timestep refers to the length of time that each password is valid. Timesteps are usually 30 to 60 seconds in length. If you're not using your OTP within the same time frame, it will expire, and to access your proposal, you'll need to acquire a new one.

1.3.2 Advantages of TOTP

1. **Cost Efficient**

   TOTP 2FA is frequently used by businesses because of its ease of usage. [18] Most authentication programs that produce TOTP tokens are free or require a nominal price, allowing businesses of any size to protect their users' identities.

2. **Lightweightt**

   For consumers to authenticate to their IT resources, organizations do not need to deploy any additional hardware. All consumers need is a PC, laptop, or phone with an authenticating app. Most TOTP app providers support 2FA for all of those devices, allowing customers to choose whatever method best meets their needs.

3. **Can be utilised on a large basis**

   Organizations may enforce TOTP 2-factor authentication at mass across all of their IT infrastructure with the proper providers. Multicultural systems, a diverse set of applications, networks, and file servers are all part of this.

4. **It's possible to utilize it as a soft token**

   A TOTP authentication mechanism may be found in both special hardware tokens and software, most commonly in the form of a mobile app like Google Authenticator. You may eliminate the expenses of hardware manufacture, distribution, stock, and managed by implementing it in the program (also referred as a software token). Transmit Protection, for example, allows you to trademark your own TOTP authentication scheme and add additional levels of security.

1.3.3 Disadvantages of TOTP

1. **A customer's device is required**

   A user will not be able to obtain their TOTP code till they have their authenticator application open. They are unable to utilize their IT facilities if they leave their cellphone or if the power on their gadget dies. Many web - based applications, however, include alternate methods for receiving 2 factor codes, which users may utilise if they are unable to acquire their TOTP key from an any of authenticator application.

2. **Expiration time is short**

   This may need a user entering many TOTP codes in order to log in before code validity gets expires, which consumes time and may result in account lockout if they surpass their allocated tries.

3. **Key that is kept hidden**

   The identifier app & server serving it share a private key in TOTP 2FA. A bad actor who cloned the secret key might produce valid codes at will and get access to the user's account.

Chapter 2

Literature Survey

2.1 Literature Summary

The titled named ”Implementation Of Two-factor Authentication Access Control In Web-based Services With Cloud Computing Using C.net” was published by Gokila R Manimaran A in the year 2018. they have worked on security achievement and privacy for web-based cloud services. They included 2FA for both user secret keys and security for device access control systems for web-based cloud computing services. The implementation of a two-factor authentication (two-factor) access control system has been recognized as a way to enable the cloud server while simultaneously restricting access to users who have the same set of traits and protecting user privacy. [27]

Sergey Babkin Anna Epishkina has publish a paper on ”Authenticatin Prtcls Based n ne-Time Passwrds” in 2019 where they studied The study of authentication methods based on OTPs was described in this publication. Their benefits, drawbacks, and weaknesses were examined. The study found that none of the authentication methods examined was robust to a masquerade attack. In future research on this topic, we expect to find this vulnerability in one or more examined protocols by employing unique strategies to protect against masquerade attacks.[8]

Sangamesh S M S S Joshi published a paper named ”A Survey on A Secure Cloud Storage System: An Approach” in 2018. In which they offer a novel secure cloud storage method to protect organizations’ The suggested system uses two authentication techniques: (TOTP) and (ABP) to boost the authentication level of security. In the proposed solution, data owners have complete control over who has access to their outsourced data on cloud storage servers. User authentication is verified using 2FA to boost security: the first is performed with a username and password, while the second is triggered by the usage of TOTP.[31]

Titled named "Preserving Privacy in Cloud-based applications using two-factor authentication (TOTP/WTP)" was published by Pranayanath Reddy Anantula Dr. G Manoj Someswa in the year 2016. where they have discussed the most difficulty in the Cloud environment is ensuring the privacy of users' and organizations' data. By employing TOTP/ WTP in Salesforce.com, users' privacy is protected by validating whether they are valid users regularly. When the session time exceeds a predetermined limit defined by the administrator, an OTP is delivered to the user, who must validate himself/herself by inputting the text/number sent by the OTP server. This ensures that the cloud application is being used by a legitimate user. Similarly, WTP protects regular users' privacy and eliminates the need for several TOTP while utilizing the cloud system.[6]

D. Divya Priya A. Mahalakshmi has discussed Researchers have published a methodology to generate TOTP by mixing a hash with a secret key for a certain period using the HMAC technique. It ensures the integrity and validity of data. All network encryption protocols make use of it. You can go even farther with this solution by creating QR codes for mobile devices to scan, making it as simple as downloading a mobile app, scanning the code, and utilizing the tokens as needed. In 2018 with a paper named "Data Security in Mobile Cloud Computing Using TOTP Generated By HMAC-SHA1 Algorithm".[26]

Amol Sure, Shrikant Udmale, Vaibhav Doifode, and Amol Waghade have published a paper named "Applying a Single Sign-On Algorithm Based On Cloud Computing Concepts for SaaS Applications Using MD5 Encryption" where they have proposed a system to improve the performance and dependability of model, they have used encryption technique. where A cloud-based application with two independent cloud servers was provided in the suggested concept to apply For accessing cloud computing environments, a safe and efficient SSO mechanism is required. Furthermore, there is some safety. Secure Socket

Layer (SSL) and MD5 cryptography are examples of tools and techniques used in transmission operations.[34]

Paper named "An Optimized Single Sign-On Schema for Reliable Multi-Level Security Management in Clouds" where they have discussed Cloud computing which is a widely established technology in both industry and academia because of its irreplaceable features. Some of the benefits of this trend include on-demand services, flexible provisioning, and a large range of functionality. The ability to employ a variety of services from different sources encourages businesses and people to follow this trend. However, this flexibility also comes with security risks, particularly in the area of permission and authentication. A flexible and safe access control mechanism is provided in this work, which supports the security-level-based authorization methodology. While the attribute mapping methodology simplifies access control by classifying a wide variety of characteristics and authentication methods (e.g., PKI, FIDO U2F, LoA, and so on), the multi-factor authentication architecture used helps to maintain security consistency.[9]

Paper named "SSO (Single Sign-On) Implementation" by Parul Garg, Dr. Yashpal Singh in the year 2013 where they have discussed a case study of a project to provide a Single Sign-On (SSO) solution for web-based applications that use the mainframe for data storage. This is a procedure that allows an authorized user to access numerous applications by entering their credentials. In contrast, the system's multiple access may be ended by a single sign-out operation. The article starts with a detailed explanation of the high-level business needs. [15]

Chapter 3

Working Functionality of SSO TOTPs

As most the web application follows browser/server or client/server architecture, with HTTP as the communication protocol. HTTP is a protocol that has no state. The server handles each browser request separately and does not correlate it with previous or subsequent requests. However, it also implies that any user may use the browser to access the server's resources. You must restrict browser requests if you wish to safeguard some server resources; to restrict browser requests, you must authenticate the browser request, reply to valid requests, and reject illegal requests; to authenticate a browser request, you must be aware of the browser request status. So we use mechanisms like 'Cookies,', 'JWT', 'Sessions,' to allow the server and browser to retain a shared state.

When we have a single system, maintaining the state mechanism via login authentication is simple. However, how do we keep the state of each system when a single system expands into several systems?

3.1 Functionality of SSO in Cloud

SSO operates on the basis that you can log in to one system in a multi-system application group and be authorized in all other systems without logging in again, including single sign-on and single sign-off.[33] We have a single independent authentication server at the heart of SSO [12], which may receive sensitive information such as the user's identity in form of username and password. Other systems don't allow users to log in and instead rely on the authentication server's indirect authorization. The token is used to achieve indirect authorization. So we will look into functionality performed by SSO Server and SSO Client in implementation.

SSO Server

- Check the user's login credentials.
- Create a global session to work with.
- Create a token for authorization.
- With sso-client communication, token is sent.
- Validate the sso-client token.
- Send a JWT including the user's data.

SSO Client

- The sso-client subsystem does not log in to the user request and instead requests authentication from the sso server.
- The sso authentication server will send you a token.
- To check the token's authenticity, communicate with sso-server.
- Receives a JWT and uses the public key to verify it.
- Create a local session.

3.2 Functionality of 2FA-TOTP in Cloud

Two-factor authentication, commonly known as TOTP, adds an extra degree of security to the standard by asking the user to provide a unique code in a second field. The length of the key is six characters, and it changes every 30 seconds.

At least two factors are required for two factor authentication:

- The user's username and the password's **knowledge factor**

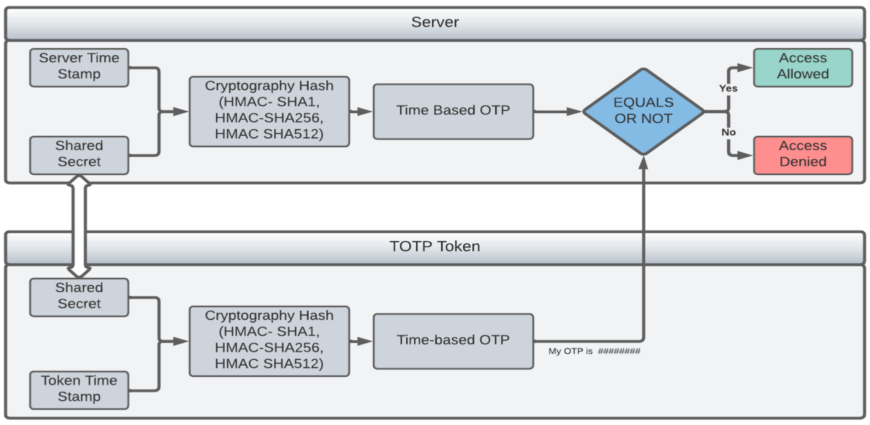


Figure 3.1: 2FA-Working flow of TOTP

- **Possession** includes a mobile device for producing the TOTP code. The code can be created via a specialised programme like Google Authenticator which can received as an SMS on the mobile device or on a app.

TOTP (Time-based One-time Password) is a one-time password technique that derives its uniqueness from the current time.[22] The TOTP algorithm is a time-based variation of the HOTP algorithm that is computed using the same function, except that the counter is substituted by the TI value, which is determined from the timeline in this case. To generate a one-time password, you can use the HMAC SHA 512 / HMAC SHA 256 algorithms instead of the HMAC-SHA-1 technique. TOTP is defined as follows:

$$\text{TOTP} = \text{HOTP}(K, TI) \tag{3.1}$$

The number of time steps between the initial time counter Ti0 and the current UNIX time is represented by TI. TI more precisely defined as:

$$\text{TI} = \text{CurrentUNIXTi} - \text{Ti0})/A \tag{3.2}$$

Here, A is a system parameter that determines the quantity of time time steps in seconds (30 seconds is the default value), and Ti0 is the UNIX timestamp at which we begin calculating time steps (the default value is 0). When a one-time password (OTP) is received, the authentication server has no idea when it was generated.

The duration gap (amount of times steps from time Ti0 onwards) between OTP was produced and when the verification system gets the OTP may be too big due to connection difficulties. As a result, the verification system should normally define an appropriate OTP transmit timeout policy. The longer this period is, the more vulnerable it is to a possible assault. It is strongly advised that the time delay be shorter than size of a single step.[14]

3.2.1 Steps needed to follow to achieve TOTP Authentication

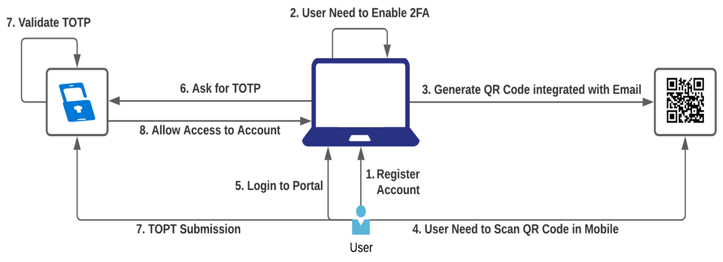

7. Validate TOTP
2. User Need to Enable 2FA
6. Ask for TOTP
8. Allow Access to Account
3. Generate QR Code integrated with Email
5. Login to Portal
1. Register Account
7. TOPT Submission
4. User Need to Scan QR Code in Mobile
User

Figure 3.2: 2FA-TOTP

1. User need to register at registration portal by submitting email & password.
2. After successfully registration user need to enable two factor authentication.
3. QR code is generated by automated system integrated with email id.
4. User need to scan QR Code in cell phone using Google Authenticator application. after scanning a 6 digit code needed to submit on same page to authentication.
5. Now after enabling 2FA user can not login with just email & password.
6. User need to submit username & password to login portal then after portal will ask for TOTP which is available on Google authenticator app which changes every 30 seconds.
7. Check for validation of submitted TOTP.
8. If TOTP is valid System will allow access to account and if not then it raise a error with invalid TOTP.

Chapter 4

Proposed Architecture for Raising Security Level in Cloud

4.1 Architecture

Our Proposed system is shown in the below diagram. This model is designed in such a way that it uses SSO (Single Sign-On) and TOTP (Time-based OTP) for 2FA Two factor authentication [20] in the cloud to enhance security for the cloud. This system is divided into different steps. This model mainly focuses on different applications published on the same cloud service provider.

In the current scenario user sign in with a different username and password in a different application. So, in this model, we have used SSO. So that we can use the same username and password for different applications on the same cloud to log in. An authentication server is used with SSO service which helps the user to authenticate. So, in the initial steps users tries to log in with their SSO credentials. When they try to log in, the request is redirected to the authentication server once a request is granted by the authentication server it will help the user to redirect to the 2FA page. If a request is discarded by the authentication server user will redirect to the same SSO login page with an error message. Once a request is granted then the user needs to submit TOTP as a part of 2FA which adds one more level of security.

Here, in this proposed system, we have used google authenticator to achieve 2FA using TOTP. Once the user enters TOTP on the 2FA page request is sent to the google

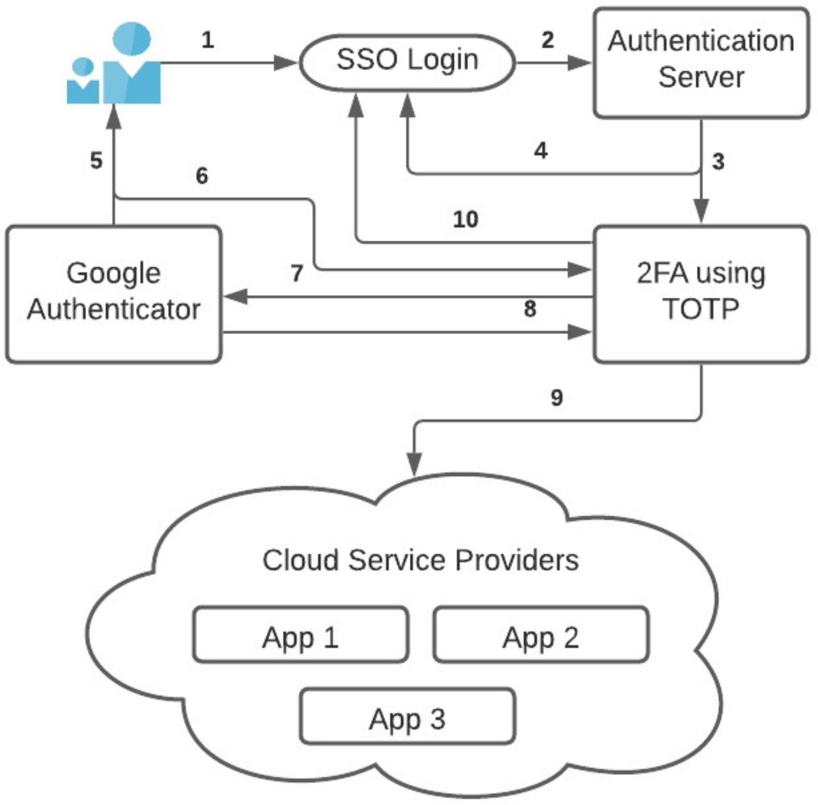


Figure 4.1: Proposed Architecture

authenticator to check TOTP entered by the user is valid or not. If OTP entered by use is validated with google authenticator, then the user will have direct access to applications. If TOTP is not validated by GA then the user will redirect to the SSO login page. Our main aim to use 2FA using TOTP is to increase one extra level of security in the cloud. Suppose, Once SSO credentials are compromised attackers will not have access to user-sensitive data stored on the cloud.

4.2 Flowchart

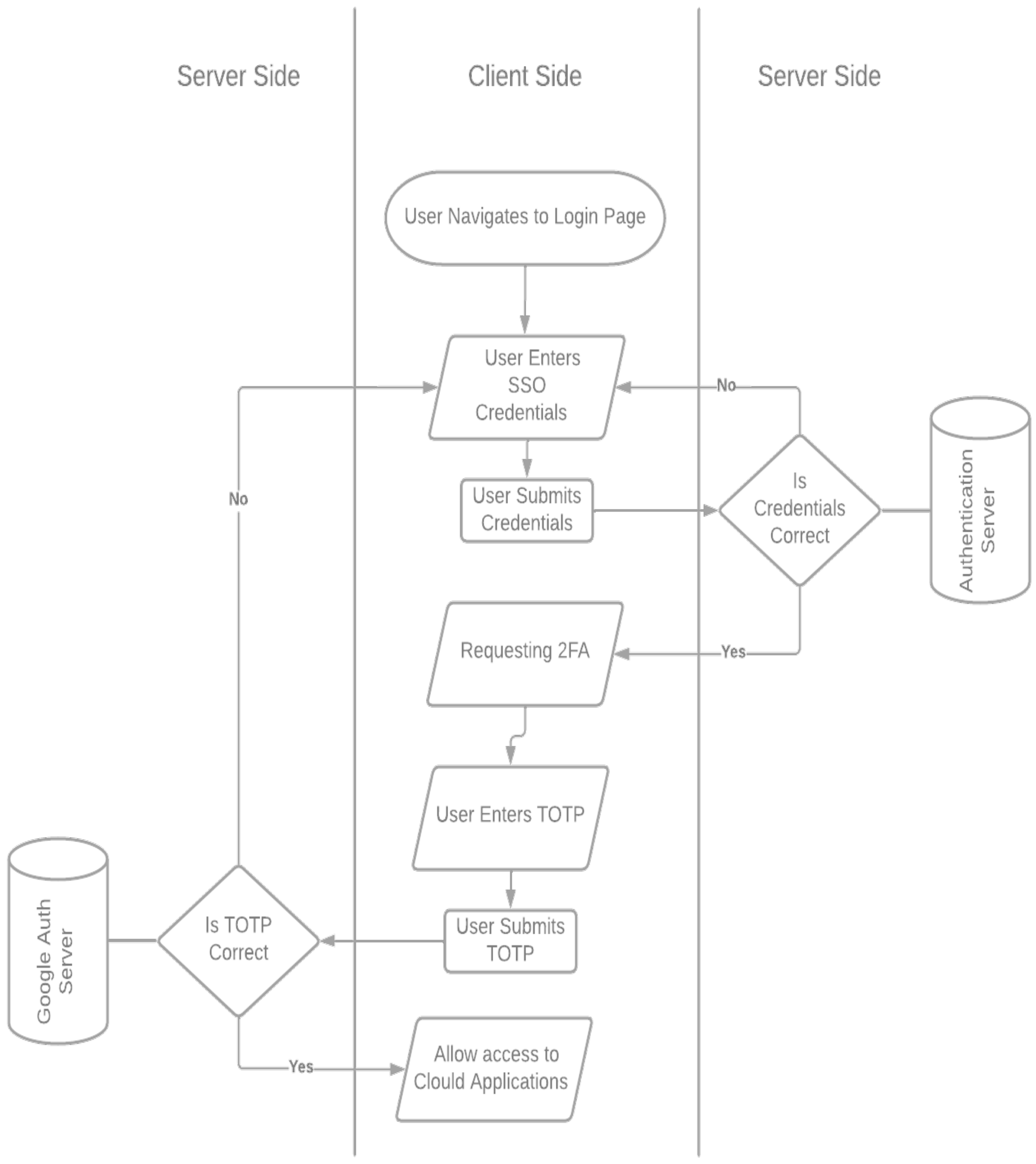
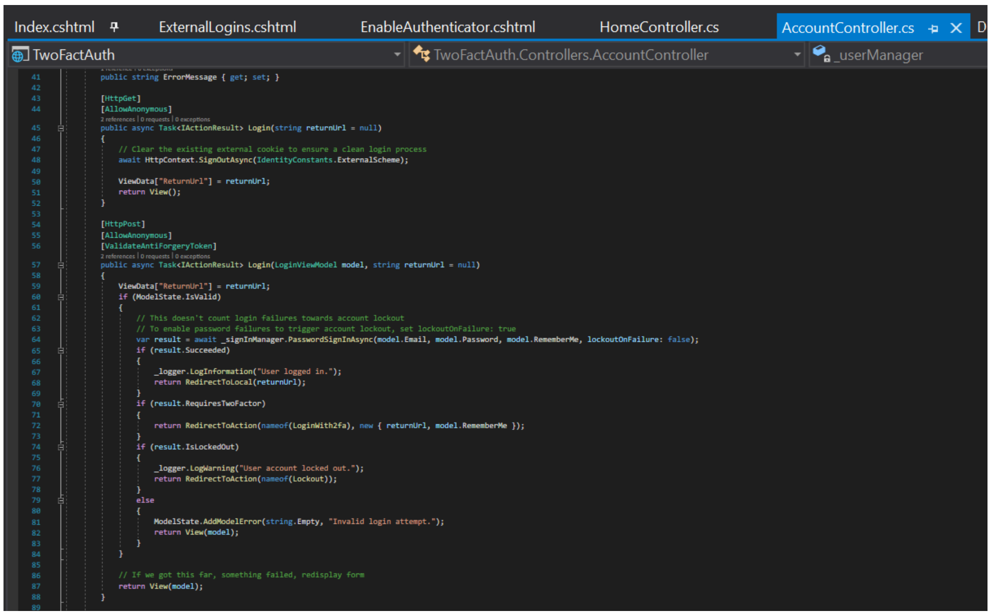


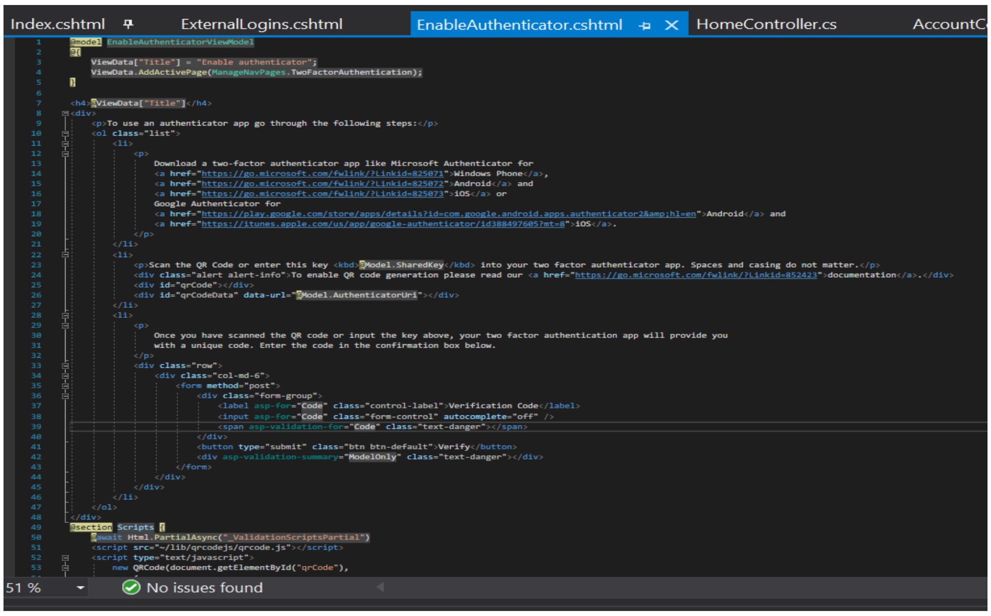
Figure 4.2: Flowchart

4.3 Implementation Snapshots



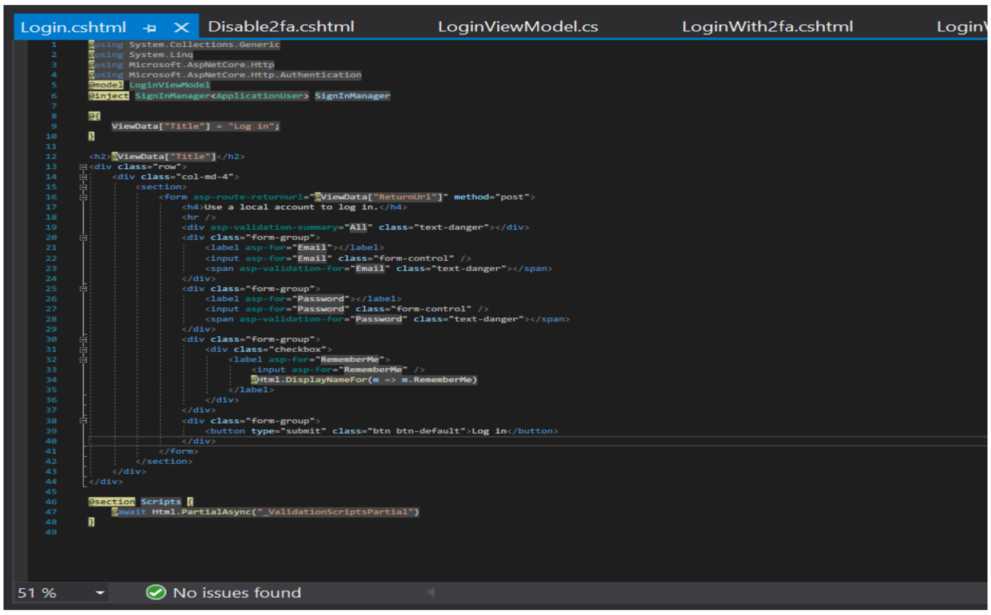
```
        public string ErrorMessage { get; set; }

        [HttpGet]
        [AllowAnonymous]
        2 references | 0 requests | 0 exceptions
        public async Task<IActionResult> Login(string returnUrl = null)
        {
            // Clear the existing external cookie to ensure a clean login process
            await HttpContext.SignOutAsync(IdentityConstants.ExternalScheme);

            ViewData["ReturnUrl"] = returnUrl;
            return View();
        }

        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        2 references | 0 requests | 0 exceptions
        public async Task<IActionResult> Login(LoginViewModel model, string returnUrl = null)
        {
            ViewData["ReturnUrl"] = returnUrl;
            if (ModelState.IsValid)
            {
                // This doesn't count login failures towards account lockout
                // To enable password failures to trigger account lockout, set lockoutOnFailure: true
                var result = await _signInManager.PasswordSignInAsync(model.Email, model.Password, model.RememberMe, lockoutOnFailure: false);
                if (result.Succeeded)
                {
                    _logger.LogInformation("User logged in.");
                    return RedirectToLocal(returnUrl);
                }
                if (result.RequiresTwoFactor)
                {
                    return RedirectToAction(nameof(LoginWith2fa), new { returnUrl, model.RememberMe });
                }
                if (result.IsLockedOut)
                {
                    _logger.LogWarning("User account locked out.");
                    return RedirectToAction(nameof(Lockout));
                }
                else
                {
                    ModelState.AddModelError(string.Empty, "Invalid login attempt.");
                    return View(model);
                }
            }

            // If we got this far, something failed, redisplay form
            return View(model);
        }
```

Figure 4.3: Coding Snapshot



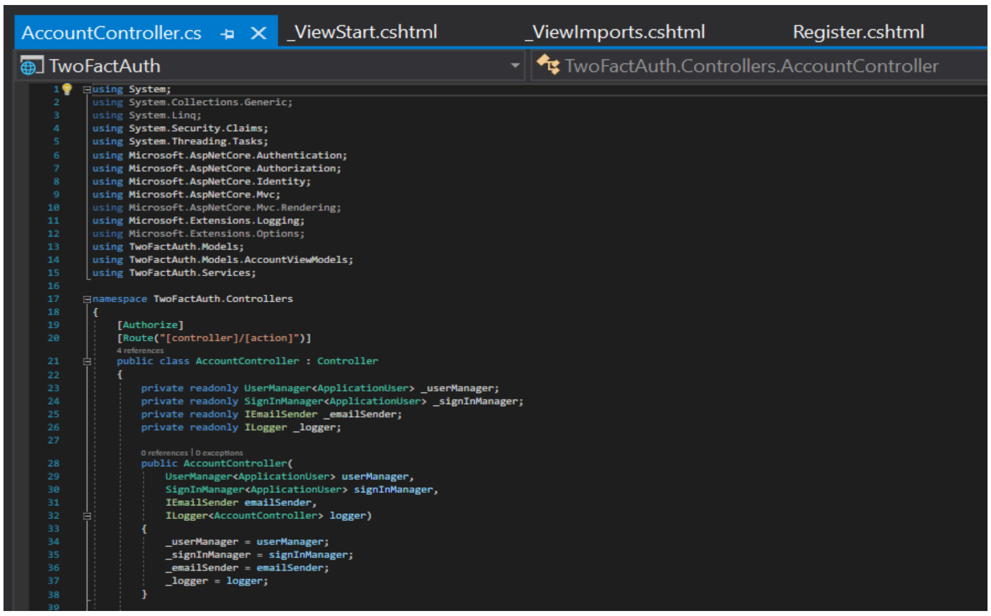
```
@model EnableAuthenticatorViewModel
@{
    ViewData["Title"] = "Enable authenticator";
    ViewData.AddActivePage(ManageNavPages.TwoFactorAuthentication);
}

<h4>@ViewData["Title"]</h4>
<div>
    <p>To use an authenticator app go through the following steps:</p>
    <ol class="list">
        <li>
            <p>
                Download a two-factor authenticator app like Microsoft Authenticator for
                <a href="https://go.microsoft.com/fwlink/?Linkid=825071">Windows Phone</a>,
                <a href="https://go.microsoft.com/fwlink/?Linkid=825072">Android</a> and
                <a href="https://go.microsoft.com/fwlink/?Linkid=825073">iOS</a> or
                Google Authenticator for
                <a href="https://play.google.com/store/apps/details?id=com.google.android.apps.authenticator2&amp;hl=en">Android</a> and
                <a href="https://itunes.apple.com/us/app/google-authenticator/id388497605?mt=8">iOS</a>.
            </p>
        </li>
        <li>
            <p>Scan the QR Code or enter this key <kbd>@Model.SharedKey</kbd> into your two factor authenticator app. Spaces and casing do not matter.</p>
            <div class="alert alert-info">To enable QR code generation please read our <a href="https://go.microsoft.com/fwlink/?Linkid=852423">documentation</a>.</div>
            <div id="qrCode"></div>
            <div id="qrCodeData" data-url="@Model.AuthenticatorUri"></div>
        </li>
        <li>
            <p>
                Once you have scanned the QR code or input the key above, your two factor authentication app will provide you
                with a unique code. Enter the code in the confirmation box below.
            </p>
            <div class="row">
                <div class="col-md-6">
                    <form method="post">
                        <div class="form-group">
                            <label asp-for="Code" class="control-label">Verification Code</label>
                            <input asp-for="Code" class="form-control" autocomplete="off" />
                            <span asp-validation-for="Code" class="text-danger"></span>
                        </div>
                        <button type="submit" class="btn btn-default">Verify</button>
                        <div asp-validation-summary="ModelOnly" class="text-danger"></div>
                    </form>
                </div>
            </div>
        </li>
    </ol>
</div>
@section Scripts {
    @await Html.PartialAsync("_ValidationScriptsPartial")
    <script src="~/lib/qrcodejs/qrcode.js"></script>
    <script type="text/javascript">
        new QRCode(document.getElementById("qrCode"),
```

Figure 4.4: Coding Snapshot



```
@using System.Collections.Generic
@using System.Linq
@using Microsoft.AspNetCore.Http
@using Microsoft.AspNetCore.Http.Authentication
@model LoginViewModel
@inject SignInManager<ApplicationUser> SignInManager

@{
    ViewData["Title"] = "Log in";
}

<h2>@ViewData["Title"]</h2>
<div class="row">
    <div class="col-md-4">
        <section>
            <form asp-route-returnurl="@ViewData["ReturnUrl"]" method="post">
                <h4>Use a local account to log in.</h4>
                <hr />
                <div asp-validation-summary="All" class="text-danger"></div>
                <div class="form-group">
                    <label asp-for="Email"></label>
                    <input asp-for="Email" class="form-control" />
                    <span asp-validation-for="Email" class="text-danger"></span>
                </div>
                <div class="form-group">
                    <label asp-for="Password"></label>
                    <input asp-for="Password" class="form-control" />
                    <span asp-validation-for="Password" class="text-danger"></span>
                </div>
                <div class="form-group">
                    <div class="checkbox">
                        <label asp-for="RememberMe">
                            <input asp-for="RememberMe" />
                            @Html.DisplayNameFor(m => m.RememberMe)
                        </label>
                    </div>
                </div>
                <div class="form-group">
                    <button type="submit" class="btn btn-default">Log in</button>
                </div>
            </form>
        </section>
    </div>
</div>

@section Scripts {
    @await Html.PartialAsync("_ValidationScriptsPartial")
}
```

Figure 4.5: Coding Snapshot



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Claims;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Options;
using TwoFactAuth.Models;
using TwoFactAuth.Models.AccountViewModels;
using TwoFactAuth.Services;

namespace TwoFactAuth.Controllers
{
    [Authorize]
    [Route("[controller]/[action]")]
    4 references
    public class AccountController : Controller
    {
        private readonly UserManager<ApplicationUser> _userManager;
        private readonly SignInManager<ApplicationUser> _signInManager;
        private readonly IEmailSender _emailSender;
        private readonly ILogger _logger;

        0 references | 0 exceptions
        public AccountController(
            UserManager<ApplicationUser> userManager,
            SignInManager<ApplicationUser> signInManager,
            IEmailSender emailSender,
            ILogger<AccountController> logger)
        {
            _userManager = userManager;
            _signInManager = signInManager;
            _emailSender = emailSender;
            _logger = logger;
        }
```

Figure 4.6: Coding Snapshot

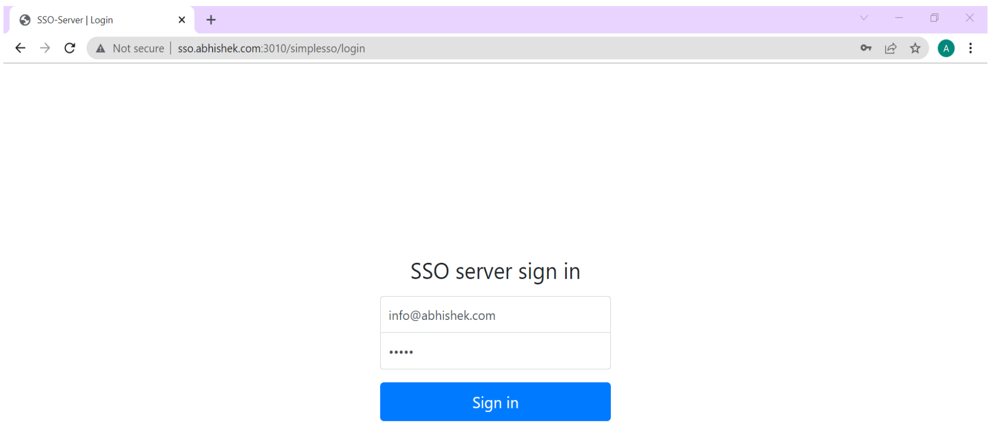
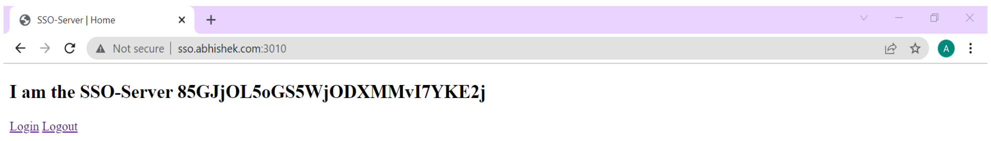


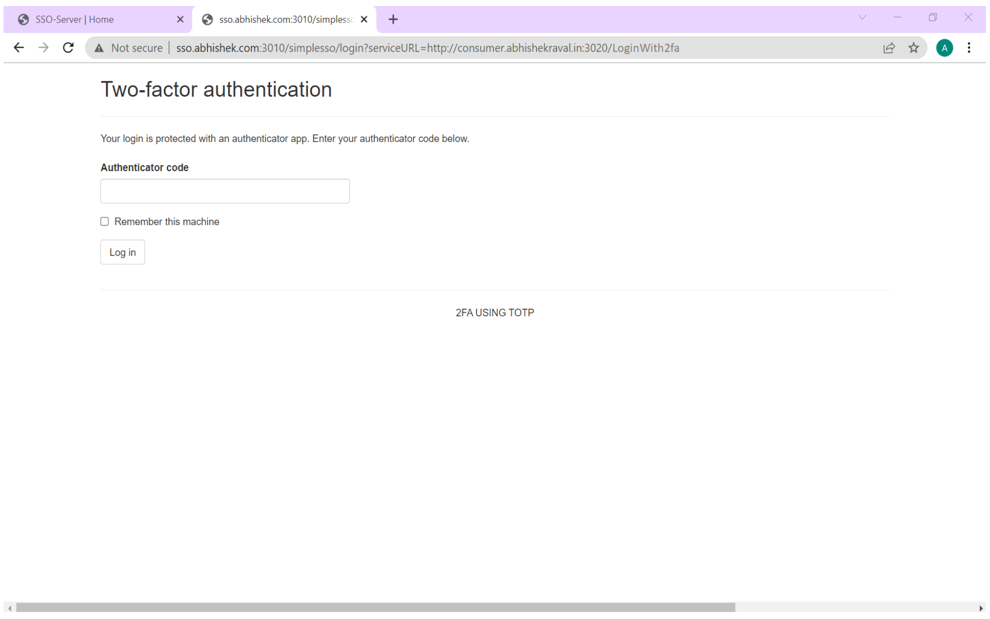
Figure 4.7: SSO Server Login



I am the SSO-Server 85GJjOL5oGS5WjODXMMvI7YKE2j

Login Logout

Figure 4.8: SSO Server Login Successfully



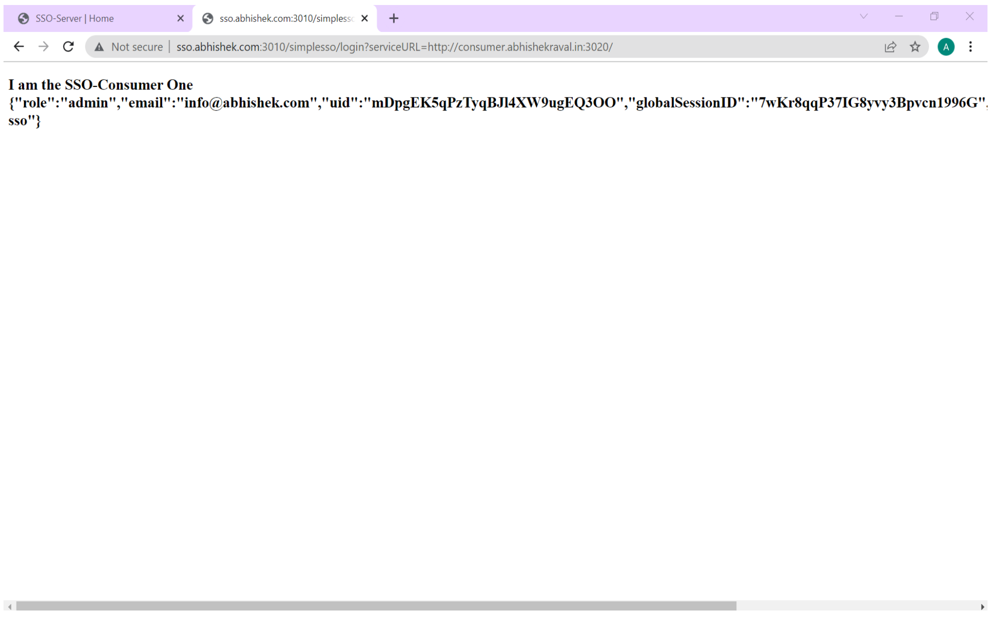
sso.abhishek.com:3010/simplesso/login?serviceURL=http://consumer.abhishekraval.in:3020/LoginWith2fa

Two-factor authentication

Your login is protected with an authenticator app. Enter your authenticator code below.

Authenticator code

Figure 4.9: 2 Factor Authentication using TOTP



sso.abhishek.com:3010/simplesso/login?serviceURL=http://consumer.abhishekraval.in:3020/

```
I am the SSO-Consumer One
{"role":"admin","email":"info@abhishek.com","uid":"mDpgEK5qPzTyqBJl4XW9ugEQ3OO","globalSessionID":"7wKr8qqP37IG8yvy3Bpvcn1996G",
sso"}
```

Figure 4.10: SSO Consumer Login

Chapter 5

Use Case - MFA Solution for V2X Application

We explore the MFA mechanism with clear factors I and Fa in this use case. Each factor Fai has its own secret Sei, which is retrieved from the user via the appropriate process (PIN, biometric, etc.). Under the worst scenario, it is linked to biometric information, which has a low possibility of changing through time. The factors and secrets related to them may therefore be expressed as:

$$Fa1 : Se1, \tag{5.1}$$

$$Fa2 : Se2, \tag{5.2}$$

$$Fal : SeL, \tag{5.3}$$

$$Fl + 1 : T, \tag{5.4}$$

Here, Fal+1 is a timestamp at point T. while I am factors needed to rebuild the secrets. whereas Sei is the sensor secret value. Since a fingerprint is often an irreversible component, supplying the real secrets to the validator is not an alternative, particularly in the case of confidential biometric data. As a result, allowing even a trustworthy instance to get the necessary data is a risky move. So, the algorithm requires Sei to be derived from the factors. In other words, rather than allocating parameter Sei, the suggested approach generates the system secret Se based on the gathered factor values Sei. Rather than allocating them in the very first position, a system of equations related to the Lagrange interpolation technique with the components, their levels, and the secrets for system privileges is created.

$$\begin{cases} Se_1 = \bar{S}e + a_1Fa_1 + a_2Fa_1^2 + \cdots + a_{l-1}Fa_1^{l-1} + a_lFa_1^l \\ Se_2 = \bar{S}e + a_1Fa_2 + a_2Fa_2^2 + \cdots + a_{l-1}Fa_2^{l-1} + a_lFa_2^l \\ \cdots \\ Se_l = \bar{S}e + a_1Fa_l + a_2Fa_l^2 + \cdots + a_{l-1}Fa_l^{l-1} + a_lFa_l^l \\ T = \bar{S}e + a_1T + a_2T^2 + \cdots + a_{l-1}T^{l-1} + a_lT^l \end{cases} \tag{5.5}$$

Here,ai denotes the calculated coefficients, and f(x)=Se+a$_1x$+a_2x^2+$\cdots$+$a_{l-1}x^{l-1}$, $f(0) = Se$.

A Lagrange interpolation algorithm reveals that the equation (7) in the system has just a resolution for Se. As a result, the framework secret Se may be retrieved using the usual Lagrange interpolation method and l obtained shares without transferring the part of the unique secrets Si to the validator. As a result, critical personal information is kept confidential. Because of the features of the Lagrange formulation, the matching polynomials can only represent one line; hence, every set of [Fai: Sei] will yield a unique Se. Furthermore, if the biometric data obtained by MFA does not vary with time, the secret is always the equally, which is a clear weakness in the system under consideration.

The suggested approach is resilient against the scenario in which all Sei stay constant throughout time. This is accomplished by introducing a one-of-a-kind time T parameter, which permits the existence of Fal and the related secret. Over the last phase of the studied cutoff methodology based on the Lagrange interpolation formula, security is ensured using the Rivest–Shamir–Adleman (RSA) technique or the ElGamal encryption/decryption algorithm. In this scenario, the secure threshold approach connected to secrets Sei in is demonstrated.

However, if all elements are present, our proposed approach may work outside the box because of the N number of sensors used for MFA. As a result, the system may be able to detect and report any obsolete factor information, such as weight fluctuation sensors, Figure print sensors, etc. When some of the elements are missing, access to a service might be automated. In the next part, we go through this functionality in further detail.

Factor Mismatching

Considering that our system has L = 4 components, the system secretes SE may be written in a simplified manner as a collection :

$$SE \leftarrow [Fa1, Fa2, Fa3, Fa4] \tag{5.6}$$

The secret recovery technique would fail if any of SEi were changed. By offering independent system solutions SEi for a smaller number of factors gathered, this approach is improved. In general, for l = 3, the number of feasible component combinations with one absent is equivalent to 4, as shown below.

$$\overline{SE1} \leftarrow [Fa1, Fa2, Fa3]\,, \tag{5.7}$$

$$\overline{SE2} \leftarrow [Fa1, Fa3, Fa4]\,, \tag{5.8}$$

$$\overline{SE3} \leftarrow [Fa1, Fa2, Fa4]\,, \tag{5.9}$$

$$\overline{SE4} \leftarrow [Fa2, Fa3, Fa4]\,, \tag{5.10}$$

As a result, the device may provide access based on a risk function policy. The second advantage is that it can notify the client (or the authorities) that a specific factor

Fi must be modified as a result of the failing SEi combination. This change does result in some transmission overheads, but it also allows for more authentication and lacks factor verification flexibility.

Chapter 6

Performance Evaluation

6.1 Single Sign-On

Global Market Trend for SSO Application Between 2020 and 2027, the Compound Annual Growth Rate (CAGR) for single sign-on or SSO enabled apps is predicted to be 12.0 %. In 2020, the market was worth \$0.94 billion, but by 2027, it may be worth \$2.13 billion.

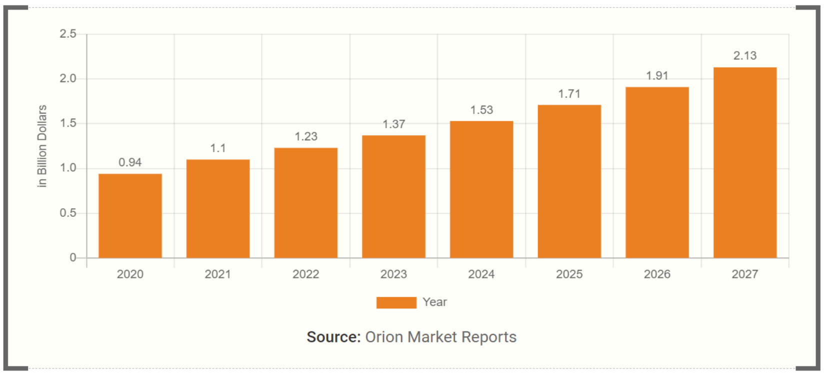


Figure 6.1: Source: Orion Market Reports

Areas to Keep an Eye on During SSO Application Performance Testing

- Generation, availability, and refresh time of authentication tokens.
- User's proposal parameters and redirect URLs are examples of SSO traffic.
- Requests to the SSO/IdP server using web services.
- Scripting for 3rd API services (synthetic monitoring).

Table 6.1: Performance Testing For Apps With and Without SSO

Factor	Application, Without SSO	Application, With SSO
Developing a Script	Scripting just needs to control application-level verification.	Scripting must manage both program and IdP level verification.
Processing of Scripts	Load testing of login requests is optional.	Load testing is required for login requests.
Observing	Supervision is only necessary at the user application and at data server levels.	All SSO systems, including app and database servers, should be monitored.

Challenges

- Several requests (Provider and IdP request/response) and the SAML token make script creation difficult.
- Finding the underlying cause of login difficulties is difficult.
- Examining variables that contribute to sluggish response times, such as network congestion and application/server length of stay.

Recommended Practices

- Redirect every request automatically to manage numerous redirection caused by request forwarding to IdPs.
- Activate the login scripts accompanied by a scenarios script to analyse the effect of SSO for every use case while executing specialized SSO test cases.
- To control cookies, go to Cookie Management Settings.
- Analyze resource on the service provider and IdP servers while the efficiency scripts are executing.
- For handling dynamic SAML tokens, use a dependable extraction option (such as regular expression, CSS extractor, or XPath extractor).
- Select a high-throughput SSO protocol. And set suitable refresh intervals and manage the cache.

6.2 2FA - TOTP

To protect cloud applications, a time-based one-time password authentication mechanism is used. The TOTP approach is commonly used in applications that must limit time, such as mobile banking and app transactions. This section outlines the essential features and discusses the methods used by the OTP authentication system. There are already numerous steps in the authentication process in the previous approaches discovered, as they have worn SMS OTP Authentication inside the authentication phase. Here, we've combined TOTP's combat with a user's individual QR Code, which might be a useful approach for ensuring high authentication security.

The Mobile Transaction Authorization Number, which is used to approve a person's transactions, is a crucial component of SMS OTP. The OTP is provided to the user's mobile device through text message via this technique. Nonetheless, SMS OTP security is dependent on SMS secrecy, which is guaranteed by mobile network security. Authenticator Apps use a shared secret that both the server and client Apps must keep track of. To create the multi-factor authentication code, this "seed" is combined with the period. The TOTP-based one-time password verification for the authentication process is raised in our technique by using TLS connection between server and client Apps. Because the seed is communicated through a secure channel, it is never revealed. For user proof of identity, user verification has grown more critical then ever before. The protection technique for usernames and passwords that are vulnerable to guessing and password-based attacks. For multi-factor systems, there is also the option of developing user authentication mechanisms. This research proposed a user authentication architecture based on TOTP for cloud application user identity verification.

Table 6.2: Comparison of the usability of the OTP and TOTP method

Parameters	**Simple OTP**	**TOTP**
Token	Cell Phone	Smartphone
Client Application	No App Needed	Application Needed
Service Access	Restricted	Access Worldwide
Secure Seed	Fixed	Dynamic
Availability	No	Yes
Derivation	SMS	Offline

Chapter 7

Future Work

In the future, the security level of the proposed system can be enhanced by adding a special cryptography algorithm in transferring TOTP as well as SSO credentials for verifying the authenticity. We can also focus on performance enhancement. we can add data coloring and data watermarking methods to add one more security level to our proposed architecture.

Chapter 8

Conclusion

In this research, we have tried to enhance the security of authorized users and parallelly serve single sign-on. The proposed system is the combination of TOTP and SSO which facilitate the user as well as provide a higher level of security. To protect the cloud from unauthorized access. Two-factor authentication is provided. SSO simplifies username password management to the cloud as well as provide access to many application through the cloud. SSO requires an extra strong password for better security. But the combination of TOTP and SSO provides two levels of authentication. If an SSO provider is hacked then also TOTP check again for authorization. By the merging of these two technologies proposed system provides facility as well as security.

Bibliography

[1] 2020 roundup of cybersecurity forecasts and market estimates. https://www.forbes.com/sites/louiscolumbus/2020/04/05/2020-roundup-of-cybersecurity-forecasts-and-market-estimates/?sh=5ff96d5381d7. Accessed: 2022-05-03.

[2] 50 cloud security stats you should know in 2022. https://expertinsights.com/insights/50-cloud-security-stats-you-should-know/. Accessed: 2022-05-03.

[3] 7 benefits of sso amp; why you need it: Loginradius: Loginradius blog.

[4] Abdelrahman Abuarqoub. D-fap: Dual-factor authentication protocol for mobile cloud connected devices. *Journal of Sensor and Actuator Networks*, 9(1), 2020.

[5] Furkan Alaca and Paul C. Van Oorschot. Comparative analysis and framework evaluating web single sign-on systems. *ACM Comput. Surv.*, 53(5), sep 2020.

[6] Pranayanath Reddy Anantula and G Manoj Someswar. Protecting data in cloud environment from intruders using fog computing mechanism and control flows. In *2017 2nd International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India*, pages 176–182, 2017.

[7] S Babkin and A Epishkina. Authenticatin prtcls based n ne-time passwrds. 1794.

[8] Sergey Babkin and Anna Epishkina. Authentication protocols based on one-time passwords. In *2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pages 1794–1798. IEEE, 2019.

[9] Aytaj Badirova, Shirin Dabbaghi, Faraz Fatemi Moghaddam, Philipp Wieder, and Ramin Yahyapour. An optimized single sign-on schema for reliable multi-level secu-

rity management in clouds. In *2021 8th International Conference on Future Internet of Things and Cloud (FiCloud)*, pages 42–49. IEEE, 2021.

[10] Carsten Baum, Tore Frederiksen, Julia Hesse, Anja Lehmann, and Avishay Yanai. Pesto: Proactively secure distributed single sign-on, or how to trust a hacked server. In *2020 IEEE European Symposium on Security and Privacy (EuroS P), Genoa, Italy*, pages 587–606, 2020.

[11] Surachai Chitpinityon and Maharat Tossa. New approach for single sign-on improvement using load distribution method. In *2021 Research, Invention, and Innovation Congress: Innovation Electricals and Electronics (RI2C)*, pages 44–47. IEEE, 2021.

[12] Min-Hee Cho, Eun-Gyeom Jang, and Yong-Rak Choi. User authentication technology using multiple sso in the cloud computing environment. *Journal of the Korea Society of Computer and Information*, 21(4):31–38, 2016.

[13] M Darshan, SR Raswanth, S Skandan, S Shakthi Saravanan, Ranjit Chandramohanan, and Priyanka Kumar. A secured blockchain based facial recognition system for two factor authentication process. In *International Conference on Electrical and Electronics Engineering*, pages 492–502. Springer, 2022.

[14] Sheren A El-Booz, Gamal Attiya, and Nawal El-Fishawy. A secure cloud storage system combining time-based one-time password and automatic blocker protocol. *EURASIP Journal on Information Security*, 2016(1):1–13, 2016.

[15] Parul Garg and Yashpal Singh. Sso (single sign on) implementation. *International Journal of Science and Research (IJSR)*, 5:988–990, 2016.

[16] Ionel Gordin, Adrian Graur, and Alin Potorac. Two-factor authentication framework for private cloud. In *2019 23rd International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, Romania*, pages 255–259, 2019.

[17] Nazmul Hossain, Md Alam Hossain, Md Zobayer Hossain, Md Hasan Imam Sohag, and Shawon Rahman. Oauth-sso: a framework to secure the oauth-based sso service for packaged web applications. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE Inter-*

national Conference On Big Data Science And Engineering (TrustCom/BigDataSE), pages 1575–1578. IEEE, 2018.

[18] Shirly Lee, Ivy Ong, Hyo-Taek Lim, and Hoon-Jae Lee. Two factor authentication for cloud computing. *Journal of information and communication convergence engineering*, 8(4):427–432, 2010.

[19] Wenzheng Liu, Xiaofeng Wang, Wei Peng, and Qianqian Xing. Center-less single sign-on with privacy-preserving remote biometric-based id-maka scheme for mobile cloud computing services. *IEEE Access*, 7:137770–137783, 2019.

[20] Jubin Luckose, Sameer Chindarkar, and Dhanamma Jagli. Cloud service security using two-factor or multi factor authentication. *Int. Res. J. Eng. Technol.*, 4(6):2066–2070, 2017.

[21] Ryan Melton. Securing a cloud-native c2 architecture using sso and jwt. In *2021 IEEE Aerospace Conference (50100)*, pages 1–8. IEEE, 2021.

[22] David M'Raihi, Salah Machani, Mingliang Pei, and Johan Rydell. Totp: Time-based one-time password algorithm. Technical report, 2011.

[23] Prashant Pandey and TN Nisha. Challenges in single sign-on. In *Journal of Physics: Conference Series*, volume 1964, page 042016. IOP Publishing, 2021.

[24] Vivek Kumar Prasad and Madhuri D Bhavsar. Slammp framework for cloud resource management and its impact on healthcare computational techniques. *International Journal of E-Health and Medical Communications (IJEHMC)*, 12(2):1–31, 2021.

[25] Vivek Kumar Prasad, Mrugesh Shah, Namrata Patel, and Madhuri Bhavsar. Inspection of trust based cloud using security and capacity management at an iaas level. *Procedia computer science*, 132:1280–1289, 2018.

[26] D Divya Priya and A Mahalakshmi. Data security in mobile cloud computing using totp generated by hmac-sha1 algorithm.

[27] Gokila R and Mannimaran A. Implementation of two-factor authentication access control in web-based services with cloud computing using c.net. *Innovare Journal of Engineering and Technology*, page 15–22, Jul. 2018.

[28] V Radha and D Hitha Reddy. A survey on single sign-on techniques. *Procedia Technology*, 4:134–139, 2012.

[29] Ashish G Revar and Madhuri D Bhavsar. Securing user authentication using single sign-on in cloud computing. In *2011 Nirma University International Conference on Engineering*, pages 1–4. IEEE, 2011.

[30] Dr. Yasir Saleem, Munwar Iqbal, Muhammad Amjad, Salman Bashir, Faisal Hayat, Muhammad Farhan, Amjad Farooq, Abad Shah, and Abad Ali. High security and privacy in cloud computing paradigm through single sign on. *Life Science Journal*, 9, 01 2012.

[31] SM Sangamesh and SS Joshi. A survey on: A secure cloud storage system: An approach.

[32] Shangcheng Shi, Xianbo Wang, and Wing Cheong Lau. Mossot: An automated blackbox tester for single sign-on vulnerabilities in mobile applications. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, Asia CCS '19, page 269–282, New York, NY, USA, 2019. Association for Computing Machinery.

[33] Mohammad Karim Sohrabi, Vahid Ghods, and Najmeh Mohammadian. Privacy of cloud data using a secure sso architecture. In *2017 Computing Conference*, pages 224–229, 2017.

[34] Amol Suruse, Shrikant Udmale, Vaibhav Doifode, and Amol Waghade. Applying a single sign-on algorithm based on cloud computing concepts for saas applications using md5 encryption.

[35] SungJin Yu, KiSung Park, and YoungHo Park. A secure lightweight three-factor authentication scheme for iot in cloud computing environment. *Sensors*, 19(16), 2019.

[36] Yuan Zhang, Chunxiang Xu, Hongwei Li, Kan Yang, Nan Cheng, and Xuemin Shen. Protect: Efficient password-based threshold single-sign-on authentication for mobile users against perpetual leakage. *IEEE Transactions on Mobile Computing*, 20(6):2297–2312, 2021.