A closer look at SQRL

Jos van Dijk
jos.vandijk@os3.nl

University of Amsterdam
Faculty of Science, Informatics Institute
System & Network Engineering

February 9, 2014

Abstract

Secure Quick Reliable Login (SQRL) is a draft for Single Sign-On (SSO) web authentication using mobile devices. It claims to offer properties that are not provided by related solutions. The principal design goals are to increase the privacy and security of authenticating to websites. Improved privacy is proposed by using random site-specific IDs that can't be cross-coupled. Improved security is obtained by replacing passwords with public-private key cryptography as a means of proving ownership of an identity to websites.

In this report a close look at SQRL's design details is presented and its properties are analysed. Moreover, SQRL is compared to the related SSO solutions OpenID and TiQR. Research findings show that SQRL offers a combination of properties that are lacking in related solutions. However, SQRL depends heavily on user responsibility. This dependency is a potential vulnerability that erodes its security level. In this report, vulnerabilities are investigated and countermeasures are proposed.

Contents

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Research Questions . . . 2
1.3 Related Work . . . 2

2 Background of Authentication **2**
2.1 Goals and Terminology . . . 2
2.2 Concepts and Terminology . . . 2
2.3 Architectures and Open Standards . . . 3
2.4 Implementations . . . 4
2.5 Identity Fraud . . . 4

3 SQRL **4**
3.1 SQRL Design Goals . . . 5
3.2 SQRL Operation . . . 5
3.3 SQRL Concepts and Design Details . . . 6
3.4 SQRL Properties . . . 8

4 Related SSO solutions **9**
4.1 TiQR . . . 9
4.2 OpenID . . . 10

5 Research Findings **12**
5.1 SQRL Vulnerability Analysis . . . 12
5.2 Properties & Comparison . . . 14
5.3 Extended Deployment . . . 15

6 Conclusion **16**

7 Future Work **16**

Appendices **18**

A
TiQR implementation details **18**

B
OpenID specification details **21**

1 Introduction

Authentication is the process of validating a person's identity. Authentication is closely related to authorization, the process of granting an authenticated user access rights to resources. A straightforward implementation that meets this concept, is to force a user surfing the Internet and visiting websites to create an account for each site. This approach results in a user possessing and maintaining many accounts.

During authentication a user provides credentials to prove his identity. These credentials are to be kept secret; moreover, a responsible user is assumed to create strong passwords, not to reuse passwords and to change them at regular intervals to prevent identity fraud. This method of working is hard for humans.[1] Figures show that identity fraud is big[2] and part of it is caused by mismanagement of credentials.[3]

A well-known concept to avoid management of multiple accounts is federated identity. "*Federated identity is the means by which Web applications can offer users cross-domain single sign-on (SSO), which lets them authenticate once and thereafter gain access to protected resources and Websites elsewhere*" (Eve Maler, 2008). From a user's point of view, a single identity is used to gain access to resources of all visited websites. A user is still assumed to protect his credentials and choose these wisely such that they cannot be guessed or brute-forced by an attacker. However, meeting this single requirement has proved very challenging in practice, as studies have shown numerous times that users choose poor passwords and easily disclose these in social engineering attacks.[4]

SQRL (Secure Quick Reliable Login) is a design for web authentication proposed by Gibson Research Corporation[1] in October 2013. Two factors triggered the need to come up with this design. First, the discussed ongoing demand for user authentication to websites. Secondly, the lack of a solid solution that can remedy this hardships and meets appropriate characteristics. SQRL claims to provide SSO but having characteristics that are distinct from other implementations.

Section 2 covers background information on authentication that is relevant to this project. The next section contains an in-depth study of the SQRL design, followed by the section that evaluates related solutions. In section 5 we discuss the results of our research about potential weaknesses in SQRL. We present our findings and propose measures to mitigate these attacks. Finally this report is concluded by answering the research questions and make recommendations for future work.

1.1 Motivation

Authentication is unavoidable when it comes to accessing resources. Anyone who will make use of resources finds herself dealing with this procedure. SSO is a convenient concept for simultaneously authenticating to multiple domains. Two implementation properties show off: security and anonymity. Security to avoid identity theft. Current figures show that identity theft is a big and growing problem.[2] SQRL claims to offer properties that increase security compared to related solutions. Secondly, support for anonymous login. SSO is also provided by social media like Facebook (*social login*). In this SSO implementation, social media act as an Identity Provider (IP). A website that supports social login (called *Relying Party*), relies on user validation performed by this Identity Provider and trusts it (Trusted Third Party (TTP)). Using this approach, all authentication requests are handled by this IP. However, commercial interests might encourage IPs to track and profile users. Users should be offered an alternative

[1] https://www.grc.com/sqrl/sqrl.htm

solution that allows for anonymous authentication. SQRL claims to provide anonymity with no cross-coupling of websites.

1.2 Research Questions

1. *How does SQRL improve authentication security compared to related solutions?*
   From this question the next sub-questions are derived:
   - *What does SQRL offer to the challenging and to the authenticating party?*
   - *What constraints must be met to guarantee this behaviour?*
2. *What additional features are relevant to extend deployment?*
3. *What attacks remain feasible and what countermeasures are to be considered?*

1.3 Related Work

As the SQRL draft is proposed recently, no related work in terms of papers is available. All input for discussion and implementation considerations is presented on the SQRL website.[1] However, SQRL is not the first of its kind; various SSO implementations have been developed and are being used. TiQR[2] and OpenID[3] are determined as solutions that aim for goals comparable to SQRL.
TiQR is developed by SURFnet[4]. At first glance, TiQR and SQRL have many properties in common. OpenID is a widely used Open Standards based specification. This specification relies on Identity Providers that act on behalf of the user during authentication. Both solutions are included in this project.

2 Background of Authentication

The purpose of authentication and properties of concepts, designs and implementations need to be addressed before any comparison among various forms can be performed.

2.1 Goals and Terminology

Authentication is closely related to *Identification*. *Identification* is the process of determining the identity of a user. An example of revealing a person's identity is to provide a username.
Authentication is the process of validating the presented identification information: is the person really the person he claims to be? An example of proving a person's identity is to provide *credentials*. A *credential* is a secret, known only by parties involved in an authentication process. Authentication is a prerequisite for *Authorization*. *Authorization* is the process of granting (limited) access to resources to an authenticated user.

2.2 Concepts and Terminology

At least two parties are involved in an authentication process: the *Supplicant* being the entity that has to be validated and the *Authenticator*, being the party that performs the validation. In this setup, the data needed for authentication is supplied to both parties.

Single Sign-On (SSO): a concept that, from the user's point of view, allows for a single identity

[2] https://tiqr.org/
[3] http://openid.net/
[4] http://www.surf.nl/en/about-surf/subsidiaries/surfnet

to be used for authentication.

Trusted Third Party (TTP): a concept that introduces an additional party that acts on behalf of the user during authentication.

Anonymous authentication: the process of validating a user's access rights to resources without revealing personal information.

Multiple Factor Authentication (M-FA): denotes the level of security. Providing multiple credentials of different types is considered a stronger proof of an identity because it is harder to compromise multiple credentials. Three well-known levels are in use

1-FA, uses a single credential. Common implementations use a factor the user *knows* (password, PIN, etc.)

2-FA, adds a second credential. Common implementations use a factor the user *has* (token, digital certificate, etc.)

3-FA, adds a third credential. Common implementations use a factor the user *is* (biometrics: fingerprint, retinal scan, etc.)

Hash is the result of a hash-function. "*A cryptographic hash function is a mathematical transformation that takes a message of arbitrary length (transformed into a string of bits) and computes from it a fixed-lenght (short) number*". (Kaufman, et al. Network Security). Hashes are often used to prove identity.

Out-Of-Band (OOB) authentication: adds an additional communication channel during authentication. OOB improves the level of security as compromising multiple communication channels at the same time interval is considered more difficult.

2.3 Architectures and Open Standards

Concepts need to be transformed into implementations. Development of open architectures and Open Standards improve interoperability and prevent from technology and/or vendor lock in. Below, architectures and open standards regarding authentication, relevant to this project, are discussed briefly.

Architectures
OATH Reference Architecture, Release 2.0 is a high-level technical description of a framework for open authentication, proposed by OATH, the Initiative for Open AuTHentication[5].

OpenID Authentication 2.0 - Final is an open high-level protocol description that provides a way to validate a user's identity, proposed by the OpenID Foundation[6].

Public Key Infrastructure (PKI) is an architecture to enable to create, manage, distribute, use, store, and revoke digital certificates.

Open Standards
OCRA: OATH Challenge Response Algorithm (`rfc6287`), developed by OATH .

[5] http://www.openauthentication.org/files/download/oathPdf/ReferenceArchitectureVersion2.pdf
[6] http://openid.net/foundation/

HOTP: An HMAC-Based OTP Algorithm (`rfc4226`), developed by OATH .

X.509 certificates (`rfc5280, rfc6818`): an ITU-T standard for a Public Key Infrastructure (PKI).

SSL/TLS (`rfc5246`): cryptographic protocols which are designed to provide authentication using X.509 certificates.

HMAC (`rfc2104`), *SHA256* (`NIST`) are hash algorithms.

2.4 Implementations

Both Open Source and commercial solutions of authentication concepts are available whether or not implementing open standards. Most relevant open solutions, relevant to this project are OpenID and TiQR.

OpenID is an open standard SSO solution that allows users to be authenticated by certain co-operating sites (known as Relying Parties or RP) using a third party service. As this TTP might be an online networking service, this type of authentication is associated with *social-login*.

TiQR is an open source authentication solution for web sites intended to be used with smartphones. TiQR is designed to benefit from smartphones by using the built-in camera. A QR-code is used to represent the data that initiates the authentication process. On this point, TiQR and SQRL match.

2.5 Identity Fraud

Identity Fraud is stealing and subsequently using a person's identity. As said before, identity fraud is big. Theft is caused by vulnerabilities of the authentication process. Attacks exploit these vulnerabilities. There is not a single spot in this process to be indicated that causes vulnerabilities. Common areas are

Design errors have a huge impact as it is not likely to come up with a quick work-around to mitigate the threat. A redesign and a new implementation will be the result leaving the user to choose for an alternative in the meantime.

Implementation errors are caused by hardware and/or software developers that implement the design. Poor input-data validation is a common implementation error.

Human errors play an important role in identity fraud. Users are often unaware of the techniques used by identity thieves. Malware and social engineering are serious threats with regard to identity fraud.[7][5, 6]

3 SQRL

SQRL is a draft for SSO web authentication using mobile devices. It uses QR-codes to represent authentication data.

[7] http://www.kindsight.net/en/blog/tags/identity-theft

3.1 SQRL Design Goals

The SQRL design goals are exemplified by Steve Gibson during a `SecurityNow` podcast.[8] An overview of the most relevant properties quoted, are listed below:

- there is a need for anonymous online authentication. Users should be able to decide themselves whether they are trackable on the Internet
- no obvious connections among different websites, to avoid profiling
- it should provide 2-FA, using separate channels (out-of-band) to assure an improved level of security
- no exchange of pre shared secrets, to avoid server security dependency
- it should provide SSO because multiple accounts are hard to handle and mandatory account creation distract users from websites
- no keyboard interaction during authentication, to allow use of public devices
- no TTP involvement, to avoid security dependency
- it should be free, not complex, to be used side by site to alternatives, to encourage adoption and deployment

3.2 SQRL Operation

Three modes of operation are supported as shown in figure 1. The leftmost scenario is discussed as it uses most of SQRL's properties. It is assumed that the user's smartphone has a SQRL-app installed.

The presented 'QR scanning' mode allows for using an untrusted device. Reasons for this setup

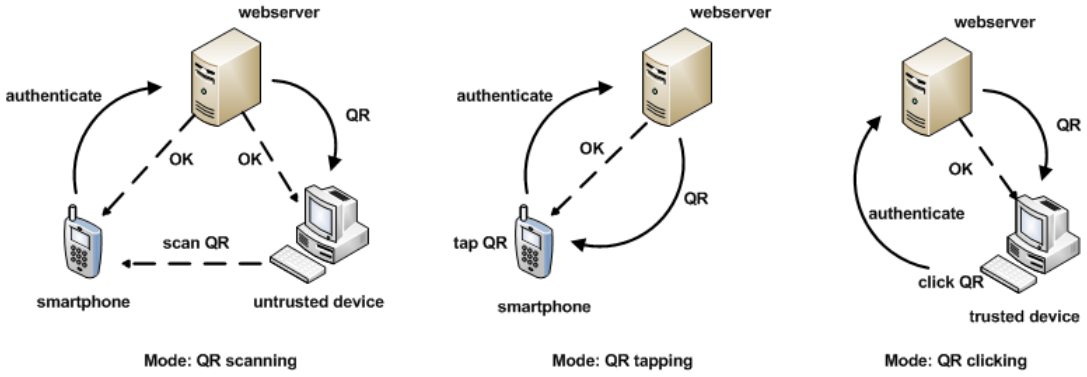

Figure 1: SQRL modes of operation

might be limited bandwidth and/or battery depletion on the smartphone. The user initiates the process by clicking the SQRL Login button presented on the visited SQRL enabled website, using the untrusted device. The website returns a QR-code visible on the untrusted device's screen. This QR-code is scanned by the SQRL-app. Subsequently, the app extracts and displays the Full Qualified Domain Name (FQDN) of the authenticating service from the QR-code. The user confirms to authenticate using this site and enters his SQRL secret password. The user is authenticated and granted access. The app shows a successful authentication message and the untrusted device's browser is unlocked and refreshed. Both remaining modes 'QR Tapping' and

[8] http://twit.tv/show/security-now/424, starting from 37.15 minutes

'QR Clicking' are used in more straightforward setups and then some security guarantees are limited.

3.3 SQRL Concepts and Design Details

Concepts The SQRL design proposes a Challenge Response authentication mechanism to be implemented. In this widely used concept, a server issues a challenge that needs to be responded by a client (figure 2). This response is used to validate the identity of the client. A common concept for this scheme is public key cryptography. Implementation proposals use public as well as secret key cryptography, one-way hashing and rely on a secure communication protocol.

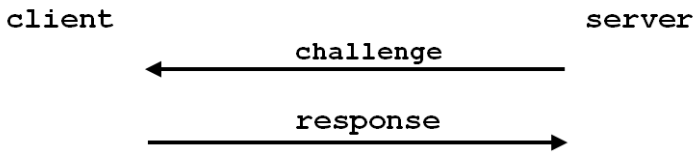

Figure 2: Client - Server Challenge Response concept

Design Details The composition of the building blocks used, is shown in figure 3.

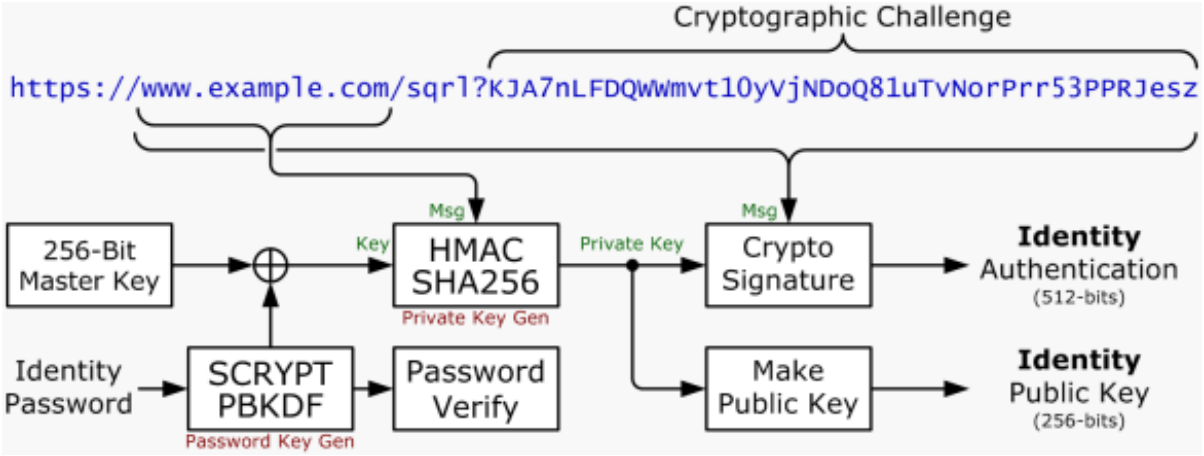

Figure 3: SQRL building blocks. `Taken from https://www.grc.com/sqrl/crypto.htm`

A 256-Bit Master Key represents the 'proof of possession' authentication factor. The algorithm used to provide this key is expected to generate a maximum-entropy key. The second ('proof of knowledge') authentication factor is represented by the Identity Password. This is the single key applied by the user. As it is frequently used, extra attention is to be paid to management of this key in terms of strength, lifetime and secrecy. Both factors are used to assemble the 256-bit random value called Key and represents the never changing user's SQRL ID.

A SQRL enabled website shows a QR-code that contains the FQDN of the authenticating service. This unique string is fed to the Msg-input of the HMAC-SHA256 building block. This function outputs a keyed Hash Message Authentication Code (HMAC), a 256-bit value, unique for the (Key,Msg) pair. Due to the fact that this value is site-specific, it is to be of use in public key cryptography, representing a Private Key. The SQRL-app uses this private key to create

a signature (called **Identity** Authentication) of the entire URI. This signature is sent to the server as part of the response.

To be able to verify the received signature, the server needs the associated public key (**Identity** Public Key). The Make Public Key building block calculates the associated public key using Elliptic Curve Cryptography (ECC). This public key is sent alongside the signature as a response of the server's challenge. A message sequence diagram reflecting the 'QR Scanning' mode is provided in figure 4. This diagram is of use when comparing SQRL to related solutions.

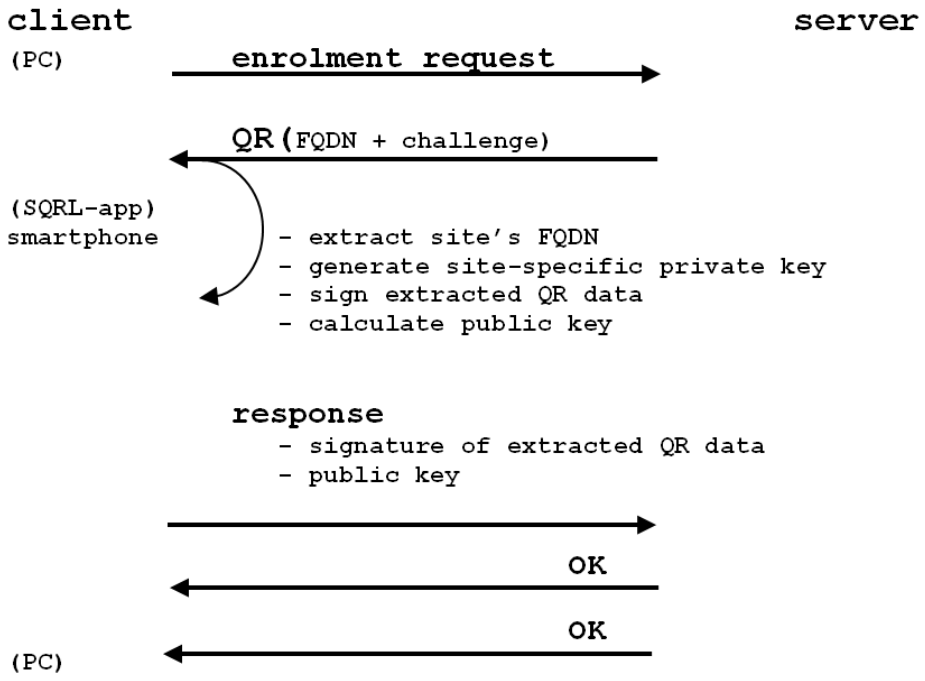

Figure 4: SQRL message sequence diagram

The SCRYPT PBKDF and Password Verify building blocks implement additional security. As said before, the Identity Password represents the user's SQRL password ('know factor'). Offline password attacks are one out of many attack schemes used to compromise a user's identity. A PBKDF (Password Based Key Derivation Function) is a function that deliberately causes a huge computational load. The goal is to slow down password cracking processes. SCRYPT (usually in lowercase) is an enhanced implementation of a PBKDF as it also forces huge memory consumption[9].
The fixed SQRL ID depends on a changing component, as a responsible user will change the Identity Password at regular intervals. This is solved by adjusting (XOR'ing) the 256-bit Master Key whenever a user changes his SQRL secret password.
Note: the presented flow for this XOR operation is not reflected in figure 3. Also note that the SQRL secret key is not related to the key that allows access to the smartphone itself.
The Password Verify building block stores 128 bits of a hash of the SCRYPT PBKDF output. Whenever a user enters his SQRL secret key, SCRYPT calculates the 256-bit value which will be hashed and verified (128 bit) by Password Verify.

[9] http://www.tarsnap.com/scrypt/scrypt.pdf

ID Revocation is an important supported feature. Because of possible vulnerabilities, attacks may succeed in stealing a person's SQRL identity. As soon as fraud is suspected, the ID has to be locked and subsequently unlocked and/or revoked by the authentic user. These latter operations are not available to the thief. A brief description of this procedure is given here. Details can be found on SQRL's site.

A compromised identity is an emergency situation. Quick response is needed but should not be at the expense of security. For this reason different situations have to be distinguished and dealt with.
Resources needed to immediately lock a SQRL ID are permanently provided on the client as well as on the server side. This operation can be performed by any user who has access to the installed SQRL-app. However, resources to unlock and/or revoke a SQRL ID are partially stored on a safe place by the authentic user during initialisation.

To enable these features, additional key-pairs are (re)generated. For a user of a compromised ID to prove its authenticity, a pre shared key is used. A (super) secret (private) key is involved in this pre shared key calculation. The user is urged to print this key on paper and keep it on a safe place. If this requirement is met, an identity thief is not able to (re)generate this pre shared key, effectively not able to unlock or revoke the SQRL ID.
Diffie-Hellman (DH) is used to generate this pre shared key. In normal DH operation both parties, client and server, are involved in the pre shared key computation. The SQRL implementation differs in the sense that the server side is not involved. The very first time a user issues a SQRL login request to a website, keys are generated that allow for lock, unlock and ID revocation regarding this single website. It goes without saying that only public keys are stored on the server side and public keys are not shared among websites to avoid cross-site coupling of keys. This enforces anonymity. A drawback of this requirement is the need for regenerating keys because it is unfeasible to store revocation private keys for all visited websites on the client. It comes down to signing and verifying a signed unlock and/or change identity request using keys that are derived from the (super) secret (private) key owned by the authentic user and using the pre shared key.

3.4 SQRL Properties

From this analysis, properties can be given to the design.

- SSO: From a user's point of view, SQRL offers SSO behaviour as the user provides the same single password during arbitrary websites authentication.
- 2FA: SQRL offers 2FA: during authentication, a user has to provide a credential he *knows*. The second factor is stored in the SQRL-app that the user *has*.
- OOB: SQRL does offer Out-Of-Band authentication as two different communication channels are supported during authentication.
- Pre shared keys exchange: SQRL does not exchange pre shared keys. Only public keys are exchanged.
- TTP: SQRL does not introduce additional Trusted Third Parties. The only TTP involved is part of the communication channel protocol TLS.
- Anonymity: The user's ID is represented by a 256-bit random value (Master Key). Site-specific keys (ID's) are generated from this Master Key. The HMAC-SHA256 algorithm assures that keys are not related. Cross-coupling of ID's is not to happen. From this point of view, anonymity is provided.
  Note: SQRL resides on the Application Layer of the network communication protocol

stack. In network communication, many more layers are involved which might reveal identity information.

- ID revocation: SQRL supports lock, unlock and identity change. To enable this, additional keys are generated during initialisation and some of them are to be regenerated upon use. To take anonymity into account, site-specific keys are generated.
- Low friction deployment: The current draft is open and based on scientific findings. These principles ensure that there are no barriers that will hold up adoption and deployment.

4 Related SSO solutions

4.1 TiQR

TiQR is an open authentication solution for smart phones and web applications[10] developed by SURFnet[11]. QR-codes, scanned by the smartphone's camera, are used to represent login and authentication data. Additional properties of TiQR are presented now briefly. Detailed information can be found in AppendixA.
TiQR is based on OCRA (OATH Challenge Response Algorithm), an algorithm intended for challenge response authentication and specified in `RFC6287`.[7]
The underlying protocol used by OCRA is the HMAC-based One Time Password algorithm, specified in `rfc4226`.[8]
The protocol name reveals its purpose: generating an OTP. A hash is computed using SHA1 as the algorithm and a shared secret key K and a synchronized counter C as input values. The result is truncated to a convenient size a user can handle (6 - 8 digits) as shown in figure 5. OCRA implements HOPT in a more general way. The counter C has been replaced by a set of

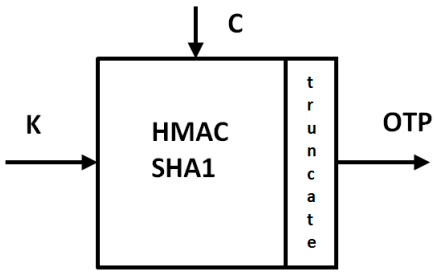

Figure 5: Block diagram HOTP, HMAC-based One Time Password generation

parameters, some of them being optional, including the original counter C. The protocol allows for negotiation. One of these parameters is mandatory: a challenge Q. The value of Q and all negotiated options are concatenated as a string and fed to the HOTP function-input C. For 2-factor authentication, a password hash is part of this string. OCRA offers no mechanism for secure exchange of secrets.

TiQR implementation TiQR-app's are available for Android and iOS. For every TiQR enabled website, a user needs to create a TiQR account. Besides a username and email address a 4-digit secret key (PIN) is shared.[9] The current implementations don't allow for changing this PIN. A secure connection is required to protect the exchange of account data.

[10] https://tiqr.org/
[11] http://www.surf.nl/

Using TiQR is straightforward: a user visits a TiQR-enabled website and requests a TiQR login. The website generates and displays a QR-code holding a challenge which will be read by the TiQR-app. The user enters the secret PIN and the app calculates and returns the hash. The authenticating service verifies the hash and authorizes the user. Upon success, the user is logged in.
For comparison purposes, a TiQR message sequence diagram is drawn in figure 6. This implementation composes of two handlers: an *enrolment handler* and an *authentication handler.*[10] It will be used in section 5.

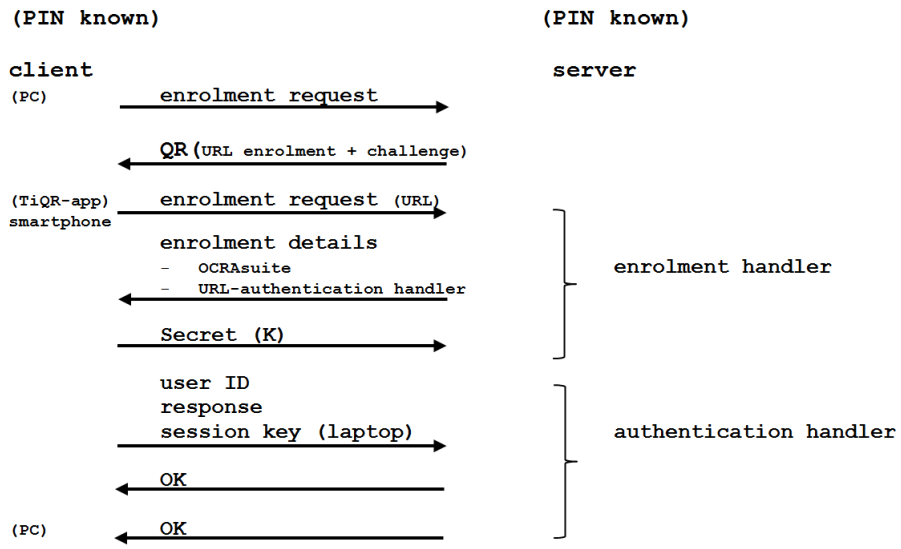

Figure 6: TiQR message sequence diagram

4.2 OpenID

OpenID is an open standards specification for authentication, intended for users that want to prove their identity to websites. A brief description of operation is presented below. Detailed information is given in Appendix B.
Users first create and register their ID with an OpenID Provider (OP). This provider acts as a TTP. An OpenID-enabled website contacts the OP upon a user initiated authentication request. In this context, OpenID-enabled websites are called Relying Parties (RP). The specification supports use of multiple ID's with multiple OP's per user, but a single ID will do. The latest OpenID specification is `OpenID Authentication 2.0`.[11]
The specification presents a high-level protocol description. For implementation details like hash functions and secret key generation, recommendations are presented. HTTP(S) is the single mandatory protocol. OpenID uses HTTP over TLS to protect information exchange. The specification merely covers implementation constraints, for example data formats and communication types.

OpenID implementation The user's view of creating an OpenID account is not different from any other account except for the name which represents a URL. A created OpenID account might look like `os3uva.clavid.com`
The method used for authentication to the OP is not part of the specification. Passwords and

TiQR are valid options, shown in Appendix B. To be able to compare the various solutions, an OpenID message sequence diagram is derived from the specification and shown by figure 7.

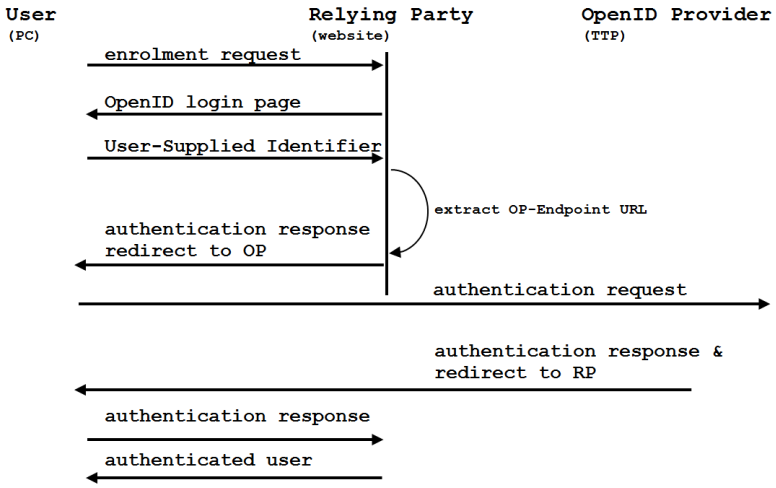

Figure 7: OpenID message sequence diagram

5 Research Findings

Approach To be able to determine SQRL's vulnerabilities, information is retrieved on causes that affect vulnerabilities. Results from audits and research are important sources here.[12, 13, 14]
Three main categories are to be distinguished: *design-*, *implementation-* and *user errors*.

Design errors cause a component to fail its specifications. Weaknesses belong to this category. Weaknesses in the MD5 hash algorithm for example, cause collisions. In case a design error can't be repaired, implementations become of no value.

Implementation errors are caused by hardware and/or software developers that make wrong decisions. A TiQR implementation audit[12] shows that types of errors made are extensive.

User errors reflect the user's behaviour. Authenticating users have to be educated and informed for all possible threats. Security audits and best-practice show[12][12] that some attacks can be mitigated but risks remain as they depend on the user's behaviour.

5.1 SQRL Vulnerability Analysis

The SQRL design analysis provided in section 3.3 shows that all building blocks are composed of standards that are currently in use. At this moment no weaknesses or flaws are known that might erode the SQRL security properties. These include:

- TLS: SQRL relies on TLS as a secure communication channel. A proper implementation and an attentive user repel MiTM attacks and eavesdropping. The validated certificate ensures that no malicious party is involved. The user has to confirm this validation as phishing is widespread.
- Master Key: Creating a duplicate of a 256-bit random value is not considered feasible.
- HMAC-SHA256: Creating a duplicate of a 256-bit random value is not considered feasible. There is no reverse operation that will reveal the Master Key.
- Public Key: Use of asymmetric key encryption is considered secure.
- scrypt PBKDF: will slow down a brute force attack considerably.

It is not obvious that SQRL's vulnerabilities are to be found in its design.

Currently, there is no mature implementation of the SQRL draft. A vulnerability analysis of this category can't be done yet.

User errors play an important role in security threats too often. Security audits confirm this observation. Common identified threats are:

- Malware: Jailbroken devices are more susceptible to downloading malicious apps. But even healthy devices suffer from these attacks.[13] Malware that is able to penetrate the SQRL-app or eavesdrops the import and/or export of keys, compromises the user's SQRL ID. This is the very worse scenario as this user is out of control.

[12] http://wiki.openid.net/w/page/12995200/OpenID
[13] https://www.security.nl/posting/40820/Malware+op+Google+Play+besmet+miljoenen+Android-toestellen

- Shouldersurfing: A person or camera viewing user interaction causes the loss of an authentication factor. An Identity Password that is known by anyone who benefits from identity theft, will encourage that person to attempt to steal the smartphone. However, the key that allows access to the smartphone itself is needed as well.
- Stolen devices: are susceptible to offline brute-force attacks but may also be part of the shoulder surfing attack.
- Phishing: might provide an opportunity for Man-In-The-Middle attacks. This attack is feasible. An identity thief may request a SQRL challenge (valid operation). Next this QR is shown on a page that mimics an authentic site that is frequently visited by the user. If this login page is presented to a negligent user that accepts the challenge, provides his Identity Password and submits, causes the thief's browser to unlock and the thief is logged in.

It is obvious that user errors have serious impact on SQRL's secure operation. All identified threats mentioned above may cause losing secrets thus causing identity theft or impersonation. In the next paragraph, countermeasures to mitigate these attacks are proposed.

Countermeasures The most severe threat is malware. In a worst case scenario a thief takes over control and is able to impersonate the victim.
A solid solution would be using an additional feature of modern smartphones: nfc-capability. These devices support Near Field Communication, a standard for short distance wireless bidirectional communication. The SQRL crypto design is to be implemented in a nfc-tag which holds a crypto processor. This setup is presented in figure 8. Using a PC instead of a smart-

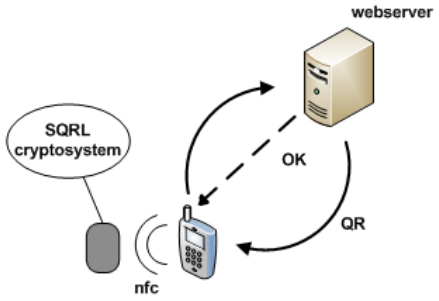

Figure 8: SQRL crypto system executed by a nfc tagged crypto processor

phone, a similar setup is proposed. Here, the SQRL crypto system is transferred to a smartcard which contains a crypto processor. This solution is presented in figure 9.

Substantial research has been conducted on graphical passwords. On the one side graphical passwords are considered to be more memorable than strong passwords and possibly more vulnerable to shoulder surfing on the other.[15, 16]. PassFaces.[14] is an example of a graphical password implementation.

Brute-force attack countermeasure is part of the SQRL design. The `scrypt-PBKDF` thwarts this attack by enforcing huge cpu power and memory consumption during key derivation.

[14] http://www.passfaces.com

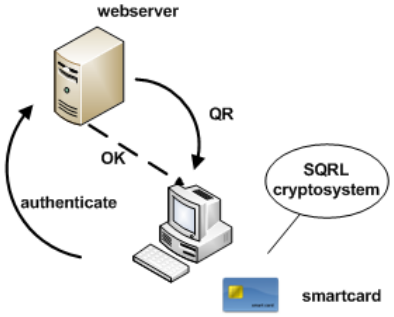

Figure 9: SQRL crypto system executed by a smartcard crypto processor

Phishing can be mitigated by user awareness in conjunction with maintaining a clear enrolment and authentication procedure. Explicitly showing the FQDN of the authenticating service is crucial. Setting up a personalized icon on the the login screen will contribute as well. However, this doesn't fit to the anonymity property.

5.2 Properties & Comparison

The properties of SQRL are to be compared to be able to determine how SQRL differs from related solutions and to denote whether these differences contribute to improved authentication security. Properties that are relevant to this question are collected and presented in figure 10. This overview shows that just a few conceptual OpenID properties can be measured. The remaining characteristics are implementation dependent. However, conclusions can be drawn.

- No secret(s) exchange: TiQR relies on shared keys as shown in figure 6: a `PIN`, exchanged during account creation and a user initiated session key `K` as part of the authentication. SQRL has no such exchange of secrets. Both implementations rely on TLS as underlying secure channel. A proper implementation of TLS is considered secure. The process of secrets exchange does not degrade the security level. However, storage of secrets is an additional aspect. A webserver that stores private authentication information introduces a threat as it might get compromised. Depending on the user's choice of the number of identities (single or multiple), a compromised webserver might cause loss of a global identity. Not sharing secrets avoids this threat.
- Anonymity: An email address is part of a TiQR account. This means is intended for account recovery in case of user lock out. An email address adds up in the amount of information that is known about a person. In this way it decreases anonymity. SQRL's ID revocation process is exercised by using keys.
  Note: a user has to keep in mind that anonymity is relative. Despite the fact that SQRL ID's are represented by random values, identity information might be revealed. This is caused by the network communication involvement. SQRL is just an application, part of a protocol stack that allows for inter network communication. Data, needed to establish communication might be linked to the user's identity.
- No (additional) TTP: This relates to 'No secret(s) exchange'. A party might get compromised. Figure 7 shows that an additional TTP is part of OpenID's concept. Lack of a TTP increases security with respect to TTP compromisation.

From this overview and analysis the research subquestions can be answered as well.
The 'SQRL column' of figure 10 shows the properties offered. Figure 11 shows what SQRL

offers to the challenging and to the authenticating party. Some implementation details are omitted here because a user's perception is presented. The dependencies that SQRL relies on

	TiQR	OpenID	SQRL
SSO	✓	✓	✓
2FA	✓	?	✓
OOB	✓	?	✓
No secret(s) exchange	X	?	✓
Anonymity	✓ ?	?	✓
No (additional) TTP	✓	X	✓
Low friction deployment	✓	✓	✓
ID revocation	✓	?	✓

Figure 10: SQRL properties compared to TiQR and Open ID

SQRL	User	Website
SSO	✓	
2FA	✓	
OOB	✓	
Anonymity	✓	
ID revocation	✓	
Low friction deployment	✓	✓
Authenticated identity		✓

Figure 11: SQRL properties offered to user and website

to guarantee its behaviour are TLS, as a secure communication channel and a responsible user; a user that is aware of techniques that exploit user vulnerabilities.

5.3 Extended Deployment

One of SQRL's key design goals is to provide anonymity. This prevents user profiling and tracking. To meet this property the SQRL design provides site-specific keys that exclude cross-coupling of ID's. This characteristic can be used to extend utilization. Anonymous login is not always applicable. Internet shopping, membership of an organisation, participation registration or email verification demand for additional information that deliberately reveals a persons identity. SQRL can be of use in this area as well because of its 'site-specific-keys' property. Figure 12 shows this side-by-side operation.

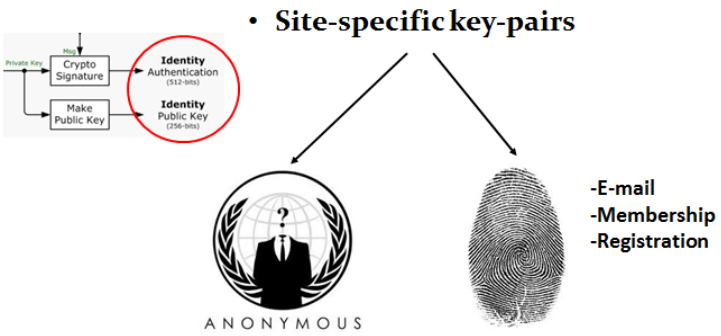

Figure 12: SQRL extended deployment: anonymous as well as identified authentication

6 Conclusion

During this project, a close look has been taken at SQRL, a draft for secure web authentication using modern devices like smartphones. The focus was on authentication security, extended deployability and distictive properties.

Analysis shows that the main vulnerability is caused by user errors. A SQRL user has to keep two fixed secret keys on a safe place that should not be connected to a network. A third key, being the user's PIN, is frequently used and also to be kept private. Besides storage, use of these secret keys by import and/or export operations may cause identity theft due to devices that are infected by malware. Negligent users are susceptible to phishing and shoulder surfing. Countermeasures that relieve users from burdensome key management and cuts off malware are implementations using an additional secure environment. Implementing the SQRL crypto system in a smartcard or nfc-tag that contains a crypto processor, will improve security regarding key management.
Secondly, promoting user awareness will contribute to a higher level of user responsibility. Users that have a clear picture of identity fraud are less susceptible.
Vulnerabilities caused by implementation errors are not part of this project. Due to the relative recent introduction of the SQRL draft (October 2013), no mature implementations exist yet.

SQRL aims for anonymity. Due to this design goal, the draft is suited to authentication that deliberately discloses a person's identity. The design allows for side by side operation of both authentication modes.

A comparison of SQRL to related solutions TiQR and OpenID shows that SQRL is distinct. It's not the individual properties that make the difference but the combination. Both TiQR and OpenID support some SQRL properties but not all of them. The most distinct combination of properties provided by SQRL is: no secret key exchange, no TTP, anonymity.
Users should be aware of the limitations of anonymity on the Internet. Many more components are involved in network communication and many of them may contribute to disclosure of user identity information.

7 Future Work

Currently, no mature implementations of SQRL are provided. Due to this, the impact of implementation errors can't be determined. However, history shows that many vulnerabilities come from this area. Security audits are to be performed on both app and server implementations.

Devices infected by malware being used for authentication allow for identity theft. Practice shows that malware is persistent. This justifies the need for a secure environment. The proposed solutions need a follow up.

SQRL supports ID revocation in case of a compromised identity. The current draft has no automated procedure on 'lock' and 'change identity' operations to be performed on all visited websites.

References

[1] Kirsi Marjaana Helkala and Tone Hoddø Bakås. National password security survey: Results. 2013.

[2] Javelin Strategy. Research.(2012, february 22). 2012 identity fraud report: Social media and mobile forming the new fraud frontier.

[3] SM Haque, Matthew Wright, and Shannon Scielzo. A study of user password strategy for multiple accounts. In *Proceedings of the third ACM conference on Data and application security and privacy*, pages 173–176. ACM, 2013.

[4] Dinei Florencio and Cormac Herley. A large-scale study of web password habits. In *Proceedings of the 16th international conference on World Wide Web*, pages 657–666. ACM, 2007.

[5] Android malware growth and possible botnet. *Network Security*, 2012(7):1 – 2, 2012.

[6] Android malware goes undetected. *Computer Fraud Security*, 2012(3):3 –, 2012.

[7] D MRaihi, J Rydell, D Naccache, S Machani, and S Bajaj. Ocra: Oath challenge-response algorithms. Technical report, IETF RFC, 2011.

[8] D MRaihi, M Bellare, F Hoornaert, D Naccache, and O Ranen. Hotp: An hmac-based one-time password algorithm. *The Internet Society, Network Working Group. RFC4226*, 2005.

[9] Jan Michielsen. tiqr user manual. Technical report, 2011.

[10] Roland M Van Rijswijk and Joost Van Dijk. tiqr: a novel take on two–factor authentication. In *Proceedings of the 25th international conference on Large Installation System Administration*, pages 7–7. USENIX Association, 2011.

[11] specs@openid.net. Openid authentication 2.0 - final. Technical report, 2007.

[12] Roland van Rijswijk and Joost van Dijk. Tiqr-security-audit-report-v1.1.pdf. 2011.

[13] Manuel Uruena and Christian Busquiel. Analysis of a privacy vulnerability in the openid authentication protocol. *IEEE Multimedia Communications, Services and Security*, 2010.

[14] Pavol Sovis, Florian Kohlar, and Jörg Schwenk. Security analysis of openid. In *Sicherheit*, pages 329–340, 2010.

[15] Haichang Gao, Wei Jia, Fei Ye, and Licheng Ma. A survey on the use of graphical passwords in security. *Journal of Software (1796217X)*, 8(7), 2013.

[16] Furkan Tari, Ant Ozok, and Stephen H Holden. A comparison of perceived and real shoulder-surfing risks between alphanumeric and graphical passwords. In *Proceedings of the second symposium on Usable privacy and security*, pages 56–66. ACM, 2006.

Appendices

A
TiQR implementation details

TiQR is an open authentication solution for smart phones and web applications[15] developed by SURFnet[16].

Open Standards TiQR is based on OCRA (OATH Challenge Response Algorithm), an algorithm intended for challenge response authentication and specified in `RFC6287`. At an abstract level, OCRA can be denoted by

$$OCRA = CryptoFunction(K, DataInput)$$

K = shared secret Key

DataInput = concatenation of various input data values

CryptoFunction = the function performing the OCRA computation

For the CryptoFunction the HOTP (HMAC-based One Time Password) algorithm is used. HOTP is specified in `RFC4226`. At an abstract level, HOTP can be denoted by

$$HOTP(K, C) = Truncate(HMAC - SHA - 1(K, C))$$

K = shared secret Key

C = (incrementing) counter

Truncate = algorithm reducing the HMAC-SHA-1 result to 6 (up to 8) digits

HMAC = keyed Hash Message Authentication Code. A hash function is used to calculate a value based on C and K. The hash function used is SHA-1 (RFC3174)

Combining both formulas present a more clear view of the OCRA output.

$$OCRA = Truncate(HMAC - SHA - 1(K, DataInput)$$

The result of this computation would be an OTP consisting of 6-8 digits. The expected output size is recorded in the OCRAsuite. This set of possible outputs include all supported hash functions as well.

Besides that, the formula shows that OCRA's use of HOTP is more general. The *DataInput* parameter consists of several options that are to be negotiated. This allows for wide use like applying a challenge. Available options are

C = counter that is equivalent to the HOTP parameter. This feature is optional

Q = challenge needed for the authentication. This feature is mandatory

P = a hash value of PIN/password that is known to all parties. This feature is optional

S = a UTF-8 encoded string that contains information about the current session

T = a timestamp. This feature is optional

OCRAsuite = a textstring holding the OCRA mode of operation and options that are included in the computation

[15] https://tiqr.org/
[16] http://www.surf.nl/

An OCRAsuite specification that provides 2-FA might look like:`"HOTP-SHA1-0:PSHA1"` which represents

HOTP-SHA1 = hash function used to compute the hash value

0 = no truncation (full sized hash value)

PSHA1 = secret password hashed by SHA1 function

Figure 13 shows a simple message diagram of a one-way authentication using OCRA.

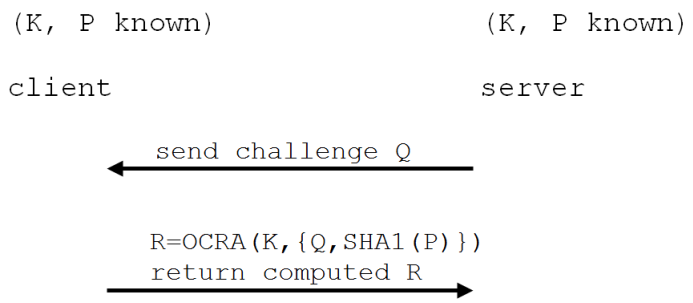

Figure 13: OCRA-message-diagram

OCRA offers no mechanism for secure exchange of K and P.

TiQR implementation TiQR-app's are available for Android and iOS.

Account creation For every TiQR enabled website, a user needs to create a TiQR account. Using his computer, a user visits the website and enters a user ID in the TiQR account creation area. The webserver will generate and show a QR-code on the computer's display, representing this account. The user is expected to read this QR-code using the TiQR-app on his smartphone and to confirm the account activation. Next the user has to enter a 4-digit secret PIN. Now the TiQR account is activated and ready for use. In cases a user decides to create multiple TiQR ID's, it is recommended to share the secret PIN among ID's to avoid confusing. In the current implementation it is not possible to change this PIN.
A HTTP over TLS connection is required for this account creation process to protect the exchange of secrets. After account creation, the user ID as well as the PIN (= P) is known by the server.

Using TiQR is straightforward

- a user visits a TiQR-enabled website
- website generates and displays a QR-code holding a challenge
- user uses TiQR-app to read the QR-code
- user enters secret PIN
- TiQR-app computes hash according to the OCRAsuite, K, challenge Q and the hashed P and returns this response
- (authenticating part of the) website verifies the response and authorizes the user if positive.

- the user is logged in

The message diagram for this TiQR implementation is given in figure 14. This implementation composes of two handlers: an *enrolment handler* and an *authentication handler*

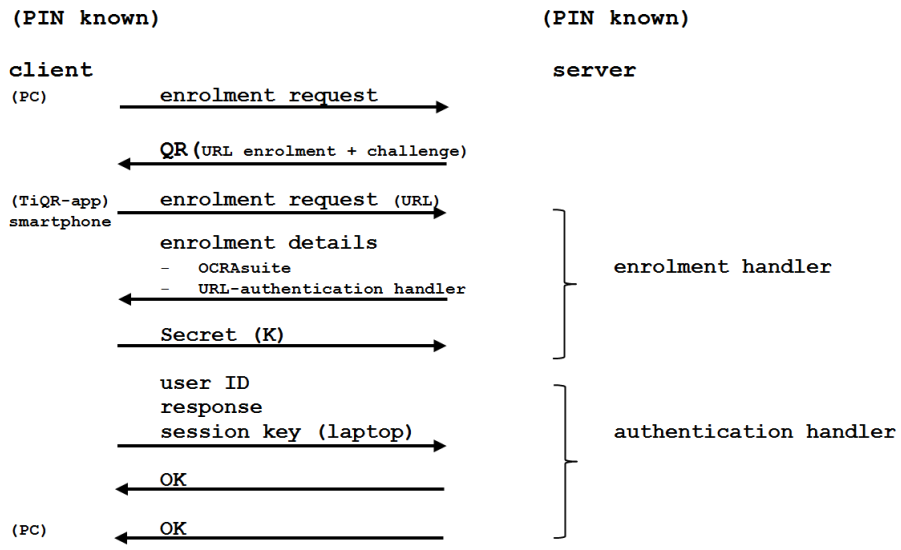

Figure 14: TiQR message sequence diagram

B
OpenID specification details

OpenID is an open standards specification for authentication, intended for users that want to prove their identity to websites. Users first create and register their ID with an OpenID Provider (OP). This provider acts as a TTP. An OpenID-enabled website contacts the OP upon a user initiated authentication request. In this context, OpenID-enabled websites are called Relying Parties (RP). The specification supports use of multiple ID's with multiple OP's per user, but a single ID will do. The latest OpenID specification is `OpenID Authentication 2.0`.

Open Standards HTTP(S) is the single mandatory protocol. OpenID uses HTTP over TLS to protect information exchange. For implementation details like hash functions and secret key generation, recommendations are presented. The specification merely covers implementation constraints, for example data formats and communication types.

OpenID specification The terminology that is needed to understand the message diagram is given here.
User-Supplied Identifier: ID presented by the user to the Relying Party.
OP-Endpoint URL: URL which accepts OpenID Authentication protocol messages.

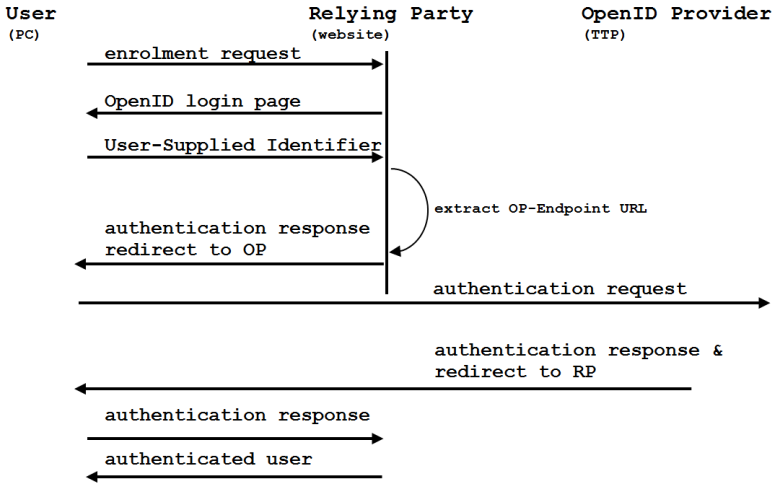

Figure 15: OpenID-message-diagram

The manner in which the user authenticates to the OP is not part of the specification.

Account creation can be done at any, free to choose, OpenID Provider. One of them is `Clavid AG`, a Swiss identity provider[17]. An OpenID is a unique URL that captures the user ID as well as the authenticating OP. Depending on the implementation, a user has to provide at least a user ID as mandatory information.
A created and registered OpenID with Clavid AG might look like `os3uva.clavid.com` As shown in figures 16 and 17 for this OP a first name, last name and email address are mandatory extra fields.

[17] http://www.clavid.com

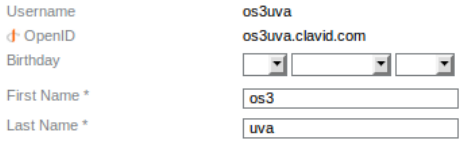

Figure 16: Clavid mandatory fields

Figure 17: Clavid mandatory fields

Multiple authentication types for a single ID are supported, among them password and TiQR. Figure 18 shows an example

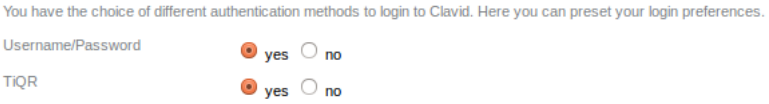

Figure 18: Clavid authentication types

Using OpenID is straightforward. An OpenID-enabled website shows the OpenID logo and a user will initiate the login. An example using LiveJournal[18] is provided.

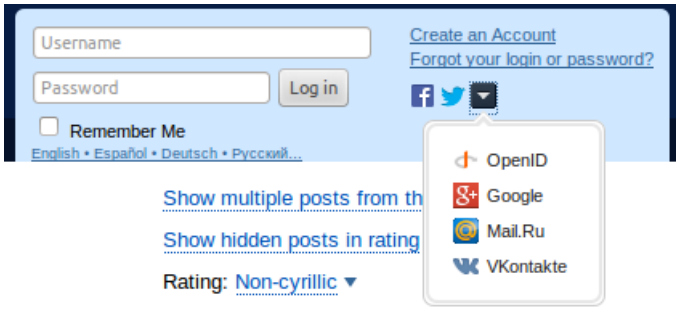

Figure 19: www.livejournal.com supports OpenID

[18] http://www.livejournal.com

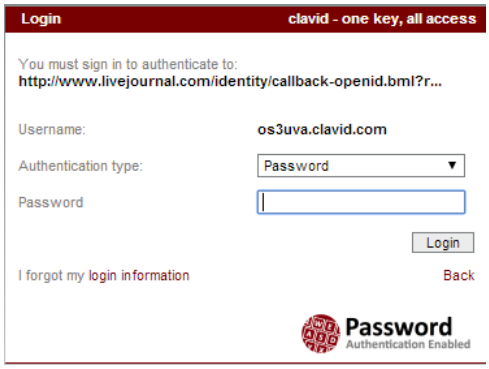

Figure 20: OpenID login using a password

Figure 21: OpenID login using TiQR

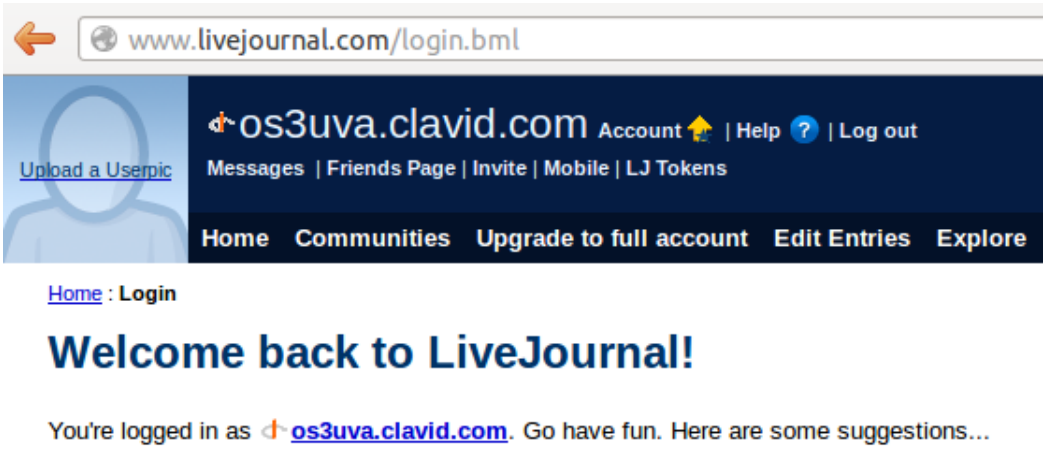

Figure 22: Successful LiveJournal login using OpenID