Technical Report
KN–2017–DISY–03
v1.1 of 2017-06-09

Distributed Systems Laboratory

X.509 User Certificate-based Two-Factor Authentication For Web Applications

Thomas Zink **Marcel Waldvogel**

Distributed Systems Laboratory
Department of Computer and Information Science
University of Konstanz – Germany

This work was supported in part by the Ministry of Science, Research and the Arts (MWK) of the State of Baden-Württemberg through the funding of project bwITsec.

Abstract. An appealing property to researchers, educators, and students is the openness of the physical environment and IT infrastructure of their organizations. However, to the IT administration, this creates challenges way beyond those of a single-purpose business or administration. Especially the personally identifiable information or the power of the critical functions behind these logins, such as financial transactions or manipulating user accounts, require extra protection in the heterogeneous educational environment with single-sign-on. However, most web-based environments still lack a reasonable second-factor protection or at least the enforcement of it for privileged operations without hindering normal usage.

In this paper we introduce a novel and surprisingly simple yet extremely flexible way to implement two-factor authentication based on X.509 user certificates in web applications. Our solution requires only a few lines of code in web server configuration and none in the application source code for basic protection. Furthermore, since it is based on X.509 certificates, it can be easily combined with smartcards or USB cryptotokens to further enhance security.

Keywords. *multi-factor-authentication, authentication, crypto token, S/MIME, certificate, X.509*

Table of Contents

1 Introduction

At the time of this writing, the most commonly used user authentication scheme for login to digital services of all kinds is still a combination of username and password. If an attacker guesses or steals the user's password he can effectively steal the user's digital identity, access the user's personal information, and perform actions in the user's name. The password is often the weakest link[1].

For many applications this level of security might be enough. But sensitive information or the ability to perform critical actions require additional layers of security. Multi-factor authentication (MFA) is a methodology that introduces additional, independent authentication factors that all need to be validated when authenticating a user. In some application domains, like e-banking, multi-factor authentication is already an established mechanism. Many well-known service providers have also adopted 2FA (two-factor authentication, often called '2-step verification'), including Apple, Google, Microsoft, and even Steam.

Authentication factors differ in nature and include (see Figure 1):

- Something you know (knowledge of a secret, like a password or PIN)
- Something you have (possession of an object, like a smart card or cryptographic token, aka cryptoken)
- Something you are (physical features, like fingerprint or retinal patterns)

Fig. 1. Types of authentication factors - knowledge, possession, physical features.

Independence of the authentication factors is critical to ensure that a compromise of a single factor does not affect the others. Although the authentication factors are independent, their affiliation to the authenticating identity must be apparent and verified by the service. All authentication factors must also be verified by the service. In fact, many authentication schemes, that claim to offer MFA, actually fail to associate all authentication factors with the same single identity, or to check all authentication factors server-side, as discussed in Section 3. For example, the typical usage of password-protected user certificates or SSH keys does not comprise a multi-factor authentication, since the password is not checked by the server but by the client.

MFA introduces additional complexity in applications as well as identity management[2]. Many standard services like LDAP or AD do not support MFA out-of-the-box. As a result, additional authentication services, user management, and changes to client applications are usually required. These requirements can lead to significant costs in time and money. It introduces additional effort not only in implementation but also in operation.

In this paper, we suggest another unique form of two-factor authentication for web applications that requires little changes in the application and builds

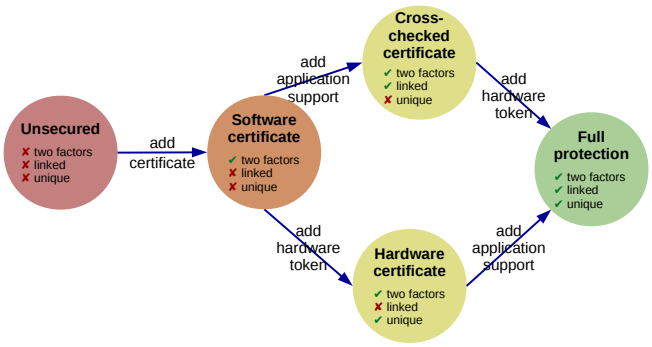

Fig. 2. The different stages of securing a web application with certificate-based second factors. Adding a software certificate provides an additional factor without linking it to the users identity. Application support is required to link users with their certificate. Instead of storing the certificate in software, it can be stored on a hardware token, to guarantee it stays private.

on existing identity management infrastructures. We utilize standard X.509 user certificates (aka S/MIME certificate) as proof for the user's identity and as a first factor for authentication. After the user has provided his certificate and thus proven his identity he is prompted for his password. This scheme can easily be combined with standard hardware solutions like smartcards or USB cryptokens (Figure 2). It is surprisingly easy to implement, yet affective and user friendly, and can be adapted quickly and flexibly for any type of web application without the need for another online service.

2 Related Work

For many decades now, passwords have been considered a necessary evil. Already in 1984, Thompson[3] noted that password checking can easily be circumvented. A few years later, Muffett[4] published *Crack* which allowed brute-force cracking of passwords (and enabled system administrators to identify passwords vulnerable to this attack). The primary target of network sniffing, keyloggers and several trojans[5] also were passwords. Herley and Florêncio[6] present a non-technical way to fool keyloggers by interspersing the password with random character sequences which are entered when the password field does not have the input focus.

Given the success of break-ins to steal passwords by the million and even billion in recent years[7]. The requirement of some organizations to force their users to regularly change their password without any exposure has led users to increasingly chose weak passwords, write them on Post-Its, or share them between multiple accounts instead of using password managers[1, 8]. Even passwords stored in encrypted form are worthwhile, as the users often use low-entropy passwords and share them between several accounts, reinforcing the importance of the weakest link.

Countermeasures including one-time passwords in software[9] or hardware tokens[10] have been introduced to overcome the password weakness. However, they have only been deployed in selected environments, as widespread compati-

bility to existing software has been limited, especially in open and heterogeneous environments such as those seen in research and higher education.

There, various forms of authentication over several protocols (local, LDAP, AD, web-based, ...) are used, often in combination to ensure access even if some systems have failed. For services offered among institutions of higher education, Shibboleth is now becoming the de-facto standard, even beyond web applications[11]. Work is under way to include two-factor authentication into Shibboleth[2]. This currently requires use of Shibboleth, which is not common for in-house applications yet and requires configuration at the Identity Provider, not (only) at the Service Provider.

3 State of the Art

Support for Multi-factor authentication in daily applications is steadily increasing. Many prominent service providers or manufacturers have started to adopt MFA in one form or another. Hardware and software developers also react to newer security threats and recent years have seen a growing number of products, services, and applications regarding multi-factor authentication; be it support in established software, new hardware tokens, or additional authentication services.

There are several different methods that are commonly used, which are often similar in use but differ in technology. The following sections provide an overview of contemporary authentication technology.

3.1 Trusted Channels

This is probably one of the earliest and still most common ways of authentication with multiple factors. The service provider keeps a record of trusted clients the user has used in the past, as well as some contact information that is verified to be in the user's possession, usually the email address or a phone number. Some service providers that also offer hardware also allow the user to register a trusted device, usually a phone or tablet, that is in his possession and serves as proof of ownership.

When the user accesses the service from a new device or client - like another browser - the service prompts the user to enter a secret which is sent to his trusted contact address or trusted device.

Many web service providers have adopted this scheme, including Steam, Amazon, Facebook, Apple, and Google.

3.2 One-Time-Password (OTP)

A One-Time-Password is a secret that can only be used a once. An important aspect is how to compute and distribute the password and and how to ensure that both the service and the client are aware of the same password.

With OTP, the user and the service provider share a common static secret which is concatenated with a variable. This secret should be kept on a dedicated device to fulfill the "what you have" premise. The result is then hashed multiple times to generate the pseudo-random password, known only to the parties knowing the secret and the variable (see Figure 4).

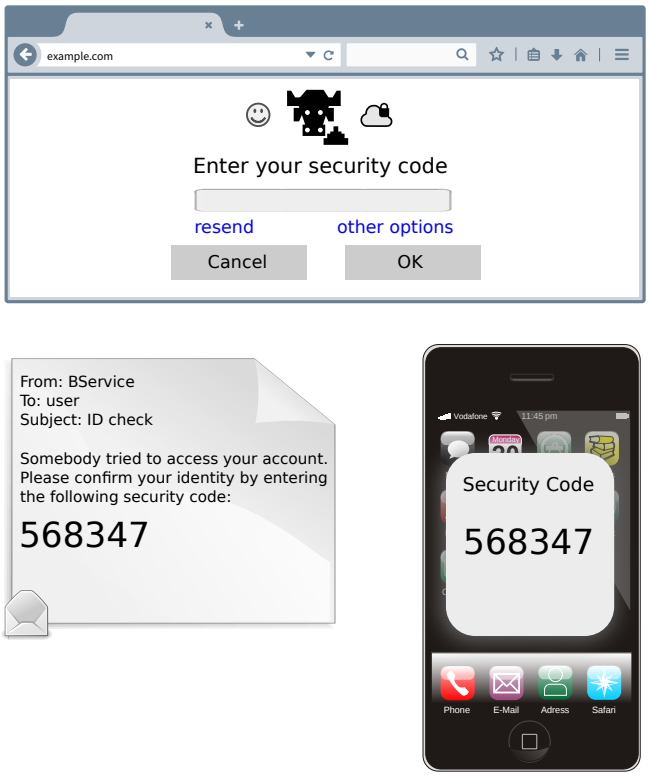

Fig. 3. Multi-factor authentication with trusted channels. When a new client is used the service sends a security code to a trusted device or email address.

Two methods have been established, the counter-based OTP (HOTP) and the time-based OTP (TOTP). With HOTP, service and client keep track of a counter that is used as variable input. The counter is incremented each time the OPT function is used and has to remain in sync in order for the parties to compute the correct OTP value. This introduces significant sync problems, which are somewhat mitigated by allowing a window of counters.

TOTP on the other hand uses the current time as variable input. The prevents that counters can run out-of-sync but requires that the clocks of server and client are synchronized.

OTPs have gained momentum in user acceptance and distribution due to the availability of smartphone apps like the 'google authenticator' or 'FreeOTP' which allow easy management and usage of OTPs for multiple services. Many online services have incorporated OTP, including Github and Amazon. There are many devices, applications, and services available, the make use of OTPs, both proprietary as well as free and open source.

For linux and SSH authentication there are libpam-google-authenticator[1] and libpam-oath[2]. We highly discourage using libpam-google-authenticator, since the built-in key-generator leaks all private information, like the secret key, to Google

[1] https://github.com/google/google-authenticator-libpam
[2] http://www.nongnu.org/oath-toolkit/

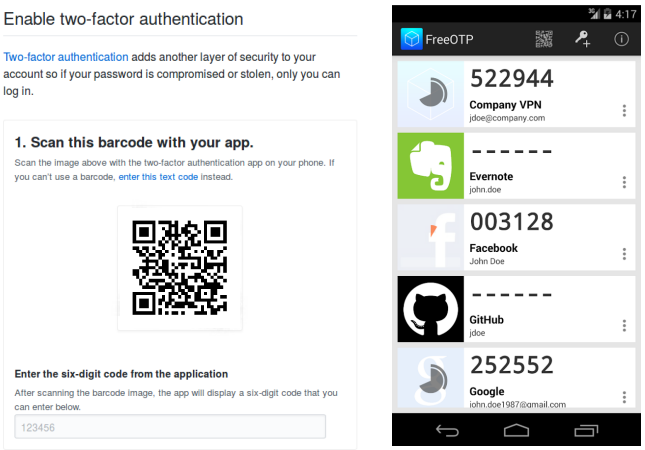

Fig. 4. OTP authentication. The service and the user share a common secret, represented by a QR code or a hex/base32 encoded number. The user can use a smartphone app, like FreeOTP, or hardware devices, like Yubikey or Nitrokey, to generate the authentication code.
Image sources https://github.com and https://play.google.com/store/apps/details?id=org.fedorahosted.freeotp.

to generate the QR code for the smartphone app. A similar module exists for Apache ('mod-authn-otp')[3] There are also open source authentication servers like 'LinOTP' and its fork 'PrivacyIdea' that aim to provide a central second factor authentication service usable by many applications (like e.g. LDAP). These services also offer other second factor methods built-in or in form of plug-ins.

While OTP is easy to use for the consumer, it requires significant changes to applications and extended user management. LinOTP/PrivacyIdea try to mitigate the user management problem by providing a self service portal that allows users to manage their own second factors. Ironically, the self service portal does not allow two-factor authentication, which takes the whole service *ad absurdum*.

3.3 Transaction Authentication Number (TAN)

Transaction authentication numbers are prominently used in bank related transactions. They are a form of a one-time-password in the sense, that they also are valid only for a single transaction. Initially, TANs where used not to authenticate a user or session, but to authenticate critical transactions like money transfers. They were been distributed in paper form as TAN-lists and the user was responsible for managing the list and keep the numbers safe. With the ubiquity of mobile phones the standard now is to send the user a SMS containing the valid TAN. The so-called SMS-TAN method is also used for user authentication and sometimes combined with the trusted devices / contacts methods above.

3.4 Universal 2nd Factor (U2F)

A new-comer is the universal second factor, or U2F. It is based on standard private/public key cryptography, much like TLS. The user has a secure token

[3] https://github.com/archiecobbs/mod-authn-otp

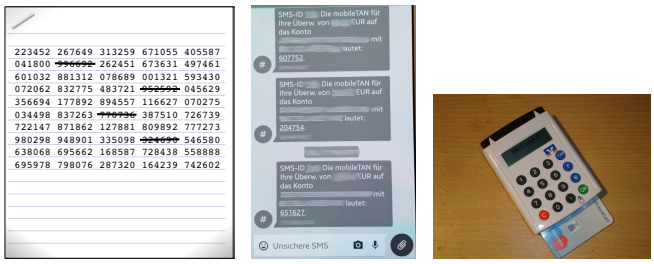

Fig. 5. Examples for TAN generation. Left: a classical printed TAN list. Middle: SMS-TAN. Right: chip-TAN
Image source : https://commons.wikimedia.org/wiki/File:SmartTAN_optic-Gadget.jpg

that generates the private key and then publishes the associated public key to the online service. When trying to login, the online service uses the public key to generate a challenge and sends that to the user's U2F device. Only if the user's device can solve pass the challenge, the user is authenticated.

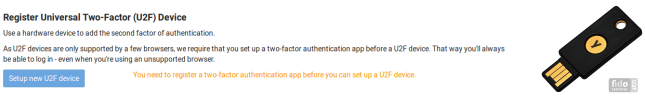

Fig. 6. Registration of a U2F device. The user needs to use a supported browser that is capable of communicating with the U2F device. The device itself can be a Yubikey or any other U2F capable token.
Yubikey image source: https://www.yubico.com/

U2F requires a secure token as well as client software that can communicate with the token. The web service must also be made aware of U2F. At the time of this writing, Chrome is the only browser that supports U2F out-of-the-box. The prime developer Yubico also provides a REST enabled authentication service that online service providers can use to reduce their implementation and maintenance costs. Yubicos authentication servers are open source and can also be self hosted.

3.5 X.509 User Certificates

X.509 Client certificates for authentication do not see widespread distribution, at least in consumer products. They are mostly found in areas with restricted access, usually in the form of smartcards.

Web server support for client certificates has been around for years, however, the common use case is what we call a 'certwall' Figure 7. That is, access to a resource on the server is only granted with a valid certificate signed by a specific authority. If access must only be allowed to certain people, the webserver can also be configured to only allow specific certificates, identified by a unique identifier like the serial number or the dinstinguished name (DN).

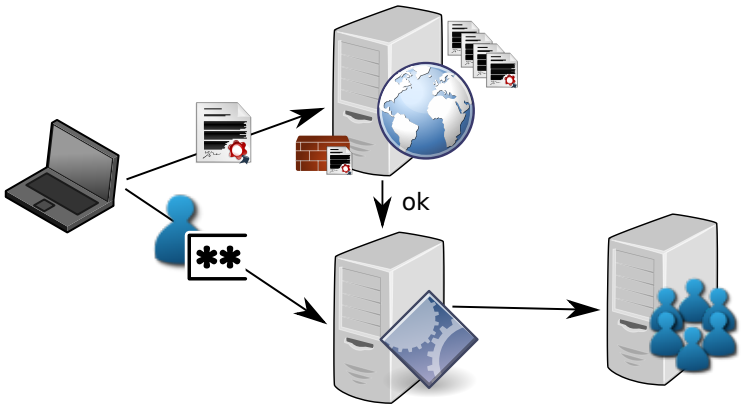

Fig. 7. A typical "certwall". The server requests a valid certificate from the client to allow access to a location. The user then authenticates with his username and password. No crosschecking of the certificate and user's identity is performed.

However, this form of certificate authentication is not a valid form of multi-factor authentication, since the identity of the user is not crosschecked with the identity provided by the client certificate. Ironically, it still requires extensive additional user management.

3.6 Verdict

Although there are a multitude of available methods and products to choose from, we found that most of them are not suitable for widespread deployment, especially in heterogeneous and complex environments like those found on university campuses.

The products either won't support all the required platforms or have too high requirements on production and maintenance. Some introduce additional dependencies on additional, maybe external, services.

For widespread implementation and both admin as well as user acceptance, the solution must be simple and introduce very little additional costs.

4 Threat Model

Our solution was triggered by the requirements of the following two high-risk groups:

Help-desk support personnel. To underline the openness of higher education institution, face-to-face support help desks are increasingly used, often residing in a public space such as the library. The opening times often extend beyond the staffing period of the desk (Figure 8).
Therefore, the machines will be left unattended overnight and could be tampered with, including installing software or hardware keyloggers. Physically locking the machines away daily would require significant effort.
Furthermore, the personnel operating these machines often has significant privileges, such as creating a new user or resetting a user's password.

Roaming system administrators. In a user-friendly setting with diverse client setups throughout an education and research organization, support staff and system administrators will have to visit users at their desk. Often, some configuration steps have to be performed from these computers. However, it remains unclear whether the machine is clean from trojans or keyloggers.

It can generally be assumed by the user in front of the machine that the machine has not been tampered with. However, a small risk remains.

Fig. 8. The university helpdesk located in the library. It is exposed to the public 24 hours every day. Machines are not secured and are to be considered compromised.

Similar thinking generalizes this to personal machines (laptops or desktops) in office rooms. An intruder might have stealthily broken into the room and modified the computer, i.e., performing an *Evil Maid* attack[12]. Even a machine of a vigilant system administrator can be infected by malware, especially in a targeted attack. The risk there is even smaller, but remains[4].

Our threat model therefore includes

- potentially untrusted terminals (public, customer, possibly hacked, keylogger),
- operated by a small, controlled user group with elevated privileges,
- logging in to (one or several) web services with elevated privileges.

The goals therefore are to

1. avoid additional effort on this group for unprivileged operations,
2. minimize the impact for privileged operations, and
3. prevent stealing of credentials with long-term validity required to perform these privileged operations, while
4. requiring minimal intervention to those web services.

[4] The analysis of one local incident suggests that this has happened at least once.

5 Solution

As shown in Section 3 existing solutions pose significant overhead in application changes, user management, and maintenance of additional services. While this might be appropriate in certain application domains, the overhead still leads to increases in investment and operation costs which are hard to justify.

Especially the public authorities have strict budgets and low to none economic pressure for additional security. Universities in general also are attractive targets for attackers due to their open and diverse infrastructure and the potential theft of knowledge and technology. Thus, some general conditions exist, that highly influence the probability of the implementation of additional security measures Figure 9.

Fig. 9. The conditions for additional security measures in the public sector.

1. Few changes to existing applications
2. Little to none additional user management
3. Easy to maintain
4. Work on all major operating systems
5. Free and open source or little to no cost

Our solution is based on the observation, that the vast majority of web applications use the user's email address – or the local part of the email address – as the user's login name. This holds for most public web services, like Amazon or eBay, but also for many public or private organizations and companies that operate their own email infrastructure. Usually, the user's email user name/address is also his unique identifier (UID) used for organization-wide authentication. If the email address is not used for authentication in the organization, another UID scheme usually is in place. X.509 certificates must also be unique for a user and demand the usage of a 'distinguished name' as subject. This subject can also include the UID, which can then be used for authentication instead of the user's email address.

Many organizations also either have their own certificate authority, or use the services of a public or commercial certificate authority. Especially the public authorities are likely to operate their own public certificate authority for use in government agencies or educational institutions.

While client certificates slowly take off, particularly in areas where trust or confidentiality is crucial, X.509 certificates are still usually found on servers and used for host authentication and SSL/TLS encryption. Because of this, support for X.509 client certificates has long been neglected. However, interest in stronger authentication schemes and X.509 client certificates as well as cybersecurity in general has risen in recent years. This is due to a combination of events, including Snowden's leak of thousands of confidential NSA documents and well-known

attacks on cloud providers (e.g. Apple's iCloud "celebrity breaches"). As a result application developers and service providers have started to adopt a variety of multi-factor authentication schemes, from relying on trusted devices (Apple) to hardware tokens (Amazon, Github).

Still, aside from specialized applications that use smartcard authentication, X.509 client certificates are usually only used on web servers to protect locations or directories. For example, anyone providing a valid certificate signed by a specific authority is allowed to access the login screen of an application (Figure 7). While this adds another authentication factor this cannot be considered 2FA, since there is no process that actually validates that the owner of the certificate is the same entity that tries to login to the application. The reason for this is because the certificate request is issued by the web server while the login is requested by the application.

5.1 2FA with client certificates

In order to enable true multi-factor authentication the application needs to check the identity of the client certificate and somehow match that to the identity of the user trying to login. This requires additional effort in identity management to provide this missing link.

We observed, that for most web-based applications, users login with their email address or the local-part of their email address. Since X.509 user certificates are issued for a specific email address, it is surprisingly easy to extract the user name directly from the certificate on the server and use this as a validated remote user for the application (Figure 10). If the web server supports client certificate parsing, this can even be done directly on the web server. If not, the web server needs to expose the client certificate to the application which in turn parses the certificate and extracts the user name.

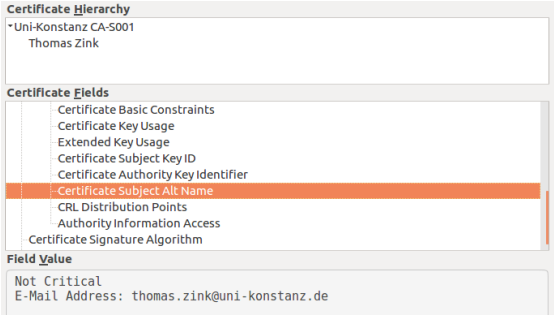

Fig. 10. A X.509 user certificate includes the user's email address in an extension field. This information can be extracted and then used as the user's login name for applications.

In our scheme, the webserver requests the user's certificate, checks the validity, and then exposes the certificate to the application. The application than extracts the user's username directly from the (validated) certificate and sets the username server-side. Note, that the client or user is not able to actually

change or set the username, since this is completely done server-side. The user only provides his password to complete the authentication. The process is shown in Figure 11.

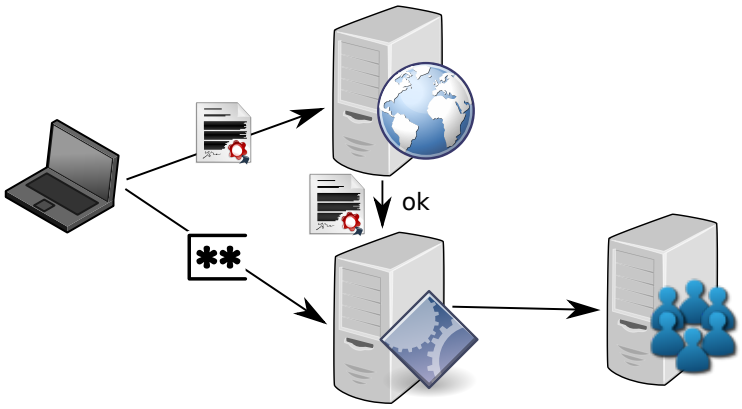

Fig. 11. 2FA with X.509 user certificates. The webserver requests a user certificate, checks the validity and exposes it to the application, which extracts the username. The user only has to provide his password.

Apache2 supports the necessary options to extract the since version 2.4.12. Prior to that, the client certificate must be exposed to the application. Listing 1 shows the configuration for Apache2. The important part is the export of the certificate and the environment variables.

Listing 1. Apache config

```
<VirtualHost *:443>
...
SSLVerifyClient require
SSLVerifyDepth 3
SSLOptions +ExportCertData +StdEnvVars
...
</VirtualHost>
```

On the application side, parsing the client certificate and extracting the email address can be achieved in few lines of code (Listing 2).

Listing 2. Email address extraction with apache < 2.4.12

```
$clientcert = openssl_x.509_parse($_SERVER['SSL_CLIENT_CERT'])
    ;
$subjectaltname = $clientcert['extensions']['subjectAltName'];
$email = substr($subjectaltname,strpos($subjectaltname,":")+1)
    ;
```

With Apache2 version 2.4.12 or later, extraction of the user's email address can be done with a single line of code (Listing 3), since support to access the required field is already built-in.

Listing 3. Email address extraction with apache $\geq$ 2.4.12

```
$email = $_SERVER['SSL_CLIENT_SAN_Email_0'];
```

In any case, Listing 2 shows the appropriate login form for the application. The form uses the formerly extracted email address as readonly value for the user name. The user is thus not able to actively change the login name. Since the web server verifies the authenticity of the client certificate, the user's email address is also verified. It is important to note, that the user is not allowed to change the username in any way. That is, setting the username must be done server-side, not in a POST evaluation. Applications usually set the username in the login page POST evaluation. This must also be removed from code, to prevent client side attacks.

Listing 4. Login form with readonly user name

```
<form>
<label><b>Username: </b></label>
<input type='text' name='user' value='<?php echo $email; ?>'
    readonly /> <br/>

<label><b>Password: </b></label>
<input type="password" placeholder="Enter Password" name="psw"
     required />

<button type="submit">Login</button>
<button type="button">Cancel</button>
</form>
```

5.2 Securing web applications

The proposed scheme can be used to efficiently secure any kind of web application with multiple authentication factors. A remaining issue is the storage location of the client certificate itself. Saving the certificate in the browser's local store exposes the private key to a number of attacks that might lead to it's loss.

Fortunately, our solution can be easily combined with hardware tokens like smartcards or USB cryptokens to ensure that the private key stays private, even if the user's machine is compromised. Not all browsers support hardware-based certificate stores, though. Depending on the required trust level, the application can be secured accordingly.

Example use cases The ease with which our authentication scheme can be implemented allows many applications.

Web Server or Proxy Using http auth and authentication modules, the whole web server can be secured. To quickly add 2FA support to multiple applications or servers, an authentication proxy could even be used.

Locations/Specific Applications Secure only specific locations or applications served by the web server, using true two-factor authentication.

Specific Actions Using the option `QUERY_STRING` to match post parameters or keywords in the URL, additional authentication factors can be requested for these specific actions.

6 Analysis

Our solution perfectly fulfills the goals stated in Section 4, as follows:

1. Unprivileged users or operations incur no extra burden.
2. Privileged operations are performed with zero effort for the basic (*software certificate*) case of Figure 2 and only a little bit more effort for the advanced protection modes. This applies to both setup and actual use.
3. The credentials that can be stolen do not provide any long-term access to privileged operations. If the computer is infected and under carefully targeted control, the current session may be hijacked; but there is no known general protection against this risk.
4. The web application hidden behind the 2FA does not need to be modified. Minimal changes are required to the web server (or a proxy inserted) for a first strong protection step.

As administrative commands can be identified by URL path or parameters, environment-specific definitions of what is considered a privileged operation can be easily applied. As access to these URLs is only granted to specific certificates, our 2FA solution even protects against several implementation or verification errors inside the application itself. As no on-line verification is needed with a central service (unlike other services like U2F or linOTP), this system does not increase the damage caused by a partial network or service outage at the university. The user certificates distributed for this purpose can be used to secure other services as well, including email, or vice versa, if policy allows. Then two birds are caught with one stone.

Our solution also excels compared to the conditions presented in Section 5. It

1. requires very few changes to existing applications
2. does not require any additional user management since it builds on existing infrastructure
3. is very easy to maintain
4. work on all major operating systems
5. is completely open source, with the exception of hardware token devices

7 Evaluation

We successfully implemented our client certificate-based authentication in our support front-end application used by supporters in our public help desk center. The workstations used to login are publicly exposed and leakage of passwords is a real threat. We already use the DFN PKI for X.509 certificates which allowed us an easy transition towards the certificate authentication.

Each support staff member receives a USB cryptoken to store his personal certificate. The workstations' browsers are configured to access the certificate on the token, which is secured by a person identification number (PIN). To successfully log into the support front-end the supporter has to provide the token, his PIN and then his personal password.

We conducted a user study among the support staff to evaluate the quality of the solution. Unfortunately, only six staff members reacted. However, according

to a study by Nielsen [13] a small sample is enough to identify a majority of usability problems.

Table 1 shows questions about the awareness of the user regarding the security of his password and the offered solution.

Table 1. Questions regarding user awareness

Question	Yes	No
Do you have to enter your password on untrusted devices	83.0%	17.0%
Do you fear that your password is in danger	83.0%	17.0%
Do you think the solution is justified	66.7%	33.3%

We also wanted to know the impact the solution has on the users' daily work and how well it is perceived (Table 2).

Table 2. Questions regarding user experience

Question	not	slightly	moderate	very
How much does the solution disrupt your work	16.7%	16.7%	16.7%	50.0%
How satisfied are you with the solution	16.7%	16.7%	16.7%	50.0%

Finally, we asked if the users know other methods for multi-factor authentication, and if they perceive our solution as better, worse oder equally good. The comparison is shown in Table 3 and Figure 12.

Table 3. Comparison to other solutions

solution	don't know	ours better	ours worse	equal
Smartcard	16.7%	16.7%	0.0%	66.7%
SMS-TAN	0.0%	16.7%	83.3%	0.0%
U2F	16.7%	16.7%	0.0%	66.7%
HOTP/TOTP	16.7%	16.7%	33.3%	33.3%

In summary, most users are aware that their credentials are in danger and that additional security measures are justified. However, users feel that the additional security impedes their daily work. Compared to other solutions our proposed scheme is perceived as at least equally well or even better with the exception of SMS-TAN.

7.1 Comparison to other methods

We compare our proposed method with the previously mentioned schemes with respect to the changes required to applications; dependence on additional services, hardware or software; and administrative overhead (e.g. user management).

Apart from smartcards, all other schemes require at least additional user management, in case of U2F or some OTP solutions even additional services

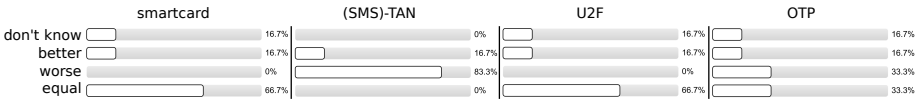

Fig. 12. How well is the solution perceived compared to other methods.

that need to be maintained, patched, and require high availability. In addition, applications that want to utilize these additional authentication factors need to be changed significantly.

Our solution can easily be used with smartcards or with USB crypttokens that have X.509 support. In case of smartcards, card readers must be provided. USB tokens work without any additional hardware requirements except for the standard USB port. In practice, however, we realized that driver and application support for both smartcards and cryptokens is still in its infancy. But that also holds for other second factor methods.

8 Conclusion and Future Work

Sections 6 to 7 clearly show how well the system performs and how easy and flexible it is to set up and use. This applies to both service provider and users.

According to our experience, security mechanisms are best deployed, when they can be installed with just a few keystrokes, no questions asked. The instructions in this paper, while easy to follow, can already be considered complex by some, which might prevent wide deployment. Ready-to-use software packages and containerized applications could reduce barriers and easy integration into existing infrastructures.

References

[1] D. Florêncio and C. Herley, "A large-scale study of web password habits," in *Proceedings of the 16th International Conference on World Wide Web*, ser. WWW '07. New York, NY, USA: ACM, 2007, pp. 657–666. [Online]. Available: http://doi.acm.org/10.1145/1242572.1242661 1, 2

[2] M. Simon, "2fa mit shibboleth mit linotp," Presentation at 65. DFN-Betriebstagung, Berlin, 2016, retrieved 2017-02-01 from https://www.dfn.de/fileadmin/3Beratung/Betriebstagungen/bt65/BT65_AAI_Shib-LinOTP_Simon.pdf. 1, 2

[3] K. Thompson, "Reflections on trusting trust," *Commun. ACM*, vol. 27, no. 8, pp. 761–763, Aug. 1984. [Online]. Available: http://doi.acm.org/10.1145/358198.358210 2

[4] A. D. Muffett, "Crack: A sensible unix password cracker," Message-ID <`1991Jul15.183637.6511@aber.ac.uk`> in newsgroups `alt.sources`, `alt.security`, Jul. 1991, retrieved 2017-02-01 from https://groups.google.com/d/msg/alt.sources/NAlVInmbvpk/GLdI4Dgv95MJ. 2

[5] S. McClure, J. Scambray, and G. Kurtz, *Hacking Exposed: Network Security Secrets and Solutions*, 3rd ed. Osborne/McGraw-Hill, 2005. 2

[6] C. Herley and D. Florêncio, "How to login from an internet café without worrying about keyloggers," in *Proceedings of SOUPS 2006*, Jul. 2006, retrieved 2017-02-01. [Online]. Available: https://www.microsoft.com/en-us/research/publication/how-to-login-from-an-internet-cafe-without-worrying-about-keyloggers/ 2

[7] S. Thielman, "Yahoo hack: 1bn accounts compromised by biggest data breach in history," *The Guardian*, Dec. 2016, retrieved 2017-02-01 from https://www.theguardian.com/technology/2016/dec/14/yahoo-hack-security-of-one-billion-accounts-breached. 2

[8] L. Cranor, "Time to rethink mandatory password changes," https://www.ftc.gov/news-events/blogs/techftc/2016/03/time-rethink-mandatory-password-changes, Mar. 2016, retrieved 2017-02-01. 2

[9] N. Haller, "The S/KEY one-time password system," IETF, RFC 1760, Feb. 1995, https://tools.ietf.org/html/rfc1760. 2

[10] A. Biryukov, J. Lano, and B. Preneel, "Cryptanalysis of the alleged securid hash function," Cryptology ePrint Archive, Report 2003/162, Tech. Rep., 2003, http://eprint.iacr.org/2003/162. 2

[11] M. Simon, M. Waldvogel, S. Schober, S. Semaan, and M. Nussbaumer, "bwidm: Föderieren auch nicht-webbasierter dienste auf basis von saml," in *5. DFN-Forum Kommunikationstechnologien: Verteilte Systeme im Wissenschaftsbereich*, 2012, pp. 119–128. [Online]. Available: https://netfuture.ch/wp-content/uploads/2012/simon12bwidm.pdf 2

[12] B. Schneier, ""evil maid" attacks on encrypted hard drives," https://www.schneier.com/blog/archives/2009/10/evil_maid_attac.html, Oct. 2009, retrieved 2017-02-01. 4

[13] J. Nielsen and T. K. Landauer, "A mathematical model of the finding of usability problems," in *Proceedings of ACM INTERCHI'93 Conference*, Apr. 1993, pp. 206–213. 7