tiqr: a novel take on two-factor authentication

Roland M. van Rijswijk – SURFnet bv, Utrecht, The Netherlands
Joost van Dijk – SURFnet bv, Utrecht, The Netherlands

ABSTRACT

Authentication is of paramount importance for all modern networked applications. The username/password paradigm is ubiquitous. This paradigm suffices for many applications that require a relatively low level of assurance about the identity of the end user, but it quickly breaks down when a stronger assertion of the user's identity is required. Traditionally, this is where two- or multi-factor authentication comes in, providing a higher level of assurance. There is a multitude of two-factor authentication solutions available, but we feel that many solutions do not meet the needs of our community. They are invariably expensive, difficult to roll out in heterogeneous user groups (like student populations), often closed source and closed technology and have usability problems that make them hard to use. In this paper we will give an overview of the two-factor authentication landscape and address the issues of closed versus open solutions. We will introduce a novel open standards-based authentication technology that we have developed and released in open source. We will then provide a classification of two-factor authentication technologies, and we will finish with an overview of future work.

1 INTRODUCTION

1.1 AUTHENTICATION

Authentication is something we all do every day. And whether it is to log in to our e-mail account, to access our Facebook page or to tweet about that cool new album we've just bought, the use of username/password is by far the dominant paradigm.

There are – of course – applications that require a higher level of assurance such as electronic banking. The traditional approach for achieving this higher level of assurance is to use multi-factor authentication (also referred to as *strong authentication*). There is a multitude of multi-factor authentication solutions on the market. Traditionally, this market has a strong tendency towards closed solutions with strong vendor lock-in. This invariably leads to a high cost per user, hampering the wide-scale rollout of multi-factor authentication technologies. Another common limitation of current multi-factor authentication technologies is the fact that they are often single-purpose solutions (e.g. they can only be used for one bank). Furthermore, there are serious usability issues with many multi-factor solutions that make it difficult to enforce their use in most communities.

Exactly what an acceptable level of assurance is should not only be decided by a service provider; users may also have an opinion on this. Almost all solutions currently on the market give very little control to the end user.

1.2 RECENT INDUSTRY DEVELOPMENTS

In recent years there have been some promising developments in the industry. In 2004, the Initiative for Open Authentication (OATH, [1]) was formed. The intention of this initiative is to create an industry-wide reference architecture for strong authentication. The OATH initiative has been very successful in creating industry standards for two-factor authentication that have been embraced by the Internet community in the form of IETF standards ([2], [3], [4]). A number of companies and open source initiatives have adopted these standards in products and services (see also section 3).

Other developments have underlined the need to adopt open standards in the security and authentication/identity management industry. Time and again closed solutions and algorithms have been shown to be vulnerable to attack because of the lack of peer review (poignant examples include the MIFARE system and the GSM A5/1 cryptographic algorithm [5]).

Finally, large players on the Internet have recently introduced two-factor authentication for some of their services ([6], [7]). This is the first time two-factor authentication is deployed on a (potentially) large scale for applications outside the financial industry or enterprise domain.

1.3 OVERVIEW OF THIS PAPER

In this paper we aim to give an overview of the current two-factor authentication landscape in section 2. In section 3, we will further clarify some of the issues we believe exist in current two-factor authentication market offerings. Section 4 proposes a way to classify authentication solutions and con-

tains a classification of the solutions discussed in section 2. In section 5, we will introduce the innovative two-factor authentication solution we have developed which is based on open standards and open technology. Section 6 revisits the classification proposed in section 4 and adds a classification for the solution we introduced in section 5. Finally, in section 7 we will draw conclusions and provide suggestions for future work.

2 THE TWO-FACTOR LANDSCAPE

2.1 INTRODUCTION

In this section we aim to give an overview of the two-factor landscape. Before we do that, we will first give a definition of what we think constitutes two-factor authentication.

We then describe the solutions currently on offer, which we divide into two categories:

- Traditional solutions – these rely on single purpose (i.e. only used for identification) hardware devices or on a unique quality of the user (i.e. a biometric)
- Hybrid solutions – these rely on non-single purpose devices owned by the user, possibly in combination with software running on these devices

2.2 DEFINITION OF TWO-FACTOR AUTHENTICATION

In this paper, we define two-factor authentication as a means of authentication relying on the user demonstrating at least 2 separate factors from the following list:

- Something the user *knows* (e.g. a PIN code or a password)
- Something the user *has* (e.g. a hardware token)
- Something the user *is* (e.g. a biometric, such as a fingerprint)

Solutions that we place in the "hybrid" category rely on something the user has but where there is a chance of this factor being duplicated as could – for instance – be the case with a soft token running as an application on a smartphone. Some people in the blogosphere have coined the term "1.5 factor authentication" for this category (e.g. [40])

In this paper we will refer to a solution as two-factor authentication whenever the device on which the user is authenticating is physically separate from whatever constitutes the second factor (e.g. a soft token on a phone is only a second factor if it is used for authenticating a session on a separate device such as the user's computer).

2.3 TRADITIONAL SOLUTIONS

2.3.1 OTP TOKENS

One-Time Password or OTP tokens are devices that generate single-use passwords (often composed of strings of up to 10 digits). There are two variants: time-based tokens – these generate a new password at regular intervals (e.g. every 30 seconds) and event-based tokens – these generate a new password after a user intervention (e.g. pushing a button on the device).

The second factor most often combined with these devices is either a password that is entered on the user's computer or a PIN that is entered on the token device itself.

OTP tokens rely on symmetric cryptography for their operation; they contain some secret that is securely stored in the device, which can never leave it. The same secret is also known on the server that validates the user's credentials when they log in.

Examples of OTP tokens include: VASCO Digipass [10], RSA SecurID [11] and Feitian OTP Tokens [12].

Figure 1 - example of an OTP token (RSA SecurID)

2.3.2 CHALLENGE/RESPONSE TOKENS

Challenge/response tokens are similar to OTP tokens in that they also rely on symmetric cryptography for their operation. Some OTP tokens also have challenge/response capabilities.

Whereas OTP often suffices for simple authentication, challenge/response tokens are mainly used for transaction authentication such as, for instance, approving a money transfer. This is achieved by having the user enter one or more sequences of digits on the token (the challenge) and using these as input for a cryptographic algorithm to produce another sequence of digits (the response) that the user then returns to the party requesting authentication.

Challenge/response tokens are usually protected using a PIN code as the second factor.

Examples of challenge/response tokens include VASCO DigiPass [13], SafeNet SafeWord GOLD [14] and Feitian Challenge/Response tokens [12].

Figure 2 - example of a challenge/response token (SafeWord GOLD)

2.3.3 PKI TOKENS

In contrast to the previous two solutions, PKI tokens rely on public key cryptography.

Under the hood, almost all PKI tokens rely on smart card ICs with a cryptographic co-processor capable of performing public key operations and – in most cases – key generation. They come in a variety of form factors, the two most common being the smart card and the USB dongle.

Authentication with PKI tokens usually relies on some form of challenge/response algorithm. The aim of these algorithms is to prove that the user is in possession of the private key belonging to a public key that is usually stored in an X.509 certificate (for more details, see [5] sections 3.2 and 4).

Contrary to the previous two solutions, PKI tokens usually interface with the end user system. They rely on software running on that system to integrate with, for example, the browser and mail client. There is an exception to this rule: Mobile PKI (see [8]). In Mobile PKI, the user's SIM card is used as a PKI token; interfacing with the token takes place using special SMS text messages.

PKI tokens have a broader applicability than just authentication. They can also be used to create advanced – and in some jurisdictions legally binding – digital signatures (for more information, see [5] section 4.9).

2.4 HYBRID SOLUTIONS

2.4.1 SMS OTP

For many years now, the fact that almost everyone has a mobile phone is being used as a means for two-factor authentication. Many users will be familiar with SMS One-Time Passwords.

SMS OTP relies on an authentication server sending one-time passwords by SMS text message to the user. The user's mobile phone is thus leveraged as an authentication factor. The other factor is commonly username/password (thus the user first logs in using username/password and then provides additional proof of his or her identity using SMS OTP).

There is some discussion about whether SMS OTP constitutes real two-factor authentication ([15], [16]). Especially the fact that it is hard to protect the user against (temporary) stealing of their phone is a concern (putting a PIN lock almost never provides protection since SMS's are displayed even if the handset is locked).

There are many vendors of SMS OTP services; a Google search for "SMS OTP" will produce a long list.

2.4.2 OTP APPS

Another more recent development is the appearance of One-Time Password Apps. These run on modern handsets (smart phones) and usually mimic the behaviour of OTP tokens (see section 2.3.1). The difference between these Apps and 'real' OTP tokens is that the secret is stored and processed in software on the handset. This makes them somewhat more vulnerable to attacks.

Most OTP token vendors now also have App versions of their OTP tokens that interface with the same backend server systems that are also used for their hardware tokens.

3 ISSUES IN TWO-FACTOR AUTHENTICATION

3.1 INTRODUCTION

We feel that there are several issues surrounding two-factor authentication that are hampering rollout on a larger scale; most solutions are closed, they often use single-purpose tokens, are not easy to use, may have prohibitive costs associated with them and almost always lack user control. We will address these issues in more detail in the remainder of this section.

3.2 CLOSED SOLUTIONS

The most important issue with most current solutions is that they are closed ecosystems. For example, the majority of OTP tokens is based on proprietary algorithms and can only be integrated into applications by using servers or server-side components supplied by the token vendors.

Ironically, for PKI tokens it is even worse. They always require integration software on the client system in the form of cryptographic middleware (although they normally do not require server-side integration, since they are based on built-in X.509 client authentication). If the tokens are smart cards, they require smart card readers (which are not commonly installed in systems apart from some enterprise-market laptops). And both smart card readers as well as USB tokens may require specific drivers before they will work although that is less common nowadays with most of them supporting the CCID [17] standard.

Because most solutions require proprietary software, they are not easily integrated on all platforms (i.e. they will only work on vendor-supported platforms).

For OTP tokens, the advent of the OATH initiative brings hope since both the algorithms in the tokens as well as the way that the token secrets are distributed are now specified in open standards. This makes it possible to develop the server-side integration software independent of the token vendor and allows these components to support tokens from many vendors. There is already quite a bit of uptake among token vendors.

In contrast, for PKI tokens, the situation is different. Although there is an open source initiative [21], this project has not really seen a wide use or deployment and indeed most PKI token middleware is still proprietary and closed. On a positive note, PKI middleware at least adheres to the open PKCS #11 standard [22].

One final thing to mention is Mobile PKI. From an integration perspective it is fully open, because it is based on an open standard web service interface called MSSP [23], [24], [25], [26]. The downside is that a special application needs to be installed on the user's SIM card. The mobile operator owns the SIM card and access to it is strictly guarded. This means that in order to be able to deploy Mobile PKI co-operation of the mobile operator is required, which has been proven to be difficult on many occasions.

3.3 SINGLE PURPOSE TOKENS

Almost all OTP tokens are single-purpose tokens by nature because they rely on a shared secret. The tokens themselves can only contain one secret, which means that they can only be paired with one server. Unless the server is used as an authentication service for multiple applications (which is very rarely the case), the tokens can thus only be used for a single purpose (e.g. to log in to online banking for a single bank). This is very inconvenient for users, and indeed many users will know the hassle of having more than one token because they are customers at more than one bank.

In principle, PKI tokens should be more flexible because they often support storage of more than one X.509 certificate together with the associated key-pair. Unfortunately, the issuance process of PKI tokens is usually such that users have no control over the content of their token and can very rarely add credentials for additional identities. Thus, PKI tokens can only be used for multiple purposes if they contain an identity issued by a Certificate Authority that is supported by the party to which the user is authenticating.

In theory, mobile App-based solutions can more easily support multi-purpose deployments, in practice this does not happen very often yet.

3.4 (LACK OF) EASE OF USE

Many users will have experienced how difficult it can be to use OTP tokens. Most of them require typing in complicated codes. The challenge/response variety is even more complicated where users regularly have to type multiple codes on the token and then they have to copy the result from the token by typing it on the site they are authenticating to.

SMS OTP is no better. In fact, it is even more complicated in our opinion as the one-time passwords used often consist of both capitals and lower case letters as well as digits and punctuation marks.

PKI tokens fare a little better. As long as the software integration with the user's browser is properly installed, the user experience is usually quite smooth.

A common issue shared by all tokens except mobile phone-based ones is that they are all too easy to forget or lose.

3.5 COST

Both OTP and PKI tokens can be quite costly, both in initial investment as well as yearly licence fees. It is not uncommon to pay tens of US dollars per user per year. SMS OTP becomes gradually more costly the more it is used.

The only exception to this rule is a new class of OTP tokens that are emerging, based on open standards developed by the OATH initiative. Because they work with open source software, the only substantial cost is the initial investment. Yubikey tokens [27], for example, can be purchased for less than USD $30 and the price goes down for larger volume purchases.

3.6 (LACK OF) USER CONTROL

Users seldom initiate deployment of two-factor authentication solutions. They are usually deployed by corporate IT departments or banks. The organisations deploying these tokens strictly control what they can or cannot be used for, severely limiting users.

It is very hard for users to acquire personal two-factor tokens and deploy them in a useful way because very few services provide the means to self-enrol identities. A notable exception to this is Google Authenticator [28].

4 CLASSIFICATION OF AUTHENTICATION SOLUTIONS

4.1 INTRODUCTION

In this section we will introduce six different ways to classify authentication solutions in order to judge their suitability. We will use this classification at the end of this section to classify the two-factor authentication solutions discussed earlier.

4.2 HARDWARE INDEPENDENCE

The first way to classify authentication solutions is by their dependence (or lack thereof) on specific or specialised hardware for their operation.

We feel that hardware independence enhances the usability of a solution, because the more independent a solution is from specific hardware, the fewer devices a user has to carry around.

From a security perspective, however, using special purpose-made hardware has distinct advantages. Devices can be tailored for one goal, which is to protect the secrets associated with a user's credentials.

In this paper, we will focus mainly on the enhanced usability that comes with hardware independence; we will factor in the security advantages that special hardware can offer when we judge the security of a solution. We rank solutions that offer stronger hardware independence more favourably than solutions that require specific hardware to operate.

4.3 SOFTWARE INDEPENDENCE

Just like hardware independence, software independence is mainly a usability enhancing aspect. In some cases, dependence on specific hardware goes hand in hand with dependence on specific software. For example, smart cards cannot operate without the accompanying security middleware that users will have to install on their computer.

Some solutions only depend on specific software on the server side and do not require the user to install software (for example OTP tokens).

We will judge solutions on the amount of effort required to install software by both end users as well as by the system administrators of the server side. We will also factor in the availability of integration in off-the-shelf products as this can significantly reduce the effort required to install the required software.

4.4 SECURITY

Security is – of course – one of the most important factors when judging authentication solutions.

There are several aspects that influence the security of a solution:

- Is the solution a multi-factor solution? If so, is it a true multi-factor solution (see §2.3) or a hybrid solution (see §2.4)?
- Does the solution rely on purpose-built hardware that has provisions for e.g. tamper resistance?
- Are there well-known attacks that (severely) impact the security?
- If the solution relies on cryptography, does it rely on sufficiently strong as well as open cryptography?
- Has the security of the solution been verified by reputable independent security auditors?

4.5 COST

Cost is an important factor, especially for large-scale deployments. It can be considered from a number of different angles:

- The one-time setup cost (e.g. in software and hardware purchases) and recurring cost of the actual solution (e.g. yearly licence fees).
- The cost for troubleshooting for users who have misplaced their credentials or forgotten their password or PIN.
- The cost of integrating the solution into existing IT infrastructure (what skill level is required and how much time do system administrators or system integrators spend setting up the solution).

4.6 OPEN STANDARDS COMPLIANCE

Open standards form the backbone of the Internet. Vendors implement these standards that are available free-of-charge or for a reasonable fee to guarantee interoperability with systems from other vendors.

There is a whole host of open standards in the authentication arena that make it easier to integrate solutions into existing IT infrastructure. They also offer a certain level of vendor independence as one solution can be more easily exchanged for another. Of course, this also depends on the level to which open standards have been integrated. For example: OTP tokens that fully support the open standards of the Open Authentication Initiative can easily be integrated with server-side software from a range of vendors that support these standards. On the other hand, PKI tokens that rely on PKCS #11 middleware are less easily replaced by another solution as they will require specific middleware supplied by the token vendor.

For a long time supporting open standards was not common practice, especially among OTP token vendors. Fortunately, this is now changing for the better with the advent of consortia like the Open Authentication Initiative.

4.7 EASE-OF-USE

A final factor that can go a long way in determining the success of a solution is ease-of-use. At first glance, solutions that are already familiar to a user – such as username/password – may seem easy-to-use. But when all the kludges that have been added to enhance the security such as complex password policies and requirements to change passwords on a regular basis are considered, it is easy to see that such solutions may not be as easy-to-use as initially assumed.

Other things that need to be factored in when considering the ease-of-use of a solution are:

- Does the solution require the user to carry around additional devices (that he/she otherwise would not need to operate their computer)?
- Does the user have to re-type complicated codes (such as may be the case for OTP tokens)?
- Has care been taken to design the user experience such that the solution can be used intuitively by the user rather than requiring them to learn how to operate the solution from e.g. a manual?

4.8 CLASSIFICATION

Table 1 below shows the scores we have assigned to each solution described in section 2 for each of the 6 different classification categories described earlier; we used a five point scoring system ranging from ++ (indicating that a solution is (one of) the best in class for the given classification category) to -- (indicating that a solution has very unfavourable characteristics compared to other solutions in the given classification category). Any scoring system is, of course, subjective; we endeavour to justify the scores in Table 1 in section 4.9.

	Hardware indep.	Software indep.	Security	Cost	Open Standards	Ease-of-use
Usern./pwd	++	++	--	++	=	+/-
OTP token	-	-	++	--	-/=	+
C/R token	-	-	++	--	-/=	+
PKI token	--	--	++	--	=	+
Mobile PKI	+	+	++	?	+	++
SMS OTP	+	=	-	-	--	-
OTP Apps	+	+/=	+	+/=	+/=	=

Table 1 - Classification of authentication solutions

4.9 JUSTIFICATION

We would like to highlight certain points of the classification we made in the previous section. Given the endless stream of news articles about username/password getting compromised we feel that – even though it is tried and tested – this paradigm is really lacking in security. And even if organisations enforce secure password policies and users adhere to them, they may still be at risk. Recent developments in password cracking such as using GPU-based cracking systems make the security of any password under a certain length questionable [45]. With the increasing value that online identities have (how would you feel if your GMail, your FaceBook or your Twitter account got compromised and someone reads your private data or tries to impersonate you?) we, as authors, are of the opinion that two-factor authentication should become much more common than it is now.

As the classification shows, to get rock solid security using two-factor authentication we feel that a real purpose-built hardware token should be used. Nevertheless, emerging solutions that rely on mobile phones as personal devices, such as OTP Apps, show great promise. If implemented properly, these solutions can add significant value security-wise.

There are three key problems currently inhibiting wide-scale deployment of two-factor authentication outside of the corporate and banking environment. The first is cost; OTP and PKI tokens are expensive (there are exceptions: interestingly, one of the largest deployments of OTP tokens is for online World-of-Warcraft [41]). The second is dependence on bespoke hard- and software. Especially PKI tokens suffer from the problem that they require the end-user to install driver software and security middleware that is not always available for all end-user platforms.

Finally, the last problem is the lack of adherence to open standards. This not only stops people integrating support for two-factor authentication into their online services, it also means that many two-factor products are single purpose only (e.g. a token issued by a bank cannot be used to authenticate for other services).

We have tried to let these three problems be reflected in the classification given in Table 1.

5 tiqr: EXAMPLE OF AN OPEN APPROACH

5.1 INTRODUCTION

In 2009 we experimented with Mobile PKI (see also §2.3.3) as a means of authentication. As the report [8] of our experiment shows, we were very happy with the results. The technology is user-friendly, very secure and – because of the open standards it is based on – easy to integrate.

The only major hurdle we encountered is the dependence on mobile operators. These operators are very hesitant about deploying the technology because it requires a SIM swap (most SIMs deployed in The Netherlands are not PKI capable), and because they do not feel that there is a strong business case to deploy the technology in terms of potential revenue from it.

As operator of the National Research and Education Network (NREN) in The Netherlands, SURFnet operates a so-called identity federation (see [29]) called *SURFfederatie*. This federation enables users to log in at a multitude of online service providers using a single identity hosted by their home institution. Furthermore, this federation offers users single sign-on.

As is the case on most of the Internet, almost all authentications in the *SURFfederatie* rely on the tried and tested username/password mechanism. We would like to improve on this situation by introducing alternative means of authentication based on two-factor authentication technology. There are two reasons for this: first, we feel that some services require a stronger form of authentication than username/password. Secondly, we would like to offer users a safe alternative that they can use on untrusted systems such as, for instance, computers in Internet cafés.

SURFfederatie has a sizable and very heterogeneous user population consisting of approximately one million students, researchers and other staff from over a 160 different institutions. It would be impossible to deploy a token-based two-factor authentication solution because of the logistics involved. It would, however, be ideal if we could deploy a secure two-factor authentication system that uses mobile phones. Almost everyone owns a mobile phone (in fact, in The Netherlands, a country of 16.5 million people, there are over 19 million active mobile subscriptions [30]) and users are very motivated to carry their mobile phone at all times [31].

For reasons mentioned before, we could not rely on Mobile PKI so we started searching for an alternative. The criteria for this alternative were that it should be secure, user-friendly, easy to deploy, open and suitable for managing multiple identities. We believe that we have developed a novel solution that meets all of these criteria.

5.2 THE CONCEPT

5.2.1 BASIC FEATURES USED

The concept we call tiqr is based on three features of modern smartphones:

- The ability to run Apps
- A camera
- Internet connectivity

5.2.2 QR CODES

Relying on these smartphone features allows tiqr to make use of two-dimensional barcodes called QR codes. They were invented by Toyota subsidiary Denso-Wave in the 1990s.

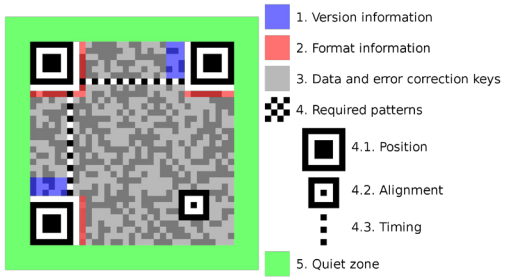

Figure 3 - a QR code with specific features highlighted (source [9])

Although patented, QR codes can be used royalty free. The technology behind the codes has been standardised as ISO/IEC 18004:2006. Up to 4KB of alphanumeric data can be stored in the codes and numerous libraries are available that can extract information contained in a QR code from images captured by a camera. For more details about QR codes, we refer readers to the excellent Wikipedia article [9].

QR codes have become quite popular, because most phones are equipped with cameras and can run QR code reader software. The codes are almost exclusively used in a static fashion, for instance in advertising or on public transport stops. They usually contain an encoded URL that QR code readers can open in a mobile browser.

The innovation we have come up with is to use QR codes in a dynamic rather than a static fashion. By encoding a challenge in a dynamically generated QR code that is displayed to the user when he/she wants to log in, we use QR codes to take away the burden on users of typing challenge/response codes. QR codes are also used during enrolment to tie the user's phone to an identity. Although this solution is not unique – the Google Authenticator App [28] can use a QR code to convey the user secret during enrolment – we have taken this technology further by creating a seamless user experience.

5.2.3 THE TIQR USER EXPERIENCE

To illustrate how tiqr works, we will go through the tiqr user experience during authentication (assume for now that a user already has a tiqr-enabled account).

The flow starts by a user surfing to a website that requires them to log in. Where most sites would display a username/password dialog (or an entry field to enter a one-time password), with tiqr users will see a QR tag as shown in Figure 4.

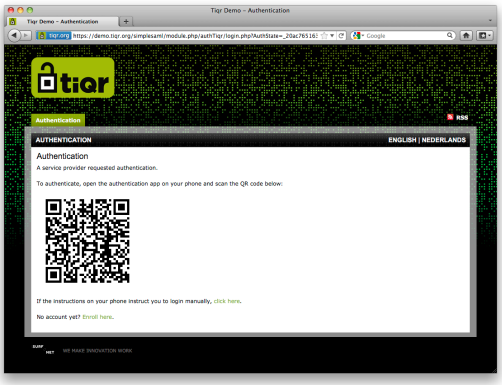

Figure 4 - tiqr login page showing a QR code

Contained in the QR code is a challenge. The user now launches the tiqr App on their smartphone. The App will activate the camera allowing the user to scan the QR code.

Figure 5 - the user scans the QR code with the tiqr App

Apart from a random challenge, the QR code also contains information on the relying party requesting authentication. The App can manage multiple identities and will select an appropriate identity that can be used to log in to this particular site (if multiple identities are present, the user will see a list and can choose the appropriate one). The tiqr App now asks the user to confirm that he/she wants to log in, also displaying the domain name of the site they are logging in to in order to reduce the risk of phishing.

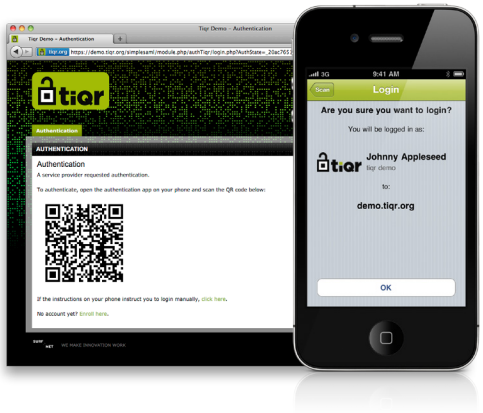

Figure 6 - tiqr asks for user confirmation

Once the user has confirmed their identity, they will be asked to enter their PIN code (the second factor).

Figure 7 - user entering their PIN

The user is helped in remembering his or her PIN by means of animal icons displayed in the PIN entry dialog. Errors made during PIN entry (such as swapping two digits or a completely different PIN) will lead to a different sequence being displayed. When the user presses OK, login will proceed. If the user's phone is online, the Internet connection of the phone will be used to submit the response to the authenticating server thus obviating the need to type one-time passwords in to a website. When authentication is successful, the user is notified both on the phone as well as by the website proceeding with login by redirecting the user to the protected content as shown in the screenshot (Figure 8).

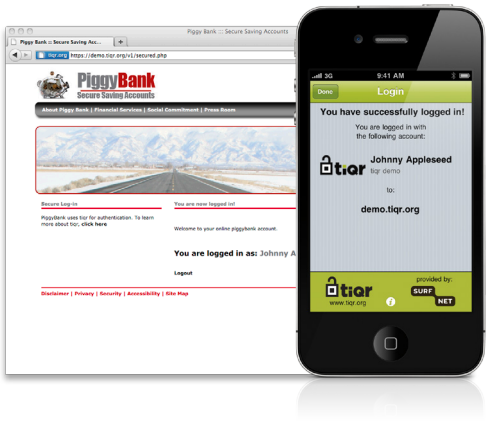

Figure 8 - the user has successfully logged in

In case no Internet connection is available on the phone a fall-back scenario is used where a one-time password is displayed on the phone for the user to type into the website (more on this in section 5.8).

5.2.4 FROM PROOF-OF-CONCEPT TO PRODUCT

We first came up with the concept that led to the development of tiqr in September 2010. In order to prove that the concept would work, we designed the initial protocol and developed a proof-of-concept implementation, both of the server side as well as of the phone side. For the proof-of-concept an implementation was created for Apple's iOS platform.

The proof-of-concept quickly showed that the technology worked very well. We first demonstrated the working proof-of-concept at an event held every two years to showcase SURFnet innovations to our connected institutions in December 2010 and received helpful and positive feedback from the people attending. This led us to decide that we should continue development.

In April 2011 we released the first Apple iOS production version in the Apple App Store and we presented on the project at the Internet2 Spring Member Meeting in Arlington, VA. The Android version was released in May 2011 just before we presented on further improvements to tiqr at the TERENA Networking Conference 2011 in Prague, Czech Republic.

The remainder of this section will go into more detail about the tiqr technology.

5.3 MOBILE APPS

5.3.1 PLATFORMS

We wanted to make tiqr available on the two most common smart phone platforms. According to a Q1 2011 market survey, those platforms are Apple's iOS and Google's Android platform:

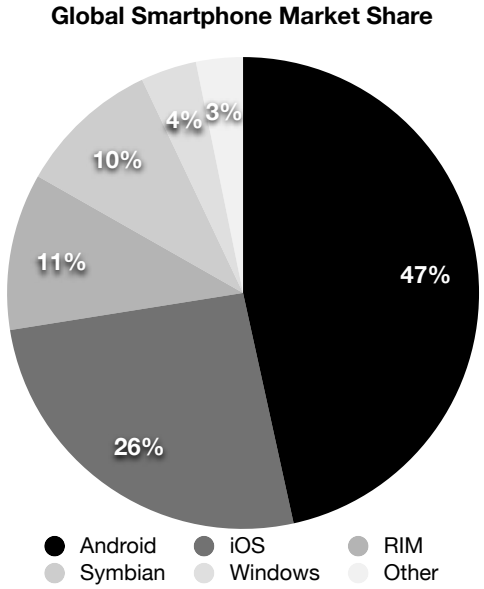

Figure 9 - smart phone market, source: The Guardian/Kantar

We have developed Apps for both these platforms. The Apps rely on the excellent ZXing QR code library developed by Google (see [18]) for QR code detection and decoding. The Apps implement the tiqr challenge/response protocol, which is based on OCRA/HOTP [19], [2]; more information on the protocol can be found in section 5.5.

5.3.2 APP SECURITY CONSIDERATIONS

The tiqr protocol relies on shared secrets for the challenge/response implementation. The secret is stored both on the phone as well as on the server.

We can only reasonably assume that the phone with the App and the secret on it is a secure authentication factor if it is hard for an attacker to gain access to the actual secret. We therefore protect the secrets belonging to user identities by encrypting the secrets using PKCS #5 password-based encryption [20]. The basis for encryption is the 4-digit PIN code the user chooses for the identity.

Of course there are only 10000 possible PIN codes with a 4-digit PIN. We assume that it is easy for a motivated attacker to gain access to the encrypted secret so we need to protect it against brute-force attacks. We achieve this by applying two principles. Firstly, the encrypted secret contains no internal structure (i.e. only the secret key – which is assumed to be truly random – is encrypted, there is no formatting around the key data before it is encrypted). This automatically leads to a second level of protection: because the encrypted key has no structure around it, it is impossible to check if the

correct PIN was used to decrypt the secret since the decrypted data will look like random data in all cases. As a result of this, only the server can check if the correct PIN was entered because the computed response is only valid if the correct secret key was used.

To prevent online attacks, we recommend that the server block an account after a pre-set number of failed authentication attempts (in fact our demo implementation blocks an account after 3 failed attempts). Depending on the desired security level, the server administrator may also decide to implement some form of exponential back off mechanism to mitigate brute-force attacks. In this scenario, accounts are temporarily blocked after a failed login attempt. This thwarts brute-force attacks but is also more user friendly for legitimate users since entering the wrong PIN more than a certain number of times will not immediately lead to a blocked account.

5.3.3 APP USER EXPERIENCE

One of our main goals was to create an easy-to-use system. We have taken special care to ensure that the user experience of the App is as straightforward, self-explanatory and smooth as possible. The prototype developed for the proof-of-concept was handed over to user-interface designers. They studied the concept and the prototype implementation. Using storyboards, they designed an optimised user workflow. The main focus of the workflow is to make it self-evident to the user what the next logical step is going to be. Another change they introduced was to do away with a separate enrolment workflow (in the prototype, we had two completely separate workflows for enrolment and authentication). In stead, the user just scans the QR code that is shown and information in the code determines whether an authentication or an enrolment workflow is going to be followed.

Another design decision that was made was to try to steer users toward using the same PIN code for all the identities managed by the tiqr App. The reasoning behind this is that the user-interface designers feel that it is counter-intuitive for most users to have multiple PIN codes in a single application. This concept is on the one hand very subtly integrated in the user experience by using suggestive wording (i.e. when enroling a new identity using the text "Please enter *your* PIN" when they have to choose a new PIN for the identity rather than "Please choose a new PIN"). On the other hand, it has also been taken quite far in that if a single identity becomes blocked due to entering the wrong PIN too many times, the App will block all identities it manages. We have not had sufficient user feedback to be able to decide whether or not this was a good choice; so far, we have had some feedback from third party developers that they feel this to be a bad choice. We hope to learn more in a pilot implementation that we are planning for the fall of 2011.

One final thing to note about the user experience is that we integrated an *aide-memoire* into the PIN entry dialog. We use icons with animal shapes to help the user remember their PIN (as shown in the figure below).

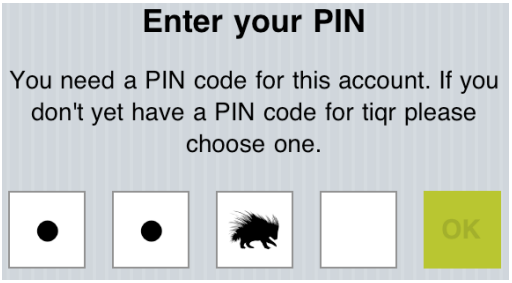

Figure 10 - PIN entry showing animal reminders

If the user enters the correct PIN, the same four animal icons should show up in the PIN entry field. Users can either remember the whole sequence or elect to remember just the last icon. To ensure that the sequence changes when common PIN entry mistakes are made (such as swapping two digits) we use the Verhoeff checksum algorithm [32] for error detection.

5.4 SERVER SIDE

5.4.1 REQUIREMENTS

As was already mentioned in the previous section, the basis for tiqr is challenge/response authentication using shared secrets (more information on the protocol can be found in section 5.5). This means that the secret key information that is present on the phone also needs to be stored on the server.

This, of course, puts certain requirements on the server implementation. User secrets should be stored encrypted, either on disk in a database or in a Hardware Security Module (HSM).

Another thing that is required on the server side is a library that generates the QR codes used to convey the challenge to the user. There are good open source implementations available for most common web application platforms. For our reference implementation we use PHP QR Code [33].

The most important thing to pay attention to on the server side is that the protocol is implemented correctly. We provide a reference implementation to show how the protocol works (which is discussed in the next section).

5.4.2 REFERENCE IMPLEMENTATION

To give developers a head start at integrating tiqr into their application, we have developed a reference implementation in PHP. This reference im-

plementation shows how the tiqr protocol works (see section 5.5).

We did not put any security provisions in the reference implementation so it should not be used in production. We are considering creating a more secure implementation that people can deploy straightaway.

5.4.3 SIMPLESAMLPHP MODULE

As outlined in section 5.1, we plan to use tiqr in an identity federation. To show this concept in action, we have developed a plug-in module for the popular SimpleSAMLphp [34] identity management suite.

Our demo portal (https://tiqr.org/demo/) uses this implementation.

It is currently based on our reference implementation so it is not sufficiently secure yet for production use. We are collaborating with the SimpleSAMLphp team to create a production-ready version, which we hope to release in the autumn of 2011.

5.5 PROTOCOL

5.5.1 GENERAL

We rely solely on open standards and open specifications as a basis for the tiqr protocol. The following standards are used:

- OCRA [19] – this is the suite of one-time password algorithms used for tiqr challenge/response
- JSON [35] – this is the object notation used to exchange data in the tiqr protocol
- HTTP over TLS – used to transport information exchanges securely
- QR codes [9]

5.5.2 ENROLMENT

Enrolment starts with a QR code that is displayed to the user. This QR code contains a URL with the following schema:

```
tiqrenroll://<url>
```

Where *<url>* must be a valid HTTPS URL that points to a location where the details for the enrolment request can be retrieved, for example:

```
tiqrenroll://https://demo.tiqr.org/enroll
/details?session=082176122169132630
```

The tiqr App will contact this URL to retrieve a JSON object with enrolment details. This object has the following syntax:

```
{
    "service": {
        "identifier": <id>,
        "displayName": <name>,
        "logoUrl": <logo-url>,
        "infoUrl": <info-url>,
        "authenticationUrl": <auth-url>,
        "ocraSuite": <OCRA-suite>,
        "enrollmentUrl": <enroll-url>,
    },
    "identity": {
        "identifier": <uid>,
        "displayName": <fullName>
    }
}
```

The *service* section of the object identifies the service to which the user is enrolling. The *identity* section provides details about the identity that is being enrolled. The fields in both sections of this object have the following semantics:

- Service section
  - *identifier* – should contain a reversed domain name (e.g. org.tiqr.demo)
  - *displayName* – should contain the name of the service
  - *logoUrl* – should contain a valid URL to a service logo; we recommend a PNG24 image
  - *infoUrl* – a URL linking to a webpage with more information about the identity provider; this link is displayed on the "detailed information page" for the identity
  - *authenticationUrl* – should contain the URL for the authentication handler for this service
  - *ocraSuite* – the OCRA suite the server requires; the App uses this to determine the appropriate OCRA parameters (see [19], section 6)*
  - *enrollmentUrl* – should contain the URL for the one-time enrolment handler
- Identity section
  - *identifier* – should contain a unique user identifier used to identify the account
  - *displayName* – should contain the full name of the user

*An example OCRA suite as specified by the server could for instance be:

```
OCRA-1:HOTP-SHA1-6:QH10-S
```

This OCRA suite specification breaks down as follows:

- *OCRA-1* – the OCRA algorithm version (in this case version 1, the current version)
- *HOTP-SHA1-6* – the cryptographic function to use (in this case HMAC OTP [2], with SHA-1 as hash algorithm using dynamic truncation to a 6-digit value); the tiqr App supports all algorithms specified in the OCRA standard
- *QH10-S* – the input for the challenge (in this case a 10-digit hexadecimal value represented as a string) and the size of the session data (in this case the default value of 64 bytes)

For more examples see [19].

When the user confirms enrolment, a new HTTPS connection is made to the enrolment server URL specified in the JSON object *enrollmentUrl* property. A POST request is sent across this link. This POST contains the following parameters:

- *secret* – this is the shared secret; the secret is generated by the App on the phone; we currently use 256-bit AES keys as secrets
- *notificationType* – optional; this is the notification type used to send push messages to the App and can be set to either APNS (for Apple Push Notification Service) or C2DM (for Android push notifications)
- *notificationAddress* – optional; notification-protocol specific address to which push notifications can be sent
- *language* – contains the user interface language of the user; this information may be used to display appropriate error messages in the user's preferred language

If enrolment is successful, the server will return the string *OK* (with no white space before or after the string). When an error occurs, the normal HTTP error procedure is followed to return the error to the App.

5.5.3 AUTHENTICATION

Authentication starts by displaying a QR code to the user. This QR code contains a URL encoded according to the following URL schema:

```
tiqrauth://[<identityIdentifier>@]
     <serviceIdentifier>/
     <sessionKey>/
     <challenge>[?<return Url>]
```

The fields in this URL have the following semantics:

- *identityIdentifier* – optional field specifying the user identity to use for authentication; may be used in a so-called step-up authentication scenario where the user has already logged in using another means of authentication
- *serviceIdentifier* – the service identifier as specified during enrolment (the service domain name in reverse domain notation, e.g. org.tiqr.demo)
- *sessionKey* – session key for this authentication request; links the response to the active user session when submitted
- *challenge* – the authentication challenge; the size of the challenge depends on the OCRA suite as specified during enrolment
- *returnUrl* – optional field specifying the URL to return the user to after successful authentication; this URL is only used if the session originated from the mobile browser on the device containing the tiqr App

The tiqr App will compute the response to the challenge using the algorithm that was specified during enrolment. It will submit the response by setting up a HTTPS connection to the authentication endpoint specified during enrolment. The submission is done using a POST with the following parameters:

- *sessionKey* – the session key received in the QR code identifying the user session that requires authentication
- *userId* – the user identifier of the user attempting to log in
- *response* – the response computed to the challenge specified in the QR code
- *language* – the user's preferred language; this information is used to display error messages in an appropriate language

If authentication was successful, the POST request returns the string *OK* (with no white space preceding or following the string). If authentication fails, the server will return one of the following error messages:

- *INVALID_RESPONSE[:attemptsLeft]* – the response provided to the challenge was invalid; this is interpreted by both the App as well as the server as an incorrect PIN entry. The optional integer value *attemptsLeft* indicates the number of tries left to return a correct response (and enter the correct PIN)
- *INVALID_USERID* – the server does not know the specified user
- *INVALID_CHALLENGE* – there is no known challenge for the current session; this usually indicates that the challenge has become invalid because of a timeout
- *ACCOUNT_BLOCKED[:seconds]* – indicates that the response provided to the challenge was invalid and that the associated user account is now blocked on the server; optionally, the account may be temporarily blocked for a specified number of seconds (this feature will be implemented as of version 1.2 of the tiqr App)
- *INVALID_REQUEST* – the POST request contained incorrect parameter data and was not accepted by the server
- *ERROR* – an unspecified error occurred

5.6 INTEGRATION WITH APPLICATIONS

As we already mentioned in section 5.4.2 and 5.4.3, we already provide several options for integrating tiqr into existing applications. The reference implementations we provide can serve as a basis for integration into web applications, but tiqr can also be used in other contexts.

Shortly after the first release of tiqr an independent software vendor, RCDevs from France, integrated support for tiqr into their OpenOTP Authentication Server [36]. Based on their existing integration with several products they were able to show that tiqr can – for instance – be used as an

authentication method to log users in to a secure shell (SSH) session (see Figure 11 for an example).

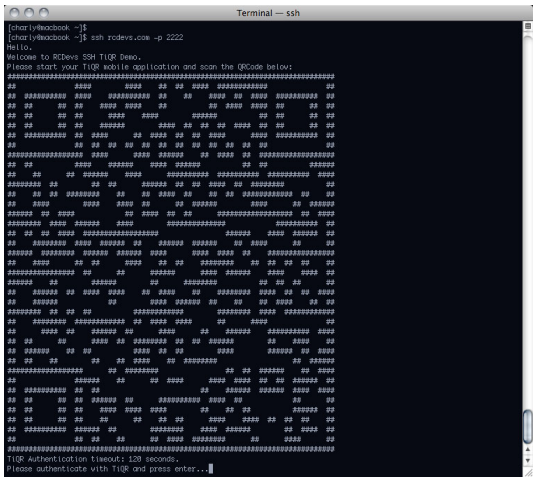

Figure 11 - using tiqr to authenticate an SSH session*

*Screenshot courtesy of Charly Rohart from RCDevs

Another example of integration into third party frameworks is the open request to integrate tiqr support into Shibboleth [37], a much-used framework for federated identity management.

5.7 SECURITY AUDIT

We hired an independent security auditor – Eindhoven-based Madison Gurkha, see http://www.madisongurkha.nl/ – to assess the security of tiqr. The goals we set them were to:

- Assess the architecture and design of tiqr from a security perspective
- Perform a code audit of both the App for iOS as well as for Android
- Perform a code audit of the reference server-side implementation
- Perform security tests on the live solution (both server as well as client side)

The security audit was performed on version 1.0 of the App (as released in April 2011 for iOS and May 2011 for Android) and was finished in June. The outcome of the security audit was positive; although the auditors identified several issues that needed resolving, they did not find any flaws in the architecture of the solution (note: the audit report will be published on the tiqr website, https://tiqr.org/, in the autumn of 2011). The most important remark from the auditors was that – strictly speaking – tiqr does not offer full two-factor authentication since the smart phone platform is much more accessible to evildoers than say a purpose-built hardware OTP token. We agree with this but would like to add that tiqr nevertheless is a vast improvement security-wise over username/password. And relying on a smart phone also has distinct advantages; recent research has shown that users are likely to notice that their phone is missing fairly quickly (see [44]). We think that this is much less likely to be the case for e.g. OTP tokens, since the single-purpose nature of these devices means they are used much less frequently.

We have taken care to include several security measures to deal with the inherent untrustworthy nature of smart phone platforms (see 5.3.2). The auditors agree that these measures indeed significantly enhance the security and they also agree that tiqr is an attractive and more secure alternative to username/password. They caution though that it is not fully equivalent to the security a hardware OTP token can offer. We would like to note that this is also true for the more traditional OTP Apps that OTP token vendors have started offering (see 2.4.2).

Another remark that the auditors made is that tiqr is potentially vulnerable to phishing. Attackers could perform a man-in-the-middle attack by initiating an authentication session, thus retrieving the QR code containing the challenge and by displaying this on a fake site, tricking the user in to logging in but instead giving the attacker access to their account. We agree that this is a risk and as mitigation the tiqr App always displays the fully qualified domain name of the site that the user is being authenticated to. Users are expected to validate the authenticity of the site they are logging into in the same way they do for e.g. their banking site, i.e. by checking the site's URL and server certificate. Note that this problem is not unique to tiqr; all OTP solutions are equally vulnerable to phishing.

The issues in the code and design identified by the auditors were resolved in version 1.1 of the App and in the reference server-side implementation that is available from the tiqr website. We also contributed fixes for the vulnerabilities in the OCRA reference implementation back to the authors of the RFC.

5.8 AVAILABILITY

From the onset it has been our goal to make tiqr freely available to all Internet users. To achieve this goal, we have released all relevant software in open source under a BSD-style licence and we have made the tiqr Apps available for free in both the App Store as well as on the Android Market.

All source code and documentation as well as a demo server can be found on our website, https://tiqr.org/

5.9 ROADMAP AND FUTURE WORK

Now that a production-ready version is available our next step will be to deploy tiqr within our own organisation. SURFnet currently uses X.509 software certificates for authentication to certain services. We plan to replace these by tiqr. All SURFnet employees have either an iOS- or Android-

based smart phone making it an ideal user population in which to use tiqr.

When we have gained real-world experience we will evaluate this deployment. Then, we plan to gradually introduce tiqr as an alternative means of authentication (alternative to username/password and/or SMS authentication) in some of the services we offer to our constituency.

We are also talking to our connected institutions to set up a pilot with a larger population. Our goal is to provide tiqr as an alternative means of authentication on an identity provider in our federation who currently only offer username/password.

One of the things we will be evaluating in these pilot deployments is whether or not the paradigm of encouraging users to use the same PIN for all their tiqr accounts works and whether or not it makes sense to block all accounts if one account needs to be blocked because of too many failed attempts at entering the correct PIN.

From a technological perspective, we are considering pursuing several areas of research:

- Turning tiqr into a true hardware token by leveraging the possibilities offered by SD cards with an embedded smart card controller (smartSD cards, see [39])
- Incorporating attribute release into tiqr (where tiqr releases attributes about a user asserted by a trusted third party), similar to the InfoCards paradigm (see [38])
- Using advances in cryptography such as zero-knowledge proof to further enhance the privacy aspects of tiqr.
- Using tiqr for transaction signing (as is e.g. done by banks with OTP tokens to approve financial transactions).

Some of these we will probably do ourselves, others we hope to pursue together with the academic community in the areas of cryptography and digital security.

5.10 REFLECTION

When we started the tiqr project we set out to create a two-factor authentication solution that would leverage the benefits of using a device that (almost) everybody has: a mobile phone. We were also mindful that the solution should be an improvement over username/password given the criteria introduced in section 4 and if possible it should also offer advantages over more traditional solutions.

We feel that with tiqr we have achieved most of these goals. We believe tiqr to be very user-friendly (much more so than some other two factor authentication solutions) and also believe that tiqr is an improvement in terms of security over username/password and on a par in that respect with many other two-factor authentication solutions. Furthermore, we believe that tiqr can be a viable replacement for more traditional OTP solutions, especially the OTP Apps and SMS authentication.

We feel that we should point out, though, that tiqr is not a panacea that solves all problems in (two-factor) authentication. It is, for instance, just as vulnerable to phishing as traditional OTP solutions. And because it does not rely on purpose-built hardware to store the secret data associated with a user's identity its security is not as strong as traditional tokens.

Nevertheless, we feel that tiqr is a useful addition to the two-factor authentication landscape. Its user-friendliness and the control over deployment it gives to organisations are also strong points.

6 CLASSIFICATION REVISITED

6.1 INTRODUCTION

In section 4 we introduced a classification for authentication solutions, judging solutions on hardware (in)-dependence, software (in)-dependence, security, cost, open standards compliance and ease-of-use. Now that we have introduced tiqr, we will revisit this classification.

6.2 CLASSIFICATION OF TIQR

Table 1 showed a classification of authentication solutions according to the criteria introduced in section 4. In Table 2 we have reprinted this classification and added tiqr at the bottom of the table (marked in grey).

	Hardware indep.	Software indep.	Security	Cost	Open Standards	Ease-of-use
Usern./pwd	++	++	--	++	=	+/-
OTP token	-	-	++	--	-/=	+
C/R token	-	-	++	--	-/=	+
PKI token	--	--	++	--	=	+
Mobile PKI	+	+	++	?	+	++
SMS OTP	+	=	-	-	--	-
OTP Apps	+	+/=	+	+/=	+/=	=
tiqr	+/=	+/=	+	+	++	++

Table 2 - Classification including tiqr

Again, one could argue that any classification is subjective, especially since we are judging our own solution. Therefore, we have tried to justify the classification we have assigned to tiqr below:

- *Hardware (in-)dependence* – tiqr requires advanced features only available on smart phones; it is therefore not as hardware independent as some of the other solutions that rely on mobile phones
- *Software (in-)dependence* – tiqr is currently only available for two smart phone platforms

- *Security* – although not as secure as a dedicated token, tiqr is much more secure than username/password and on a par with other OTP Apps
- *Cost* – tiqr is open source and available for free; the only inhibiting factor may be the cost of the device required to run tiqr
- *Open standards* – tiqr was built from the ground up to include open standards and the tiqr protocol itself has also been published
- *Ease-of-use* – tiqr was designed to be user friendly from the ground up by skilled interface designers

7 CONCLUSIONS AND RECOMMENDATIONS

7.1 THE NEED FOR TWO-FACTOR AUTHENTICATION

Our lives are increasingly being lived in the digital world. Social networks have become the staple of a new generation and many professionals cannot live without e-mail, VoIP, and services like LinkedIn. Governments and the public sector are also increasingly making vast amounts of often personal data (like medical records) available online.

This means that the value of the digital identities we use to access these services are becoming ever more valuable. It is no longer just your credit card that is at risk of being stolen, whole identities get hijacked.

We feel that it is inevitable that two-factor authentication becomes more widespread and actively try to stimulate its adoption, both within our own community as well as on a wider scale.

7.2 OPEN STANDARDS AND OPEN SOURCE

The best way forward to bring two-factor authentication to a wider audience is the adoption of open standards by vendors. This applies foremost to vendors of hardware OTP and PKI tokens. It is also of paramount importance that integration of two-factor authentication in online services becomes easier. One way of achieving this is by releasing open source solutions with flexible licenses. These can serve as useful examples and facilitate rapid integration.

7.3 TIQR

We have strived to practice what we preach when creating tiqr. We have focused on creating an open standards-based, open source and easy-to-use solution that is freely available. We believe that we have succeeded in the goals we set ourselves in that respect and we hope that tiqr can serve as a starting point for many organisations who want to integrate two-factor authentication into their online services.

What is also noteworthy to mention is that parts of the tiqr project have now been spun off as separate open source projects because of their general applicability. These include TokenExchange (an abstraction for push notifications supporting several device platforms), see [42], and a set of OCRA reference implementations in several programming languages, see [43].

7.4 RECOMMENDATIONS

We have already outlined recommendations for future work on tiqr in section 5.9. In addition to that, we would recommend any readers of this paper to invest some time into considering what two-factor authentication could add in terms of security both within their own organisations as well as for users of their online services.

8 ACKNOWLEDGEMENTS

The authors of this paper would like to thank:

- Ivo Jansch, Peter Verhage, Felix de Vliegher and Bas 't Hoen of Egeniq for their hard work implementing the mobile Apps and the demo server-side framework
- René Scheffer and Menno van de Laarschot of Stroomt for re-designing the user experience
- Charly Rohart of RCDevs for integrating tiqr in their software and developing the PAM module with tiqr support
- Petra Boezerooy of the SURFnet/Kennisnet subsidy programme for supporting the initial proof-of-concept that led to the development of tiqr

9 REFERENCES

[1] OATH, "VeriSign Introduces Collaborative Vision to Drive Ubiquitous Adoption of Strong Authentication Solutions", OATH website, February 2004, http://www.openauthentication.org/news/040223

[2] D. M'Raihi, M. Bellare, F. Hoornaert, D. Naccache, O. Ranen, "HOTP: An HMAC-based One-T ime Password Algorithm", RFC 4226, The Internet Society, December 2005, http://tools.ietf.org/html/rfc4226

[3] D. M'Raihi, S. Machani, M. Pei, J. Rydell, "TOTP: Time-Based One-Time Password Algorithm", RFC 6238, The Internet Society, May 2011, http://tools.ietf.org/html/ rfc6238

[4] P. Hoyer, M. Pei, S. Machani, "Portable Symmetric Key Container (PSKC)", RFC 6030, The Internet Society, October 2010, http://tools.ietf.org/html/rfc6030

[5] R.M. van Rijswijk, M. Oostdijk, "Applications of Modern Cryptography", SURFnet, September 2010, http://www.surfnet.nl/documents/sncryptoweb.pdf

[6] D. Raywood, "Google adds two-factor authentication to Gmail via SMS one time passwords",

SC Magazine UK, September 2010, http://www.scmagazineuk.com/googl e-adds-two-factor-authentication-to-gmail-via-sms-one-time-passwords/article/ 179266/
[7] A. Song, "Introducing Login Approvals", Facebook, May 2011, https://www.facebook.com/note.php?note_id=10150172618258920&comments
[8] M. Oostdijk, M. Wegdam, "Mobile PKI, a technology scouting", Novay & SURFnet/Kennisnet, December 2009, http://www.terena.org/news/community/download.php?news_id=2528
[9] Wikipedia, "QR code", last visited June 2011, http://en.wikipedia.org/wiki/QR_code
[10] Vasco DIGIPASS GO range, last visited June 2011, http://www.vasco.com/products/digipass/digipass_go_range/digipass_go.aspx
[11] RSA SecurID, last visited June 2011, http://www.rsa.com/node.aspx?id=1156
[12] Feitian OTP tokens, last visited June 2011, http://www.ftsafe.com/products/otp.html
[13] Vasco DIGIPASS Readers, last visited June 2011, http://www.vasco.com/products/digipass/digipass_readers/digipass_readers.aspx
[14] SafeNet SafeWord GOLD, last visited June 2011, http://www.safenet-inc.com/uploadedFiles/About_SafeNet/Resource_Library/Resource_Items/Product_Briefs_-_EDP/SafeNet_product_brief_GOLD.pdf
[15] A. Litan, "SMS/OTP under attack – Man in the Mobile", Gartner, September 2010, http://blogs.gartner.com/avivah-litan/2010/09/28/smsotp-under-attack-man-in-the-mobile/
[16] J. Kaavi, "Strong authentication with mobile phones", Helsinki University of Technology, Fall 2010, http://www.cse.hut.fi/en/ publications/B/11/papers/kaavi.pdf
[17] "Specification for Integrated Circuit(s) Cards Interface Devices revision 1.1", DWG Smart-Card Integrated(s) Card Interface Devices, April 2005, http://www.usb.org/developers/devclass_docs/DWG_Smart-Card_CCID_Rev110.pdf
[18] "ZXing (Zebra Crossing) multi-format 1D/2D barcode image processing library", last visited June 2011, http://code.google.com/p/zxing/
[19] D. M'Raihi, J. Rydell, S. Bajaj, S. Machani, D. Naccache, "OCRA: OATH Challenge-Response Algorithms", Draft RFC, The Internet Society, March 2011, http://tools.ietf.org/html/draft-mraihi-mutual-oath-hotp-variants-14
[20] B. Kaliski, "PKCS #5: Password-Based Cryptography Specification", RFC 2898, The Internet Society, September 2000, http://tools.ietf.org/html/rfc2898
[21] "OpenSC – tools and libraries for smart cards", last visited June 2011, http://www.opensc-project.org/opensc
[22] "PKCS #11: Cryptographic Token Interface Standard", RSA Laboratories, June 2004, http://www.rsa.com/rsalabs/node.asp?id=2133
[23] "ETSI TR 102 203 v1.1.1 – Mobile Signatures; Business and Functional Requirements", ETSI, May 2003, http://docbox.etsi.org/EC_Files/EC_Files/tr_102203v010101p.pdf
[24] "ETSI TR 102 204 v1.1.4 – Mobile Signature Service; Web Service Interface", ETSI, August 2003, http://docbox.etsi.org/EC_Files/EC_Files/ts_102204v010104p.pdf
[25] "ETSI TR 102 206 v1.1.3 – Mobile Signature Service; Security Framework", ETSI, August 2003, http://docbox.etsi.org/EC_Files/EC_Files/tr_102206v010103p.pdf
[26] "ETSI TR 102 207 v1.1.3 – Mobile Signature Service; Specifications for Roaming in Mobile Signature Services", ETSI, August 2003, http://docbox.etsi.org/EC_Files/EC_Files/ts_102207v010103p.pdf
[27] "YubiKey", last visited June 2011, http://www.yubico.com/yubikey
[28] "Installing Google Authenticator", last visited June 2011, http://www.google.com/ support/accounts/bin/answer.py?answer=1066447
[29] "SURFfederatie – federated identity management for simpler cooperation and greater ease of use", SURFnet, 2007, http://www.surfnet.nl/Documents/attachment.db@189185.pdf
[30] "The Netherlands", CM International / CM Telecom, last visited June 2011, http://www.cmtelecom.com/premium-sms/netherlands
[31] R. Fergusson, "One third of mobile owners notice phone is missing within 15 minutes", Engineering & Technology Magazine, March 2011, http://eandt.theiet.org/news/2011/mar/securenvoy-survey.cfm
[32] N.R. Wagner, "The Laws of Cryptography: Verhoeff's Decimal Error Detection", University of Texas San Antonio, 2002, http://www.cs.utsa.edu/~wagner/laws/verhoeff.html
[33] "PHP QR Code – QR code generator, an LGPL PHP library", last visited June 2011, http://phpqrcode.sourceforge.net/
[34] "SimpleSAMLphp", last visited June 2011, http://simplesamlphp.org/
[35] "JSON – JavaScript Object Notation", last visited June 2011, http://www.json.org/
[36] "TiQR Authentication Server", last visited July 2011, http://www.rcdevs.com/products/ tiqr/
[37] "Add tiqr Two-Factor Authentication Mechanism", Shibboleth JIRA, last visited July 2011, https://issues.shibboleth.net/jira/browse/IDP-91
[38] "The Information Card Ecosystem - Information Cards", Information Card Foundation, last visited July 2011, http://information card.net/

[39] "smartSD", SD Association, last visited July 2011, https://www.sdcard.org/developers/tech/smartsd/
[40] "Introducing AlterEgo – 1.5 Factor Authentication for Web Apps", The Rocket Science Group, last visited August 2011, http://blog.mailchimp.com/introducing-alterego-1-5-factor-authentication-for-web-apps/
[41] D. Winder, "Blizzard introduces most powerful World of Warcraft weapon ever", June 2008, last visited August 2011, http://www.itwire.com/business-it-news/networking/19106-blizzard-introduces-most-powerful-world-of-warcraft-weapon-ever
[42] Egeniq, SURFnet, "TokenExchange for iPhone, iPad, Android and BlackBerry device tokens", last visited August 2011, http://code.google.com/p/tokenexchange/
[43] Egeniq, SURFnet, "Example implementations of the OATH OCRA algorithm in various languages", last visited August 2011, http://code.google.com/p/ocra-implementations/
[44] DarkReading, "National Survey Finds 1 in 3 Mobile Phone Owners Would Know They've Lost Their Phone Within 15 Minutes", last visited August 2011, http://www.darkreading.com/insider-threat/167801100/security/client-security/229400606/national-survey-finds-1-in-3-mobile-phone-owners-would-know-they-ve-lost-their-phone-within-15-minutes.html
[45] R. Boyd, J.L. Davis and C. Mastrangelo, "Teraflop Troubles: The Power of Graphics Processing Units May Threaten the World's Password Security System", Georgia Tech Research Institute, last visited August 2011, http://www.gtri.gatech.edu/casestudy/Teraflop-Troubles-Power-Graphics-Processing-Units-GPUs-Password-Security-System