Multifactor, multiple people. Authentication approach for unlocking encrypted files.

Yuri Gorokhov *, **Lars Noergaard Nielsen** *

*University of California, San Diego

Submitted as part of graduate security course CSE227.

Proposal of a system to authenticate access to encrypted files using both Multifactor and multiple people, across different locations. Making sure that files are only accessible with the consent of all involved participants.

Multifactor | Encryption

Introduction

Controlling who has access to files is often a requirement in industry and various other contexts. Systems for dealing with information that only is accessible with multiple people's consent is therefore interesting to investigate. Software for file access control purposes include Dell Identity Manager[2], User Lock Access Manager [1] and native OS support such as an Access Control List. These systems is not addressing security as such, as not providing encryption capabilities. Common for these solutions is that file access is administered centrally by a administrator. We propose an approach were users actively set file permissions by agreeing to encrypt files by their common consent, only allowing access to these files when all parties have responded to the access request. The latter step is additionally secured by MultiFactor Authentication.

Multifactor authorization: Yubikey

Yubikey is a marketed USB dongle used for various Multifactor authentication purposes. It can be set up in different modes, for One Time Password (OTP) based on a series of variables, including sequence numbers. In this work we used the Challenge-Response mode, where the Yubikey is configured with a shared Secret Key among the server and the key itself. The Secret Key is SHA-1 cryptographic hash function.

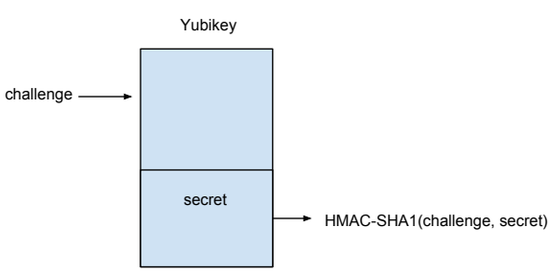

Fig. 1. Challenge-Response mode of Yubikey.

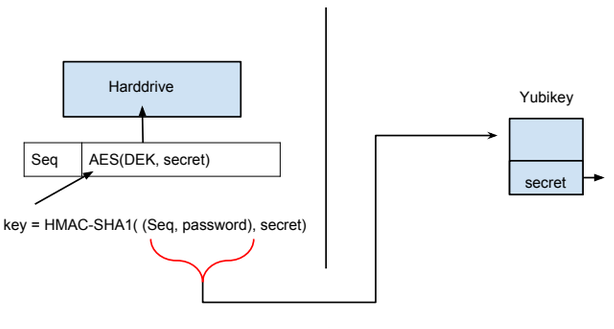

Fig. 2. Local hard drive encryption configuration.

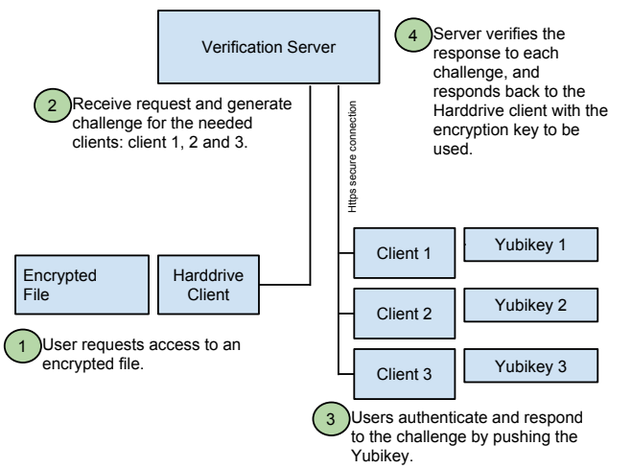

Fig. 3. System components and interaction. 4 major steps to grant access.

Yubikey furthermore provides a simple procedure for the user: only a physical touch on the device is neccesary to allow the device to respond to the presented challenge.

Reserved for Publication Footnotes

Previous work

We based our model on a proposed hard drive encryption mechanism published on the Yubikey website[3]. In their proposed configuration the Yubikey is programmed with a secret key after which it is able to perform HMAC-SHA1 encryption. The device is said to be operating in Challenge-Response mode since you can send it a challenge and it will respond with the HMAC-SHA1 encryption of the challenge with the secret key. This is depicted in figure 1.

Figure 2 shows how this mode is used for encrypting a local hard drive. The hard drive is encrypted with a Drive Encryption Key (DEK) which is stored in a table along with the secret. Both are encrypted using AES. The key used for the encryption is shown in Figure 2. Once the user enters his password a challenge is generated. The challenge consists of the password itself and a sequence number (Seq) that is also stored in the table. This challenge is sent to the Yubikey, whose response allows us to decrypt the DEK and the secret. The hard drive can now be decrypted. After decryption the DEK is re-encrypted with a new sequence number and the secret.

System Description

The proposed system is composed of a user client, hard drive client and a verification server.

A benefit from this central point of control is that the server can deny access to a file, even though participants grant access, which might be useful in some access schemes.

Encryption scheme

This sections presents the encryption scheme used, to ensure that decryption of the file is only possible when responses from all participants and their Yubikey challenges are retrieved.

Security Evaluation

We evaluate the security of the proposed system by considering various attack vectors. We leave the evaluation of specific encryption and hashing mechanisms to other work, and assume them to be computationally unreasonably hard to break. We also do not evaluate the security implications of specifically using the Yubikey product, Yubico provides a detailed evaluation on their website [4].

Server Attacks. In designing the server side component we took deliberate care not to expose enough information to compromise the security of the hard drive at any point in the future. The database on the server stores the following items:

- Yubikey secret of each user (AES encrypted)
- Key that corresponds to the user's share of an encrypted hard drive (AES encrypted)
- Mapping of hard drive to users who belong to the encryption
- Sequence number used for obfuscation

In order to decrypt the secret or the key, the attacker would need to gain access to the user's secret in order to be able to generate the decryption key via a HMAC-SHA1 algorithm. While the server does not provide any additional protection if the user secret has been compromised, it does not make matters any worse. Furthermore all users of a particular hard drive would need to be compromised in this way for a successful decryption. This is analogous to finding out everyone's password, which is stored on their YubiKey device.

Network Attacks. All communications are assumed to be SSL encrypted in a production environment to provide the first layer of security. With that in mind, let us consider what data is sent over the network during the decryption process.

Between Client and Server

When the client initiates a decryption, it sends over an **encryptionId** and a challenge. The encryptionId which is used to identify the hard drive and associated users. The challenge is a SHA hash of the encryptionId, a sequence number (seq) and an optional password. Finally the server will answer with a response.

- encryptionId
- challenge $\Rightarrow SHA(encryptionId, seq, password)$
- response $\Rightarrow HMACSHA1(challenge, SHA(\sum Keys))$

The challenge and the response leak no data, nor are they useful in a future decryption attempt since the sequence number will have changed. The encryptionId is in itself not of much use, however if somebody had also access to the server database, they would be able to find out which users are participating in a decryption. Even then, they would still need to attain the secret's of all the users to be able to decrypt their keys.

Between Server and Users

Between the server and users, only challenge and response are exchanged over the wire. These change every request due to per-user sequence numbers thus intercepting these would not lead to future vulnerabilities.

Hard Drive host attacks.

Client User Attacks.

Discussion

Discussion on strengths and weaknesses of the solution

ACKNOWLEDGMENTS. This work was supported by..

1. **http://www.isdecisions.com/lp/userlock/userlock-windows-network-security.htm?gclid=CMfPl-rqisQCFciBfgodhxwAmQ**
2. **http://software.dell.com/products/identity-manager-data-governance/**
3. **https://www.yubico.com/applications/disk-encryption/full-disk-encryption/**
4. **https://www.yubico.com/wp-content/uploads/2012/10/Security-Evaluation-v2.0.1.pdf**