Technical University of Denmark

Applied Cryptogrphy

Online Password Store

Authors:

Andreas Piculell	s113421
Mads Grunnet	s113413
Mathias Hecht	s113418
Mikkel Rømer	s113408

November 9, 2015

Contents

1 Introduction

The increasing amount of online password protected services, *Facebook, Twitter, GMail etc.*, puts a very important responsibility on the users to choose distinguished and strong passwords for each new service. Users who accept this responsibility will quickly fail to memorise all passwords, or simply choose the same password for multiple different applications. This usability-security trade-off often tips towards better usability, that is easier password memorization, thus vastly decreasing the level of security in the applications.

For an attacker; being able to successfully attack and exploit a weak point in a single application, and by that gain access to large spectra of applications, is an extremely dangerous aspect. Since it would allow the adversary to act authenticated on your behalf within the virtual space.

This paper proposes a solution which will highly increase the online security of users browsing password protected Internet services. The only requirement for a user is to remember a single strong master password, and after user-authentication with the master password, the solution will generate and handle all credentials needed by the user. Specifically, the solution will generate a random secure password for each service.

The overall solution will be an online password store, that can be used in Internet browsers as a plug-in, an application for mobile phones, and a website where all three versions will offer the same functionality across multiple platforms. The password store will authenticate users by their master password and automatically fill out login forms once the user is authenticated.
The authentication logic, security of communication channels, and cryptographic password storage will be discussed in section 6, and 5.

The focus and evaluation of the solution will be split up in the five different parts outlined below:

Usability The solution should be easy to use for a common Internet user.

Breach Detection It should be possible to detect when a master password is compromised and issue appropriate safety mechanisms such as resetting all passwords instantly.

Strong Passwords The generated passwords should be created using a secure RNG.

Safe Storage The passwords should be stored safely and should be irretrievable even if Stensikker A/S is compromised or turns evil. This ensures password integrity.

Secure communication The channel should be secured using standard protocols like HTTPS.

2 Risk Analysis

This section is partly structurally based on template by (Damsgaard Jensen 2015)

Whenever analysing risks, it is vital to first identify the assets, vulnerabilities and consequences in order to find the risks. Risk is "Exposure associated with a particular event"(Damsgaard Jensen 2015) and means that you take both cost and likelihood of the event into account when evaluation a risk.

2.1 Stakes

This section enlightens the different stakes for the different kinds of customers of the service

The users of this service ranges from the common Internet user seeking convenience, to the company storing trade secrets in a highly available, highly secure environment.
The common user will most likely not store information more important than the password for the personal mail, or if the banking institution allows it, to the personal funds.
The banking services will however mostly have two factor authentication, and a leakage from our service will not directly compromise the banking service unless the other factor is obtained as well.

While the compromise of personal banking accounts might seem like a big risk for the average observer, far more devastating scenarios might unfold if the master password of a systems administrator in a large corporation was leaked. Not only could a perpetrator with malicious intent obtain company trade secrets, they might even cause such a great havoc in databases and infrastructure, forcing the company to its knees.
Although strictly prohibited, one could also imagine a scenario where a high ranking government employee would use the service for the storage of either government mail passwords or even worse. A leakage here could stir up a whole government, especially if the employee had a high security clearance.

2.2 Assets

This section describes the key assets of the system.

Assets are the various key items in the ecosystem that potentially could be exploited by a malicious attacker in order to gain access to the system.

Listed below are the most important assets to the system.

- **Hardware**

- Servers

- Routers

- Users Computer

- Users Phone

- **Software**

- Users Browser

- Server Software

- Users Phone OS (For Two Factor)

- **Data**

- User Passwords

- Cryptographic Keys

- Authentication Tokens

- Local Storage

- External Databases

- Sessions

2.3 Vulnerabilities

Vulnerabilities are "the weaknesses in the system that makes a threat possible" (Damsgaard Jensen 2015).
A threat is the harm that can happen to an asset, and a vulnerability is the holes in the system that gives an attacker the possibility to exploit the threat.

The table below reveals some of the most likely vulnerabilities to the system.

Assets	**Confidentiality**	**Integrity**	**Availability**
Hardware			
- servers		Burglary/Modified/destroyed	DOS/loss of connection
- users computer		Burglary/Modified/destroyed	loss of connection
- users phone		Burglary/Modified/destroyed	loss of connection
Software			
- users browser		Virus/Trojan Horse/Backdoor	Deleted
- server software		Virus/Trojan Horse/Backdoor	Deleted/License expire
- user Phone OS		Virus/Trojan Horse/Backdoor	Deleted
Data			
- User passwords	Broken, stolen	Modification	Deleted
- Cryptographic keys	Copied,Broken	Modification	Deleted
- Authentication tokens	Disclosure	Modification	Deleted
- local storage	Broken	Modification	Deleted
- external databases	Copied,Broken	Modification	Deleted

Table 1: Vulnerabilities

A fault tree is the sequence in which a vulnerability takes its turn, and stepping through the various vulnerabilities and opening towards others. It is thus possible to see the true consequences of a vulnerability. The numbers represent the likelihood of the event to happen, and they are then added together to find the severity of the event. The severity is the cost the business will suffer.

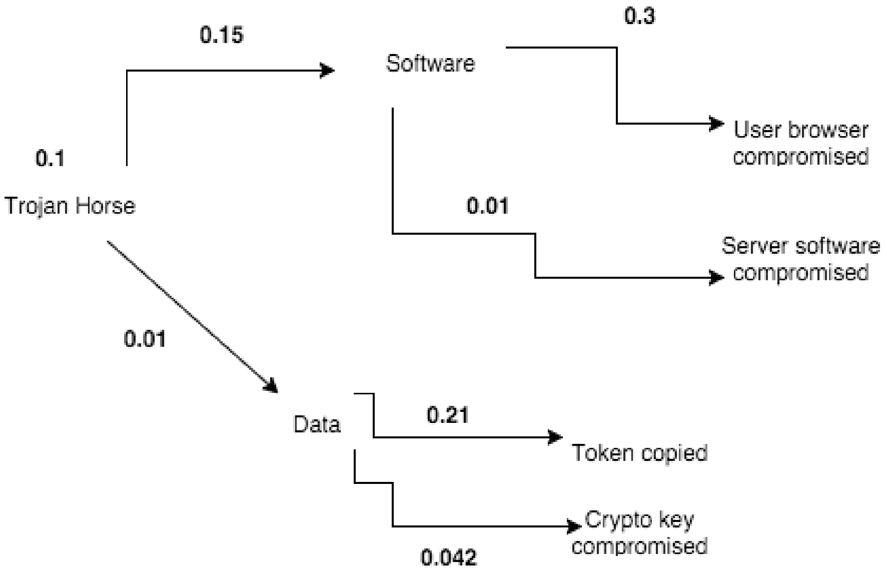

Figure 1: Fault tree.

Vulnerabilities	Consequences	Severity
User browser compromised	Users passwords might be stolen and broken	Negligible
Token copied	Users passwords might be stolen and broken	Negligible
Cryptographic Key compromised	Users passwords might be stolen and broken	Negligible
Server software compromised	All Users passwords might be stolen and broken, company secrets etc	Mission Critical

Table 2: The consequences and their severity

2.4 Consequences

The consequences of certain events are divided into four groups in order to weight their impact on the business (Damsgaard Jensen 2015):

- Safety critical: human lives are at stake
- Mission critical: survival of the company is at stake
- Costly: major impact on the earnings of the company
- Negligible: costs are small compared with operational costs

The reason for choosing Negligible for the user events is that the business will not suffer major blows if the users picks up a virus and loses the passwords. That will be a self inflicted injury by the user and since all users are isolated in the database, the user cannot inflict damage on the company, other than occasional bad standing or publicity. This can be countered by e.g two factor authentication and a smart fraud detection service (geographical logging).

It it clearly mission critical if the server software is compromised, since this would inflict the whole user-base and thus all the revenue.

3 Threat Model

This section will describe the various types of attacks and attackers that our solution must be resilient towards. The section also helps to define the scope of our security in the sense of who to protect against.

3.1 Perspective

The solution will be exposed to various threats and exploits of different nature. Indirectly, this system stores valuable assets for its clients, thus the asset-centric threat model seems appropriate. Combined with the attacker-centric model it can be analysed exactly who and what the system should be protecting against. Finally, a defensive perspective to incorporate security aspects while creating the application rather than an adversarial perspective is assumed.

3.2 The attacker

Generally speaking, the goal is a password store that is protected against *any* form of attack and every type of attacker; that is insiders, hackers, criminals and spies [11]. However, this is almost impossible so instead this paper assumes an *omnipotent attacker* with limited resources. That is, an attacker with *opportunity* and *complete knowledge* of the system but lacking the computational power of a government. For example, agencies like NSA would be outside the scope of our attacker assumption.

The above attacker-model is focused on breaking cryptography with system knowledge and computational power. This is a very defensive approach that tries to *prevent* an attack. However, it is also necessary to consider an attacker that could retrieve a password via the three B's, *beatings, bribery, and blackmail.* By incorporating this model as well, the security strategy will be more *reactive* and focused on damage control in a situation where the system is already compromised.

When not considering agencies or other entities with unlimited computational power, the worst possible attacker would be an insider at Stensikker A/S that tries to access client data illegitimately.

3.3 Motive

There can be several overall motives [12] for the attackers of our system. If an attacker could retrieve the password to the banking services of a client, there could potentially be a *financial gain* given a weak authorisation from the bank itself - that is, no two factor authorisation systems like e.g. NemID.
The *Cyber Terrorism* motive [11] is also relevant here, if an attacker could retrieve the password of a system administrator on a company-server. Strictly speaking, this motive is inherited from the financial motive since it would prove an important financial advantage to destroy the infrastructure of a competing company.

Again, at least considering the financial motive, the inside attacker from Stensikker A/S fits in the model.

Motives like *Fame*, *Acknowledgement* and *Curiosity* is not within the scope of this attacker.

3.4 Possible attacks

It is very likely, that the attacks against our communication protocol are *active* in the form of *fabrication* or *modification* of messages [**?**]. This could take place in the form of a replay attack where the attacker would try to authorize himself using the authentication token of another client. We do not consider *eaves-*

dropping, *traffic analysis* or *message deletion* as successful attacks towards this system. Hiding the existence of a client using services to avoid eavesdropping or traffic analysis, could be introduced, but this solution is considered to be sufficiently secure as long as the passwords stay safe.

The possible attacks described above are focused on the communication channel which is very important in this project since the solution will transfer valuable payloads over the Internet. However, it is also relevant to investigate attacks against our cryptographic design. Assuming an insider with free database access and complete system knowledge, cryptographic attacks will be *chosen ciphertext*. This attack is more powerful than *known plaintext*, *known ciphertext* and *chosen plaintext*, since being able to successfully recover keys without any plaintext knowledge is a forceful enemy. However, the entire credentials infrastructure is stored only in AES ciphertext, which makes CCA attacks an impractical challenge.

4 Usability

A crucial goal of this solution is to obtain a high degree of usability, which will also promote popularity [14]. It is important that the average Internet user will not find our solution too difficult. Thus the password store will be available cross-platforms (website, mobile application and browser plugin). The need of availability- and cross-domain features within this system is immense, lacking these features is most likely to force the users into introducing severe vulnerabilities.
Consider user Alice, Alice uses this solution whenever engaging in authentication forms on her laptop. However, since the solution is not cross-platform she stores all her passwords in plaintext on her phone. This is a severe risk and would compromise the entire platform if an adversary were to get hold of Alice's phone. Thus it is crucial that the solution proposed by Stensikker A/S engages all means to minimise user introduced risks. Moreover, when discussing usability, it is important to realise that usability is often a constant trade off with security.

4.1 Authentication

When creating a new user, the solution will always enforce two-factor verification (SMS or E-mail) to verify that a user is in fact who he claims to be. This will reduce the amount of fake users, and it will verify to the server that two factor verification will be available if a breach towards this user is ever detected. If a breach happens, the server will then issue a two factor verification to restore integrity to the original user.

Once the initial authentication is successful, the application will only ask for the master password if it is not already stored locally (client side). The possibility for two factor verification will greatly increase the overall security of the password store and will be to an acceptable loss of usability since the idea of two factor verification is known to almost any common Internet user. In addition, this solution, will actively engage the user into choosing good passwords as their master passwords. However, it is not expected that the user will choose completely random nor completely secure passwords, since the nature of humans simply does not enforce such complexity.

4.2 Password Generation and Retrieval

For each new Internet service, the password store will ask permission from the user to generate a new random password. Once created, the application will automatically fill in login credentials once a service is requested (for example Facebook). This will minimize the keyboard interaction needed form the client.

If a user wants to retrieve and see his credentials stored at the server then the application will ask for authentication via master password. If given, the application will show a list of all passwords for all services that can be copied to the clipboard if so desired. This security-usability trade-off is in favor of usability.

5 Architecture

The following section will introduce the proposed internal infrastructure of Stensikker A/S. Though-out the next pages, this report will evaluate previous discussed security risks and evaluate upon the solution.

5.1 Terminology

Below is presented a list, table 3, of the applied terminology of this section. In addition, this chapter introduces multi-layered networks refereed to as *DMZ*'s, that is Demilitarized Zones protected by various mediums.

Abbreviation	**Component**	**Network**
BRP	Boundary Rooter Package Filter	External DMZ
EDMZ	External DMZ	External DMZ
EPS	External Proxy Server	External DMZ
Web Server	Web Server	External DMZ
API	Web Application Interface	External DMZ
EDNS	External Domain Name Server	External DMZ
Main Firewall	Main Firewall	Internal DMZ
IDNS	Internal Domain Name Server	Internal DMZ
IPS	Internal Proxy Server	Internal DMZ
FDS	Fraud Detection Server	Internal DMZ
Auth. Server	Authentication Server	Internal DMZ
Internal Firewall	Internal Firewall	Protected Network
CDB	Credentials Database	Protected Network
ADB	Application Credentials Database	Protected Network
SDB	User Statistics Database	Protected Network
LDB	Logging Database	Protected Network

Table 3: Terminology of the internal environment structure.

5.2 Internal Infrastructure

The infrastructure proposed by this report is based upon the previous security- and threat analysis. The integrity of this solution is reflected in its architecture, since a weak architecture would increasingly higher the possibility of exploitation of internal components.

It is required that this solution engages communication over the Internet, since it must be accessible to any user at any time (section 4). Meanwhile, this implies the product to provide secure and authenticated bidirectional interaction over highly insecure channels. It is important to realise that the demand of security is bidirectional since the user requires a trustworthy authority when transmitting vulnerable information. The provider, Stensikker A/S, likewise requires authenticity such that data is not exchanged with adversaries.

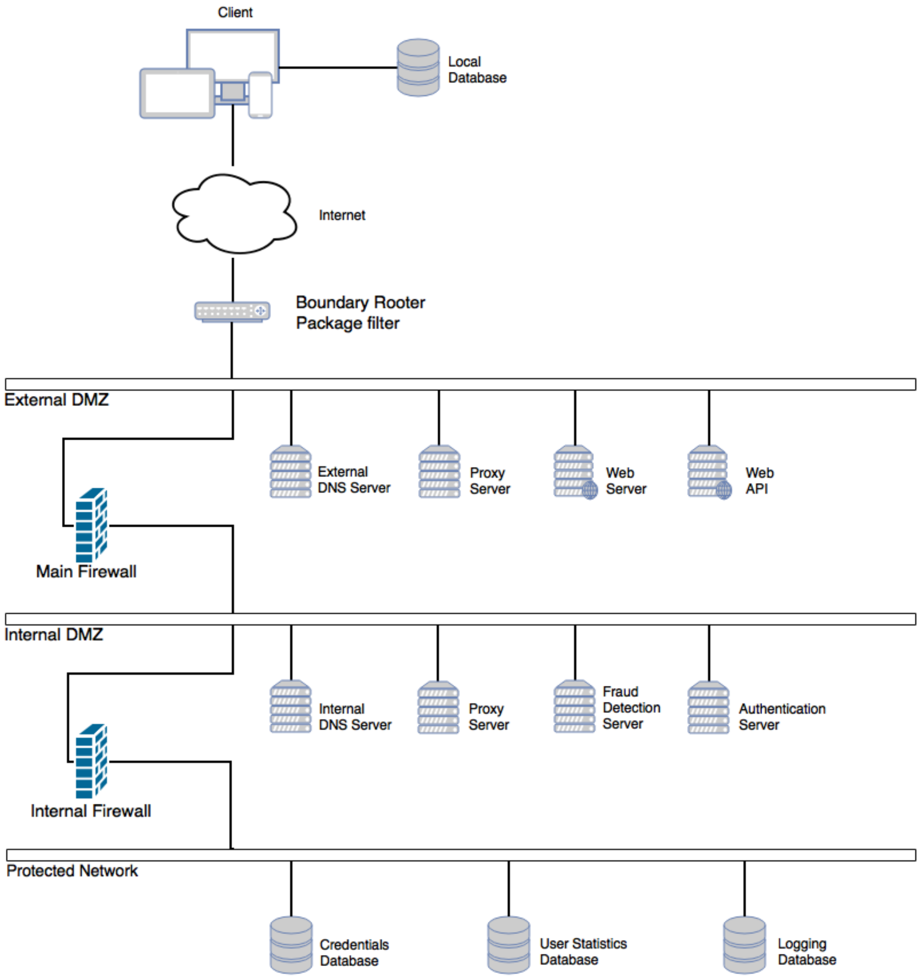

Figure 2: Proposed architecture of the internal environment.

5.2.1 Authentication

As previously discussed the solution requires 2-way authentication. From the users point of view, this solution issues server authentication through SSL certificates over the HTTPS protocol. However, whilst this provides server authenticity the user remains unknown.

In order to comprehend this and enforce 2-way authentication, the Kerberos Authentication Environment [4] is introduced. This system consists of two essential components: a *Key Distribution Center (KDC)* and a *Ticket Granting Server (TGS)*. These components are located within the secure internal DMZ, figure 2. It is crucial that these servers are trusted and firmly guarded against hostile activity, thus maintaining these servers inside the internal network is important [5]. Moreover, negotiation with these servers is thereby held within the borders of the corporation, and no direct interaction from the outside world is possible.

Whenever, an external entity requests communication, the following flow is enforced [4]:

1. User requests a *Ticket-Granting Ticket TGT* from the KDC.
2. KDC responds with an Encrypted TGT message.
3. User decrypts the response using its password and requests a Ticket with the decrypted TGT.
4. The TGS replies with an encrypted Ticket.
5. User uses this ticked to Authenticate itself towards the Service.
6. Time limited session is established.

In figure 2 the component discussed is called the Authentication Server. In addition to Kerberos authentication model, this solution introduces a fraud detecting step. By doing so, the solution increases the posibility og detecting adverse authentications. Whenever these threats are identified, the Authentication model is designed to issue a 2-factor verification. However, since this is mainly based on statistics, false-positives is expected but security is measured higher in this case since adversaries authorised on Alice's behalf would compromise all applications within the database.

5.2.2 Integrity

Means must be enforced by the architecture in order to sustain data integrity. That is, access and modification must only be allowed by authorised parties. When it comes to accessibility the proposed solution provides three layers of accessibility.

1) The External DMZ, or the outer zone, provides external access to the corporation. Within the boundaries of this zone an external user should be allowed to interact with the web interfaces alone. Moreover, this zone should not provide further environmental information to the user than these interfaces, thus communication is done behind a specialised Proxy-server. Figure 2. By communication behind this Proxy infrastructural investigations performed by adversaries is leveled to a minimum [6].
2) The internal DMZ. Within this ecosystem authorisation- and forensic systems is provided. These components is hidden from the public and secured behind advanced firewalls limiting the access to internal components alone. Moreover, access to this domain is done behind a proxy server as well. Should one of the servers located in the outer DMZ turn hostile secrecy of inner components is

sustained.
3) The protected network. The innermost network of this solution maintain the data domain of this solution. The data component represents the most vulnerable information of this product, since it holds credential information of its users. Although this information is encrypted, hostile retrieval of this information would allow more advanced offline decryption. Hence this data should firmly be protected. Moreover, by introducing a third layer, data access is limited even if a server is compromised.

Integrity implies denial of unauthorised data manipulations. From the infrastructural point of view, this is solved using authorisation components along with an enforced architectural separation of concern. However, data modification is also likely to occur on the fly. That is, doing communication flows such as within the Man in the middle attacks. Such attacks could occur in two different contexts. One, and most likely, doing transmission over the Internet, and secondly from within the cooperative domain. In the first case this problem is engaged by only allowing communication over secure HTTPS. In the latter case, this solution ensures integrity on inside attacks by the previously discuss usage of Proxy servers. This design makes it hard to identify communication partners, moreover this solution provides encrypted communication between these partners.

5.2.3 Fraud Detection and 2-Factor Authentication

Multiple security measures has been discussed so far. However, as in the nature of a password distribution system, this implementation is dependant upon a a strong master password. But, even if this password was selected wisely, it might happen, that this information falls into the hand of an adversary, through fishing etc. This solution must provide decent protection against such scenarios as well.
For this reason, a fraud detection component is introduced to the model[7], figure 2. The purpose of this component is to assemble user behavioural statistics, such as trusted devices, location, time, and amount of requests. When this component is introduced, it grants the possibility of detecting odd behaviour, such as a sudden authorisation attempt from foreign countries. When these diversities is identified immediate service denial of the concerned user is engaged. The authorisation component will then issue a need for two-factor authentication, before allowing withdrawal of further information.

5.3 Client Site Application

The proposed architecture consists of two different domains. That is, a client side- and a server side domain. This report already discussed the server side architecture, hence the focus of the following pages will be drawn towards the client application.

From the client perspective, the main purpose of the application is to retrieve application specific credentials. That is, whenever a user engages a authentication form the client plugin should initiate and request the needed data. From figure 3 this exact flow is visualised. Two important notations to make in this chart is: 1) It is not possible to perform any request task before authentication is successfully settled. For more information upon the Authentication see figure 4. 2) A user has the possibility of storing the retrieved credentials securely on the host machine. However, this is only the encrypted versions of the credentials, thus the master key is still required for decryption, lastly notice that all allocated RAM storage is flushed in the very last step.

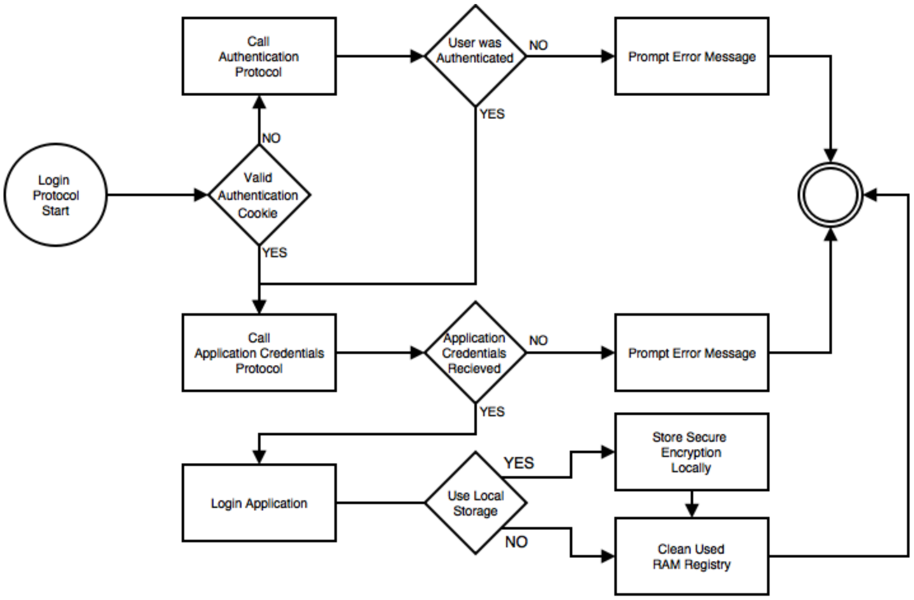

Figure 3: Procedure initiated on application password request.

5.3.1 Application Authentication

From figure 3 the overall protocol was introduced. But, it remained only to the application termination, and abstracts away from the actual Authentication protocol. This protocol is defined by figure 4. It is important to realise the that again the user has the posibility of storing the masterpassword securely on trusted devices, increasing the very usability of this solution. Moreover, notice the implementation of 2-factor authentication, or even the complexity of failed login attempts. That is, if a user is unsuccessfully requesting the authorisation form, eventual he/she is locked out. Thus making online brute force attacks impossible.

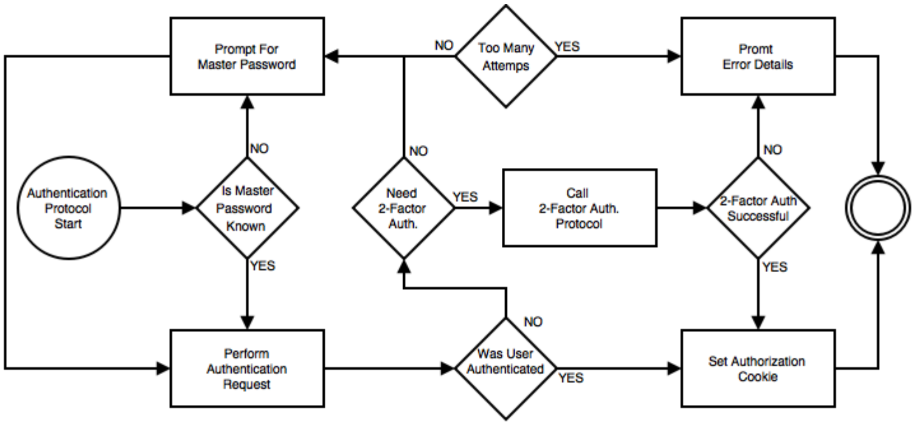

Figure 4: Procedure initiated when user authentication is needed.

5.3.2 Application Credential

Lastly, figure 3 referees to a protocol called Credentials Protocol. This protocol is responsible for receiving and decrypting the application credentials. The most important thing to realise here is the redundant usage of 2-Factor authentication. This feature is introduced even-though 2-Factor authentication was already implemented in the authentication protocol, figure 4. But in order to prevent potential session hi-jacking 2-factor authentication could be initiated on credential requests as well. Limiting potential damage provided by such attacks.

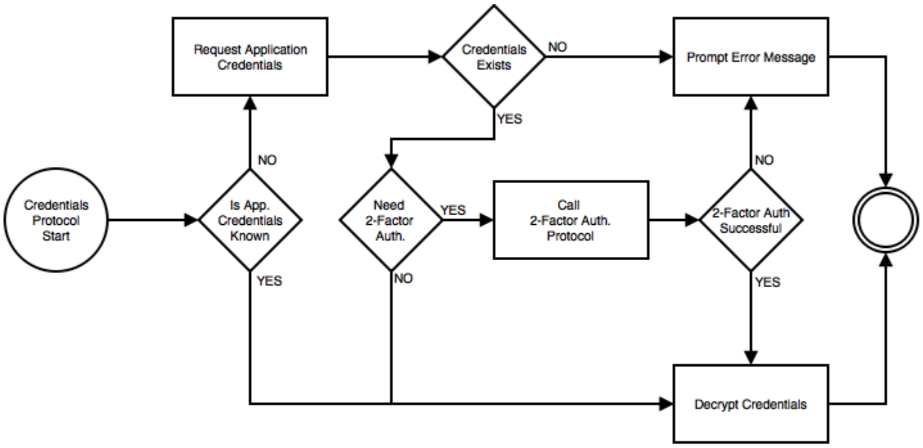

Figure 5: Procedure initiated when Application Credentials is needed.

6 Design

This section will state how to protect against the threats raised in the previous sections. Finally, it seeks to outline which algorithms it is going to implement and why.

6.1 Process

The client master password, MP, is used to both logging in to the system and to decrypt the stored passwords. This is obviously a security risk, without any other precautions. In this section, this report will state which precautions must be taken to sustain integrity within the system. In order to give a clean overall description of the design, this section will not go into detail with the specific algorithms used; this will be stated in section 6.3 on page 19.
To authenticate a user a normal log in procedure will be engaged, with the exception of the first time a user sign in using a new device. Here the user most be authenticated by two factors; that is, email or phone message verification both confirmed when the user registered in the system. To insure confidentiality of user passwords, these are never known in plain text by the remote environment. This is done using a non-invertible hash function. Specifically, the proposed solution uses a cryptographically secure hash function and unique seeding before sending it to the server. The properties of a secure salt will be discussed later in this section. The remote authentication server returns a token if the hashed password, send by the user who wants to be authenticated, is identical to the password stored in the credentials database. The token returned by authentication server is used in the future communication with server API. For further information on the proposed authentication protocol please refer section section 5 on page 10-Authentication
To ensure that user singularity, that is only the user who owns the password is able to use it, every stored password is encrypted with a symmetric encryption, with a key generated by a hash of the password and another salt. Moreover, the encryption of the password happens client side to protect the password secrecy. A globally unique means that the salt [3] is not used any other place in the world. This is achieved by using a secure (pseudo) random number generator which generates an at least 128 bit string.

The process of a typical log-in procedure, is illustrated by the sequence diagram in fig. 6 on the following page. Here the user is logging into the online platform facebook.com, with the help of the StenSikker A/S browser plug-in. First, the user opens the browser, and the browser instantiates the plug-in. The plug-in engages an authentication protocol of the user. The user credentials are send to the authentication server, *password is hashed as discussed.* The authentication server replies with a session token. The plug-in listens to the web pages opened by the user. Now the user opens facebook.com and the plug-in detects it, and request the encrypted password from the authentication server. The password is decrypted and the plug-in fills the facebook.com log in form for the user.

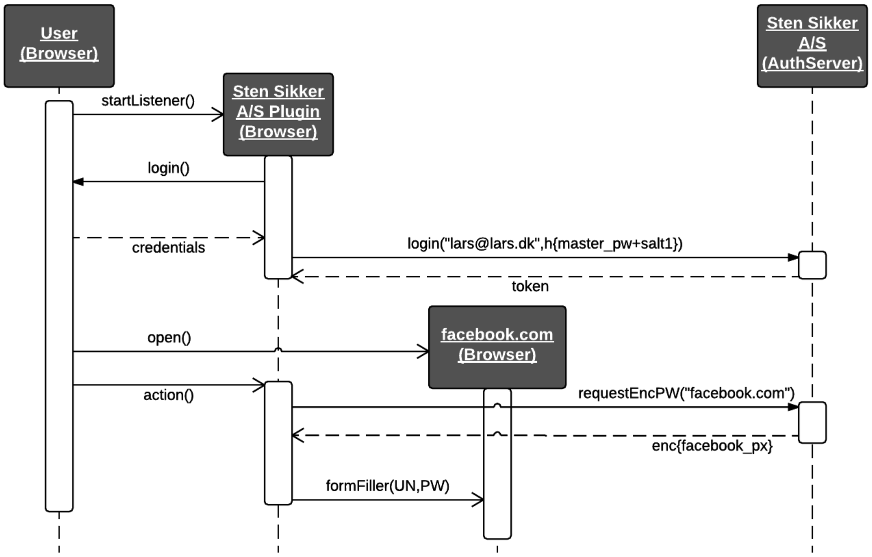
User (Browser)
Sten Sikker A/S Plugin (Browser)
Sten Sikker A/S (AuthServer)
startListener()
login()
credentials
login("lars@lars.dk",h{master_pw+salt1})
token
facebook.com (Browser)
open()
action()
requestEncPW("facebook.com")
enc{facebook_px}
formFiller(UN,PW)

Figure 6: Sequence diagram of a typical process

6.2 Passwords

Since the master password is user generated, it is considered weak. However, this password could compromise the secrecy of the users stored passwords collection, thus making the master password the weak link of this system. To prevent absurd weak master passwords this solution enforces, on user registering, that the master password has a minimum length of 8 characters, with a mix of lower case, upper case, and numbers. Furthermore, none of the users personal information, known by the system, is allowed in the password nor known weak phrases as e.g. *'123456'.* When the user chooses a password the interfaces should provide the user with guidelines to optimize the security of password, and give an security evaluation of the passwords.

As an additional security measure it is required of the user provide to register a phone number. Such that StenSikker A/S is capable of issuing a two factor authentication of any user, in case of a password is compromised[2]. The phone is chosen as the second factor, when authenticating in stead of an email because the email will, in case of a compromised password, most likely also be compromised as well. Since the user is encourage to store every password in our vault.
For usability reasons, it is allowed for the user to access his/her passwords from

any device as long the user can provide the system with a two factor authentication. When there is no sign of the password is compromised the system allows email authentication as the second factor.

6.3 Algorithms

This section will outline the chosen cryptography algorithms. Furthermore, it will be discussed what is required in order to keep the algorithms secure.

6.3.1 AES

Since all the encrypted messages used in this system are not shared with any other user, it is not needed any key exchange. Therefore, it is sufficient to only use symmetric encryption. This solution implements AES encryption. AES is an obvious choice when choosing a symmetric encryption. It is tested exhaustively, has high performance which is even more optimized on most newer processors as they have a dedicated AES hardware implemented. As AES only encrypts 16 bytes at the time two equal blocks will be encrypted to the same cipher text, this can be used for an attack. Therefore, we need to use a mode of operation and an IV (initialization vector) for the encryption. We have chosen CBC mode as the mode of operations. This means that each encrypted block is XOR'ed with the next plain text block before encryption. The first block in this mode is XOR'ed with the IV. A good IV has most of the same properties as a good salt. That is, the IV must be globally unique such that no two plain texts encrypts to the same cipher text. To satisfy this we need to use a cryptographically secure random number generator. The IV is public which means it can be stored alongside with the cipher text.

6.3.2 bcrypt

When choosing a cryptographically secure hashing algorithm, it is important that the output of the function is sufficiently long. Before determining what is sufficient, the reader is introduced to the Birthday Problem. The message of the Birthday Problem is that collisions are much more likely to happen than one might think. This fact in used to perform specialised attacks of hashed passwords using so called rainbow tables. Therefore, a hashing algorithm for this application such as SHA-256 is needed. SHA-256 outputs a 256-bit string, which is computational infeasible using rainbow tables. Another approach, which is chosen, is to use a hash function which has a built-in slowing function. This could be bcrypt. This slowing function is faster than what a human is able to notice, but too slow to be able to compute an entire rainbow table. With bcrypt the confidentiality of client credentials is sufficiently protected. The bcrypt algorithm also introduces huge time-complexities when brute forcing a users master password. As the hash happens client side, the server will not be affected by the slow algorithms in case of many requests.

7 Evaluation

Through-out this final section, this paper will question the proposed solution. Is the solution feasible, that is does it in fact improve security over regular user decided passwords?

This paper proposed a double sided solution. On the client side, this product acts as a browser plug-in, or an Application in the mobile domain. This choice was made for two main reasons: Firstly this improves the overall usability, since it easily integrates auto filling of authentication forms. However, this double sided solution also enables better security. The reason is found within the client side master password hashing. By hashing this password client-side through an cryptographic secure non-invertible hash- function, the actual password is never known server-side. This is in fact a huge improvement of the confidentiality. That is, even if a insider should turn evil, the master password would never be known.
One might argue that the hash of the password would be sufficient in order to recover the application keys. However, the very structure of the client application removes this possibility as well. This is realised by having a completely different hash of the master password done locally for decryption use. Consider the flow: Alice types in the master password and gets authorised by server using a hash `h1`. When Alice is authorized she uses the master password to create `h2` which is her decryption key for the application specific password. It is possible to do both hashing, encryption, and decryption client-side since this solution only makes usage of public available and secure algorithmic standards.

As mentioned previously, this solution employs secure communication over HTTPS only. However, this was done mainly to establish server-side trust. In fact, the proposed solution does not at any point ship vulnerable information through insecure channels in plain text. Whilst this is of course true while using HTTPS, the claim remains even without. By inspecting the data exchanged through the Internet two categories appears. 1) Data is exchanged doing authentication. 2) Data is exchanged doing key-retrieval. In the first case, the client will communicate the following data `<user_id, password>`. However, even if the solution were to communicate this information in plain-text, the data observed would be the secure hashed password, which by it self is worthless when decrypting application specific passwords. However, this information could be used to replay authentication procedures, but the application data remains confidential. In the latter case an adversary, Charlie, collects data from an application key retrieval. This time he observes `<session_key, user_id, app_pass>`. Again the password he observes, is in fact an AES cypher text, thus only through brute force he might establish the actual key. Again the data could arguably be used for session attacks, but the client data remains intact.

Moreover, two-factor authentication was introduced to minimise the change of authentication theft. That is, whenever an user authenticates using an new hardware, a two factor authentication is issued. What is more, the proposed environment introduced a specialised fraud detection server, allowing detection of odd behavioural user patterns. Thus minimising the change of identity theft. As a counter measure of this functionality, remains the potential likelihood of false detection of valid behaviour. Making the application issue two-factor authorisation to the actual owner. However, it is thought of as a valid trade off with usability that once in a while the application might engage the 2-factor authentication incorrectly, but alternatively the user privacy should be lowered. It is important to keep the probability of false-positives at a minimum.

Security is often reflected in actual network structure. It is for this reason crucial that the very components which is assumed secure is protected sufficiently. This paper proposed a network strategy in three different layers. An outer accessible network interface, a secure infrastructure maintaining the authentication and fraud detection components. And a final guarded layer, protecting the very data layer against internal breaches. Moreover, these data warehouses provides a simple set of commands, that is firing SQL from any component towards this data structure would simply be rejected. This report did however, not introduce actual firewall initiatives. Thus it is crucial that a well set of firewall rules is chosen for these filters. Moreover, it would be appropriate to introduce clever forensics tools [9][10]. One important measure could be application level tools, on the Web API, or even the authentication server. This could improve detection of malicious intrusion. It does however introduce performance costs, while making the subject ground its own report is recommended.

8 Conclusion

This study investigated and discussed an implementation of an online secure password store. The study grounded in usability and confidentiality. That is, the proposed solution implements wisely chosen trade-offs in order to reach the best secure performance. Not only did this study suggest discuss an implementable application. It also holds a valuable and concise Risk and Threat analysis. These subjects weights the assets at risk and the motivation behind attacks and wisely choose counter measures. In the modern digital age, this might be some of the greatest tools when protecting cooperatives against hostile entities. Reason being complexity. As institution experience grow, the assets at stake become blurred. Moreover, it might be even harder judging why any entity would be interested in that specific data. As an example it has been shown that military, top secret missions could be revealed by counting the amount of pizzas going to pentagon in the late hours.

Hence knowing your assets and their value to others is a complex but powerful tool when protecting privacy. Finally, this report was supposed to touch upon modern technology. This unfolded in a two sided application empowering only public known algorithms. And it was clarified how these techniques together can construct Authentication, Confidentiality, and Integrity even through the Internet. It is easily realised that nothing is perfectly secure, however the goal is to achieve computationally infeasibility.

References

[1] Theo de Raadt, Niklas Hallqvist, Artur Grabowski, Angelos D. Keromytis, and Niels Provos. *Cryptography in OpenBSD An Overview.* ftp://ftp.tuwien.ac.at/.vhost/www.openbsd.org/www/papers/crypt-paper.pdf.

[2] Eldefrawy, M. H., Khan, M. K., Alghathbar, K., Kim, T.-H. and Elkamchouchi, H. (2012), *Mobile one-time passwords: two-factor authentication using mobile phones.* Security Comm. Networks, 5: 508–516. doi: 10.1002/sec.340

[3] Gauravaram, P., "Security Analysis of salt‖password Hashes," in Advanced Computer Science Applications and Technologies (ACSAT), 2012 International Conference on , vol., no., pp.25-30, 26-28 Nov. 2012 doi: 10.1109/ACSAT.2012.49

[4] Willian R. Cheswick, Steven M. Bellovin, Aviel D. Rubin. *Firewalls and Internet Security, Repelling the Wily Hacker.* Second Edition, ISBN-13 978-0-201-63466-2, 2013.

[5] David R. Grantges, Jr. Gte Service Corporation. *Secure gateway having user identification and password authentication.* US6324648 B1. 27. nov. 2001.

[6] Rakesh Radhakrishnan, *Securing applications based on application infrastructure security techniques.* US 20030177390 A1. 18. sep 2003.

[7] Tom Fawcett, Foster Provost, *Adaptive Fraud Detection*, Volume 1, Issue 3 , pp 291-316. 1997-09.

[8] Sami Levijoki, Helsinki University of Technology, *Authentication, Authorization and Accounting in Ad Hoc networks.* 26th. of May 2000.

[9] Karen Scarfone, Peter Mell, NIST. *Guide to Intrusion Detection and Prevention Systems.* Special Publication 800-94. 2007.

[10] Karen Kent, Suzanne Chevalier, Tim Grance, Hung Dang, NIST. *Guide to Integrating Forensic Techniques into Incident Response.* Special Publication 800-86. 2006.

[11] Damsgaard Jensen, C *02232 Applied Cryptography, lecture notes distributed in Applied Cryptography at The Technical University of Denmark, Lyngby 2015.*

[12] Elizabeth Van Ruitenbeek, Ken Keefe, William H. Sanders *Characterizing the Behavior of Cyber Adversaries: The Means, Motive, and Opportunity of Cyberattacks.* 2010

[13] Oded Goldreich *Foundations of Cryptography* 2004, Volume 2, ISBN 978-0-521-83084-3

[14] Sonia Chiasson, P.C. van Oorschot, Robert Biddle *A Usability Study and Critique of Two Password Managers* Security 06: 15th USENIX Security Symposium