n-Auth: Mobile Authentication Done Right

Roel Peeters
Jens Hermans
Pieter Maene
KU Leuven/COSSIC (Belgium)
firstname.lastname@kuleuven.be

Katri Grenman
Kimmo Halunen
Juha Häikiö
VTT Technical Research Centre of Finland
firstname.lastname@vtt.fi

ABSTRACT
Weak security, excessive personal data collection for user profiling, and a poor user experience are just a few of the many problems that mobile authentication solutions suffer from. Despite being an interesting platform, mobile devices are still not being used to their full potential for authentication. n-Auth is a firm step in unlocking the full potential of mobile devices in authentication, by improving both security and usability whilst respecting the privacy of the user. Our focus is on the combined usage of several strong cryptographic techniques with secure HCI design principles to achieve a better user experience. We specified and built n-Auth, for which robust Android and iOS apps are openly available through the official stores.

ACM Reference format:

1 INTRODUCTION
It is generally acknowledged that password authentication is problematic from many perspectives. With the ever more frequent usage of smart phones and tablets, entering passwords is even more of a burden: on soft keyboards, opposed to traditional keyboards, entering a password takes considerably more time and results in more frequent errors [21]. While passwords suffer from major security and usability problems, passwords are widely used and are likely to remain so in the near future. This is mainly because password-based authentication is cost-effective, widely supported by applications and accepted by users as a security mechanism.

A plethora of alternatives for passwords, mainly aimed at web authentication, have been proposed. Bonneau et al. [5, 6] presented an overview of the most promising ones, reaching the conclusion that none of these is a worthy replacement for passwords. One of the discussed alternatives is the Pico, introduced by Stajano [22]. The Pico was deemed an interesting proposal but no more than that, due to the lack of an implementation, and the impossibility to assess the user experience.

The Pico is a dedicated hardware token to authenticate the user to a myriad of remote servers; it is designed to be very secure while remaining quasi-effortless for users. The Pico authenticates the user using public key credentials, making common attacks on passwords (e.g., sniffing, phishing, guessing, social engineering) impossible. From a privacy and security perspective the Pico solution is also better than single-sign-on solutions, since users manage the credentials themselves instead of relying on a third party. However, in its quest for mass adoption, the Pico concept has evolved into a glorified personal password manager [23], which suffers from some of the same drawbacks it originally set out to solve.

Inspired by the original Pico concept, we developed n-Auth, a mobile authentication solution. n-Auth solves several of the open challenges that the Pico concept poses, both on a technical level and in terms of usability.

Concretely, our contributions are:

- A complete technical specification of n-Auth, containing a communication architecture with detailed cryptographic protocols to ensure a high level of security in Section 3 and provisions to smoothen future security upgrades in Section 4.
- Robust and freely available Android and iOS applications (Section 5), allowing n-Auth to reach maturity at negligible cost per user and making it possible to do user testing. We also define a comprehensible user interaction model for n-Auth taking Human-Computer Interface (HCI) design principles into account.
- In Section 6 we discuss our preliminary usability study, with indicative results in Appendix B.
- In Section 7, we compare n-Auth to both Pico concepts, showing our improvements.

2 IDEA
Instead of typing a password into the browser (client) on a computer or smart phone to log in to a website (server), the user will use n-Auth to authenticate himself, together with the client, to the server directly. As a result, the user will be logged in at the client.

In our general setup, we make a distinction between the following entities:

- the client, i.e., the primary user interface to interact with the application that requires authentication (e.g., a browser);
- the n-Auth device, e.g., the smart phone (or a dedicated device) on which the n-Auth app is installed and which the user uses to perform the actual authentication;
- the server, i.e., the endpoint which the n-Auth device will interact with to perform the authentication. The server will...
make a decision on the success or failure of authentication and ensures that the client gets logged in. This process depends on the specific client-server interaction mechanism which is out of scope for this paper.

Note that the entities do not need to be separated and can coincide on the same device. The client may also be on the n-Auth device, e.g., when an app on the smart phone is used as the primary user interface. The client can also coincide with the server, e.g., when the user directly authenticates to a terminal.

Figure 1 gives an overview of how the user will authenticate to a server with the aid of an n-Auth device. n-Auth can be used in several different scenarios, with different types of clients and/or servers. For generic clients, such as web browsers, it should not be necessary to install any add-ons: i.e., no browser extensions, hardware add-ons, drivers etc. This means that the n-Auth device should be able to communicate directly to the server and get all the necessary information from the client (server identifier and a server-generated session identifier for the client). With a browser, this information can be acquired by the n-Auth device, for instance, by scanning a QR code or similar visual code. If the client runs on the same smart phone as the n-Auth app, the user can simply tap the visual code or a button. With a specific URI scheme the data can be passed directly between apps.

The n-Auth device will also be used to create an account with a server, using a similar process. In order to prevent users from accidentally registering instead of logging in to a server, the n-Auth device will only offer a single entry point to start the authentication process. The provided scannable code (or URI) will encode additional information that allows the n-Auth device to determine the purpose: logging in with an existing account or creating a new account at the server. Different scannable codes for every purpose is also consistent with current practices, e.g., the entry web page for a server has distinctly marked locations for logging in and creating a new account.

The user interface has to provide context of the authentication to the user. There is a big difference between logging in to a newspaper website and logging in to your online banking account. The context was clear to the end-user using different user names, more secure passwords for sensitive applications, easier passwords for less important ones, smart cards in combination with readers … However, this is no longer the case when scanning visual codes. For the user to establish context and give an informed decision of authorization, he will be asked for explicit confirmation after scanning the visual code, before the n-Auth device authenticates to the server.

Our solution requires changes to the server, which initially hurt deployability. However, being more and more confronted with attacks, e.g., password breaches of widely used services occur on a regular basis, it is clear that passwords on their own will not be sufficient in the future, and that changes to the server will be necessary. Because we create an extra channel between the n-Auth device and the server, we are not restricted to the traditional SSL/TLS connections and can easily deploy another protocol without interfering with the existing server-client connection. We deliberately chose not to base our main security protocol on SSL/TLS but instead adapted the SIGMA-I protocol [14] for the n-Auth application. This is mainly based on the poor security track record (which will require frequent updates of both the server, and the app or mobile operating system in case the SSL/TLS system libraries are used) and issues around (public key) client authentication of the SSL/TLS protocols. Furthermore, we believe that SSL/TLS is overly complex and does not reflect recent advances in protocol design. This can clearly be seen from the algorithm agility perspective: SSL/TLS relies on negotiation of the algorithm suite, allowing for a myriad of algorithms, several of which are sub-optimal security-wise.

3 SOLUTION

We now proceed with a detailed technical description of the different processes that underlay the authentication with n-Auth.

3.1 Login

One needs to establish a link between the client-server session and the device-server session. This link will be established indirectly

1There even exist websites where one can check if their login was among the ones being compromised, e.g., https://haveibeenpwned.com/.
by having the server communicate with the user’s client, the client with the n-Auth device and finally the n-Auth device with the server.

The client will set up a connection with the server, and receives a message SCD (Server-Client-Device) that contains a server identifier and a session identifier. This SCD message will be transferred by the client to the n-Auth device over an authenticated channel. By requiring an authenticated channel, we ensure a proper binding between the client and n-Auth. However, this does not ensure that the transferred message is confidential. Thus, in order to avoid session hijacking at the client, the session identifier should remain separate from the typical browser session identifiers (as found in cookies). Therefore, the session identifier in the SCD message will be the hash of the browser session identifier.

\[ SCD = server_{ID}, H(session_{ID}), LOGIN. \]

The device uses the server identifier in its account database to uniquely identify the server. How this database is populated is described in Section 3.3. The user is presented with all his accounts on the server with the given server identifier and asked to explicitly confirm logging into this server with the selected account. Afterwards the n-Auth device selects the corresponding private/public key pair from its database and authenticates to the server. Our authentication protocol (see Figure 2) is based upon the SIGMA-I protocol as proposed by Krawczyk [14]. The original proofs by Canetti and Krawczyk [8] still hold for our construction. SIGMA-I is a mutual authentication protocol with key agreement that is very efficient and has the added benefit that the device can delay sending its (account-specific) identity until it is convinced of the server’s identity. The shared key \( K \) is derived using a key derivation function (KDF) from an unauthenticated Diffie-Hellman key agreement (here instantiated with the elliptic curve variant). By having each party sign (SIG and VER) the exchanged Diffie-Hellman values, the origin of the messages is established. As already suggested by Krawczyk, we use an authenticated encryption (AE and AD) mode [12] to optimize the protocol.

Because we use the same protocol for different purposes (login, enrol), the purpose will explicitly be encoded in the type that is part of the signed message. Since the device already knows the identity of the server, this value can be omitted from the protocol. For the identity of the device, we use the selected public key. Before accepting the device, the server will also verify that the provided credentials are registered in its database. The provided session\textsubscript{ID} allows the server to link its n-Auth session to its client session. This concludes the 3-way handshake, with messages DS1, SD2 and DS3 as in Figure 2.

### 3.2 Continuous Authentication

The key \( K \) that was derived during the handshake can be used for exchanging subsequent messages MSG between the device and the server. These messages are of the form:

\[ MSG = IV, AE_{K, IV}(type, body) \]

where type indicates the type of message and body the optional body of the message. Authenticated encryption requires a unique initialization vector IV for each message. This will be enforced by incrementing the IV each time a new message is encrypted. An IV that has not been increased will lead to the termination of the connection.

After establishing a secure channel, the device and server engage in a continuous authentication protocol. The server sends a MSG with type PING and a timeout value to the device. The device needs
to reply with a MSG of type PONG within the specified time frame. Otherwise, the client session will be closed and the user logged out.

The user can also end a session directly through the client, in which case the server attempts to send the device a MSG with type BYE to alert the device that the user is no longer logged in. Likewise, the user can close a session on the n-Auth device, in which case device attempts to send BYE to the server.

### 3.3 Enrolment

In the case of enrolment, the server is not yet known to the n-Auth device. Hence, it requires the essential parameters from the server, such as the server’s public key and its validity period, the actual URL for connecting to the server (not necessarily the same as the URL where the user connects to with his client), the name and logo of the server as it is communicated to the general public etc. To keep the SCD message that is used for enrolment small, only the (tiny) URL to retrieve the server parameters and a hash of the server’s public key will be added:

\[
SCD = serverID, H(sessionID), ENROL, URL, H(serverPK).
\]

The server parameters will be requested by the device by sending a \texttt{INIT} message to the received URL. The server will reply with an \texttt{INIT} message that contains the server parameters, which are signed with the server’s private key. The device checks that the received public key of the server is correct by checking it with the hash value received in the SCD message and then verifies the signature over the server parameters.

When setting up a new account, the device creates an entry in its database with the server parameters and the selected private/public key pair after successfully receiving SD2 and verifying the server’s identity. The server will create a new entry in its database for the received public key after successfully receiving DS3 and having validated the signature using this key as in Figure 2.

### 3.4 Database locking mechanism

To avoid being vulnerable to theft and so-called lunch-time attacks (leaving the n-Auth device unattended for a short while), there is a database locking mechanism in place with the following properties. Note that this locking mechanism works on individual accounts (making it possible to tweak security according to the server’s needs) and not on the database as a whole.

It is based on a single secret that needs to be provided by the user. This secret can be a PIN for the user to remember, a biometric like for instance a fingerprint or coming from another device the user carries around like for instance a smart watch. To avoid brute force attacks on the n-Auth device, the secret is verified online and the server implements a lock-out mechanism based on the number of consecutive failed verifications. The verification is done in zero-knowledge, i.e., the server does not learn the secret. The latter allows us to use the same secret for all possible servers, and use any available server for verification of the secret.

As long as the n-Auth device is not locked out and upon successful verification of the secret, the server will reply to this n-Auth device with some information that allows it to compute a series of symmetric keys that will be forgotten over time. These symmetric keys can then be used to decrypt private keys that are stored encrypted in the n-Auth device’s database. When access to a private key is needed and the symmetric key to decrypt it is no longer available on the device, the user will again be asked to provide its secret for online verification. The servers can specify these timeouts, which can be rather long depending on the application (or even infinite for servers that do not require two-factor authentication).

### 4 FUTURE PROOF

A crucial aspect to take into account when designing the cryptographic protocols for n-Auth is algorithm and key agility. It is very likely that in the future certain algorithms will need to be replaced or key sizes increased due to security considerations. Keys will also need to be changed, regardless of algorithm changes.

#### 4.1 Algorithm Agility

Popular cryptographic protocols (e.g., SSL/TLS) allow for negotiation of the parameters. Even though this is great for backwards compatibility, this comes with the risk of an adversary negotiating both parties in the weakest protocol parameters\(^2\). We have opted for a different approach, by strictly limiting the number of protocol parameters. In essence, we only want to allow a single protocol and associated parameters, which should maximally satisfy all security requirements. The only reason to allow multiple (in this case: two) parameter sets is to allow for fast upgrading in case one of the ciphers or protocols gets broken.

We make a distinction between the temporary and permanent parameters. The temporary parameters are set by the server: the hash function, the elliptic curve and key derivation function for key agreement; and the authenticated encryption scheme. The permanent parameters define the signature scheme that is used. For both the temporary and permanent parameters we define two versions, one to be used initially and one to enable migration in the future to a more secure set of parameters. There is no intent to maintain an extensive list of broken parameters for backward compatibility.

The server determines which version of the temporary parameters (\texttt{pVersion}) is used, there is no margin for negotiation on the n-Auth device side. This allows for a very fast migration of the temporary parameters, which is possible since no long term keys are involved. The n-Auth device should reject weak temporary parameters. The version of the temporary parameters is passed on to the n-Auth device as a part of the SCD message.

For the permanent parameters a similar approach is desirable, but due to long term keys being involved it is impossible to do instantaneous upgrades. Using the SCD message the server indicates the preferred permanent parameter set. During the transition of the server switching from one version to the next, it is possible for the n-Auth device to set the permanent version (\texttt{pVersion}) to the previous version. When the transition period is over, it will no longer be possible for the user to upgrade his account and the user will be forced to create a new account.

This means that we have to augment the handshake to set up the secure connection as follows:

\begin{itemize}
  \item [a)] the n-Auth device sends the version of the permanent parameters along with its first message to the server (DS1);
\end{itemize}

b) upon receiving $DS_1$, the server sets $t_{Version}$ and checks if the received $p_{Version}$ is allowed or aborts otherwise;

c) in $SD_2$, the server puts a signature on $(SERV\_AUTH, A, B, p_{Version}, t_{Version})$;

d) in $DS_3$, the n-Auth device puts a signature on $(type, A, B, p_{Version}, t_{Version})$.

The last two modifications will prevent man-in-the-middle attackers from exploiting a version mismatch between n-Auth and server.

Directly after the handshake the server will request that the n-Auth device updates its key according to the new version. This is done by sending a $MSG$ with type $UPGRADE$ and a signature with the server’s new public key (suitable for the new version) on $(UPGRADE, A, B, p_{Version}, t_{Version})$. Note that the $p_{Version}$ in this and the following message are now set by the server. The n-Auth device will get the server’s new public key and validity period by requesting the server parameters through a separate request to the server with an $INIT$ message (see Section 3.3).

The n-Auth device generates a new key pair and makes a new entry in its database, using the received parameters and server public key. The n-Auth device will send a $MSG$ to the server of the type $UPDATE$ with its new public key(s) and the signature(s) with the new authentication private key(s) on $(UPDATE, A, B, p_{Version}, t_{Version})$. Upon validation of the signature, the server updates its database by overwriting the existing entry for this user. From this point onwards the n-Auth device can no longer authenticate to the server using the previous $p_{Version}$, even if the transition period is still active. The server sends a confirmation ($MSG$ of type $CONFIRM\_UPDATE$) to the n-Auth device, that can now remove the old account from its database.

4.2 Key Agility

A similar protocol will be used for the server to update its public key without migrating to a new version of algorithm parameters. This allows the server to update its key before it expires or to replace its key if there is doubt about the private key being compromised.

The handshake protocol will be augmented by having the server sending two signatures (during the transition period) in $SD_2$, one with the new key and one with the old key. The n-Auth device will try to validate the first signature (new key), and upon failure the second signature (old key). When both signatures fail to verify, the connection will be terminated. Given that the first signature fails and the second signature verifies, the n-Auth device requests the new server parameters (by sending a separate $INIT$ message) and checks the first signature, using the new public key of the server. If this signature verifies (new key is correct), the n-Auth device updates the selected account in its database. Even if this signature does not verify, the n-Auth device will still continue by sending $DS_3$ but not update its database.

5 IMPLEMENTATION

In this section we describe the design choices of the implementation of the user interface, different smart phone operating systems and other technical details of the n-Auth.

5.1 User Interface

We now go deeper into the user interaction with the application and the underlying HCI design principles. Molich and Nielsen [16] provided ten usability recommendations, among others that a usable system needs to be consistent and that it is better to prevent errors in the first place than being good at handling these. The n-Auth device only offers a single button to start the authentication process (see Figure 3a). By clicking this button, the n-Auth device will start its camera to capture a 2D visual code.

The principle of psychological acceptability [20] implies that a user interface must be designed while keeping the user’s mental models in mind and thus, by extent, the context he is working in. Yee’s key principles for secure interaction design [27] require users to provide explicit authorization and visibility of all active sessions. For the user to establish context and give explicit authorization, the user will be presented with a confirmation screen after scanning the code, before the n-Auth device authenticates to the server (step 3a in Figure 1). Depending on the purpose a different confirmation screens will be shown to the user. Examples are given in Figure 3b, 4a, and 4b.

After scanning a code for a login, the user will be asked to confirm his login, by selecting the desired account (see Fig. 3b). This approach works for one as well as for multiple accounts existing with a given server.

Visibility of all active sessions means that, from the n-Auth device, the user should be able to see to which servers he is currently logged in and have the possibility to log out (see Figure 3d). To log out, we choose to have the user swipe a label off the screen to avoid logging out by accidental tapping. A helper text is displayed the first time the user logs in using his n-Auth device.

We assume open registrations whereby creating a new account is started from within the client, in which the user first enters some basic account information, e.g., a user name. Then the user will be shown a visual code to be scanned with the n-Auth device. This visual code contains the necessary parameters of the server and the selected user name. On the n-Auth device, the user first needs to confirm the creation of a new account. This confirmation screen displays the server’s logo and url location, to help the user with establishing context. The server parameters will be verified by the user by comparing the displayed visual hash with the server’s reference image (see Figure 4a). However, if the user already has other accounts at the same server, he can be relieved from this burden. The n-Auth device will check that the received server parameters correspond with the ones of the existing accounts. In this case, the user is presented with a different screen, as depicted in Figure 4b.

5.2 Smart phone specifics

Because the majority of smart phones today runs on either Android or iOS, we choose to develop an application for these operating systems. By making our application freely available through the official Google Play Store4 and the official Apple iTunes Store5, it is easy to install, also for less tech-savvy users.

The n-Auth application fully respects the user’s privacy, by only requiring the minimal permissions (networking and camera) for

---


5https://itunes.apple.com/us/app/n-auth/id1176128845

The server’s identity remains the same over all permanent versions.
providing its functionality and not profiling the device and or the user beyond the model of the device (to ease account management) and the profiling done by Android and iOS itself\(^6\).

When browsing on the same device as our n-Auth application is installed on, it is possible to transfer the data in the visual code directly to the n-Auth application. By clicking on the visual code, the n-Auth application will open, do the authentication, and go back to the mobile browser. At the same time the n-Auth application will keep on running in the back, as will be apparent [Android version only] from the n-Auth icon in the notification bar (see Fig. 5a). This icon will be visible as long as the user is logged in at some server, keeping Yee’s [27] principle of visibility of all active sessions in mind. From the notification drawer (see Fig. 5b) it is possible to go directly to the n-Auth application, and even to close the n-Auth application, logging out of all currently logged in sessions.

---

\(^6\)Note that we do not make use of any analytics frameworks like, e.g., Google Analytics, Crashlitics …

\(^7\)https://www.n-auth.com/dev/

---

5.3 Technical

The n-Auth server is implemented as a RESTful API to ensure easy integration with the application backend server (where the actual service runs that needs authentication). The n-Auth server handles the coupling of users by the backend server, all communication with n-Auth device and open sessions. Through the API the backend server can request if there is a user logged in for a certain session identifier, and if so which user. Through the developers’ web portal\(^7\), one can set up their own server instance for testing and interact with it through the API. Note that this is on a shared server and one obviously loses the No-Trusted-Third-Party benefit.

For mobile browsing, we also implemented a direct link on the visual codes in the HTML source, and made sure that the n-Auth application listens for Intents. This way a user can simply click on the visual code to launch the n-Auth application and pass on the data needed to start authenticating to the server.

All protocol messages are encapsulated using Google protocol buffers and sent over a plain HTTP connection between the n-Auth device and the Java application. HTTP was mainly selected for simplicity of the implementation and to easily pass through firewalls. This however implies that all protocol round-trips must be initiated by the n-Auth device and require a connection identifier.
For this connection identifier we chose the Diffie-Hellman key $B$
since this is randomly generated by the server. For efficiency we
however do prefer a dedicated, permanent TCP connection. In case
of loss of connection, it can be recovered using $B$.

All cryptographic operations have been implemented using the
NaCl [4] library, which is very efficient and has constant time im-
plementations. The used elliptic curve, Curve25519 [2], has several
security advantages compared to the NIST curves. We make use of
the Ed25519 signature scheme [3] scheme, which in comparison with
the widely-used ECDSA signature scheme is less vulnerable to leakage
of the secret key in the case of bad randomness. For authenticated
encryption, we use secret_box, which combines a stream cipher
xsalsa20 and a one-time mac function poly1305.

The client’s database locking mechanism is implemented using a
4 digit PIN, where the 20 most commonly chosen PINs\(^8\) are excluded.
This was done to ensure that the success probability of an attacker
trying the most commonly used PIN codes is drastically reduced
(26.83% of all PINs could be guessed by entering the ones in the top
20). Note that this database locking mechanism is optional for each
account, depending on the server’s needs. The server implements
a lock-out mechanism that triggers after 3 consecutive failed PIN
verifications, meaning that after 3 consecutive failed verifications
of the PIN, the device can no longer get the information from the
server needed to unlock the database on the client. The default PIN
timeout is 2 minutes, meaning that the user will not be requested
to input his PIN where he normally would, e.g. to login to another
account, if the last input of the PIN was less than 2 minutes ago.

A login (excluding user interactions) and the database unlocking
mechanism are comparable in the time required and their communi-
cation overhead. The average time required is 0.4 seconds (tested on
a range of Android devices in different setups). This time is mainly
dominated by the time needed to communicate back and forth with
the server (timings of the cryptographic operations on the devices
are in the order of tens of milliseconds), also depending on the used
wireless connection: WIFI (typically the longest time needed, WIFI
is optimised for high throughput at the cost of latency), 3G, 4G.
The communication between the n-Auth device and the server is in
the order of kiloBytes. For enrolment the time and communication
overhead is slightly larger, mainly depending on the size of the logo
to be transferred to the n-Auth device.

6 PRELIMINARY USER STUDY

To evaluate the usability of n-auth, we created a survey website:
https://www.n-auth.com/survey/. Participants were asked to login
daily using n-Auth over the course of two weeks. Each day there
was a new set (each taking less than two minutes to answer) of two
to three short questions on authentication in general and on the
usability of the n-Auth application.

First, participants had to download the n-Auth app from the
official store and create an account on the site using n-Auth. At the
time of creating an account, we assigned participants at random to
one of two groups: a group for which the PIN code mechanism is
enabled and group for which no PIN code is required.

The questionnaire was provided in the native language of the
users (Finnish). The questions are translated to English and provided
in Appendix A for the convenience of the reader.

The entire study was conducted online. Participants were re-
cruited through the user base of Owela (online survey platform)
and mailings to student associations of a nearby university. Partici-
pants could win 2 film tickets (12 in total) if they participated for
at least 6 days. Participants could also opt-in to receive daily email
reminders to participate in the study over the course of two weeks
when the study took place.

Limitations of this study are that (1) no interviews with the par-
ticipants were conducted, we only had ad hoc email conversations
with some of them for trouble shooting; (2) the only demographic
information collected from the participants is their age groups; (3)
the participants had to login to a new service, i.e., it is not a service
they already logged in to on a regular basis; (4) this study was done
for a single website, not showing the n-Auth’s full potential.

The results of this usability study can be found in Appendix B.
Note that this preliminary study is only indicative as the number
of participants (24) does not allow to draw statistically relevant
conclusions.

7 COMPARISON TO PICO

The Pico concept went through an evolution after the initial pub-
lication [22]. In order to bootstrap the Pico ecosystem, a smart
phone application was proposed [24]. The paper introduced two
ways that the Pico can work, by either performing a mutual au-
thentication with the website (as in the original work) or by using
a browser plugin “Pico lens”, that rewrites websites so they appear
Pico compatible. The latter option makes the Pico send passwords
to the website through HTTPS and then transfer the cookie to the
web browser instead of performing a mutual authentication.
Essentially, this option turns the Pico into a password manager
that is triggered by the user through a visual code. In subsequent
work [23] the authors completely drop the original Pico and con-
firm that the password-based approach combined with “Password
Manager Friendly” (PMF) annotations[25] whereby the website al-
 lows password-manager-created passwords without imposing its
customary composition policy on these as long as these as suffi-
ciently long (level 2) is the new ‘native mode’ Pico. The paper also
defines level 0 (no changes to the server) and level 1 (only PMF
annotations) as the Pico in ‘compatibility mode’. This evolution
of the Pico concept loses several features such as continuous au-
thentication and the ability to log out from the device; and comes
with the intrinsic disadvantages of transferring passwords. The
very recently published usability study (April this year) on Pico [1]
seems to suggest the that Pico team has abandoned their browser
plugin for a reserve proxy setup at the server and is using the Pico
prototype (only Android) in ‘compatibility mode’ (user has to enter
his password in the app, no indication whatsoever that the Pico
updates the user’s password to a longer, hard-to-guess password)
without any kind of database locking mechanism (no mention of
Pico Siblings as in the original concept or even the use of PIN
or fingerprint). In this light, we will evaluate the current Pico in
‘compatibility mode’ without database locking mechanism.
Bonneau et al. [6] described a general Usability-Deployability-Security evaluation framework, using a concise list of properties, which we will use for our comparison. Table 1 shows the properties attributed to the original Pico description by Stajano [22] as per et al. [6], supplemented with our evaluation of their new direction [1, 23] and our n-Auth implementation. In the following sections, we always first state each property together with a short explanation, taken verbatim from [6].

Table 1: Properties for Pico and n-Auth.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Resilient-to-Physical-Observation</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Resilient-to-Targeted-Impersonation</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Resilient-to-Throttled-Guessing</td>
<td>•</td>
<td>○</td>
<td>•</td>
</tr>
<tr>
<td>Resilient-to-Internal-Observation</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Resilient-to-Leaks-from-Other-Verifiers</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Resilient-to-Phishing</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Resilient-to-Theft</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>No-Trusted-Third-Party</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Requiring-Explicit-Consent</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Unlinkable</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Deployability</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accessible</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Negligible-Cost-per-User</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Server-Compatible</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Browser-Compatible</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Mature</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Non-Proprietary</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Scalable-For-Users</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Nothing-To-Carry</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Physically-Effortless</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Easy-to-Learn</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Efficient-to-Use</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Infrequent-Errors</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Easy-Recovery-from-Loss</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Memorywise-Effortless</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>

7.1 Security

Resilient-to-Physical-Observation. An attacker cannot impersonate a user after observing them authenticate one or more times. This is clearly achieved by all three since no credentials need to be entered by the user at login. n-Auth can make use of a PIN as a second factor (database locking mechanism), however knowledge of the PIN alone is not enough to log in.

Resilient-to-Targeted-Impersonation. It is not possible for an acquaintance (or skilled investigator) to impersonate a specific user by exploiting knowledge of personal details (birth date, names of relatives etc.).

Resilient-to-Throttled-Guessing. An attacker whose rate of guessing is constrained by the verifier cannot successfully guess the secrets of a significant fraction of users.

Resilient-to-Unthrottled-Guessing. An attacker whose rate of guessing is constrained only by available computing resources cannot successfully guess the secrets of a significant fraction of users.

Both the original Pico and n-Auth make use of public key authentication, for which the private keys are assumed to be hard to guess, thus achieving the three properties above. However given that in current Pico, the user has to input his existing password for each account into the Pico app, we only rate it Quasi on these three properties. This is based on users’ well-established poor track record in password selection and the fact that some of them even keep passwords written down in plain sight. Note that Pico in ‘native mode’ achieves these three properties.

Resilient-to-Internal-Observation. An attacker cannot impersonate a user by intercepting the user’s input from inside the user’s device (e.g., by key-logging malware) or eavesdropping on the cleartext communication between prover and verifier (we assume that the attacker can also defeat TLS if used, perhaps through the CA).

Given that the original Pico is a dedicated hardware token, which can be assumed to be malware-free, it achieves this property. The current Pico relies on TLS to send over a user name and password (static data) to the server, hence it does not achieve this property. Since key storage (database), the SIGMA-I protocol and the PIN input (custom keypad) are handled by the n-Auth app directly, malware on the user’s device will not be able to get information needed for logging in.

Resilient-to-Leaks-from-Other-Verifiers. Nothing that a verifier could possibly leak can help an attacker impersonate the user to another verifier.

Both the original Pico and n-Auth only store public keys at the verifier, from which deriving private keys is assumed to be hard. Hence, leaking these public keys will not even help the attacker impersonate the user to the verifier that leaked its public key database. The current Pico cannot fully achieve this property, since users are known for re-using their passwords. Note that Pico in ‘native mode’ achieves this property.

Resilient-to-Phishing. An attacker who simulates a valid verifier (including by DNS manipulation) cannot collect credentials that can later be used to impersonate the user to the actual verifier.

All three approaches protect against phishing by embedding the server identity in the visual code and fetching the URL from their devices’ internal databases.

Resilient-to-Theft. If the scheme uses a physical object for authentication, the object cannot be used for authentication by another person who gains possession of it.

Both the original Pico and n-Auth are Quasi-Resilient-to-Theft, the former due to the Pico Siblings, the latter as it relies on the modest strength of a PIN, even though attempts are rate-controlled. The current Pico does not seem to have a database locking mechanism, hence it does not achieve this property.

No-Trusted-Third-Party. The scheme does not rely on a trusted third party (other than the prover and the verifier) who could, upon
being attacked or otherwise becoming untrustworthy, compromise the prover’s security or privacy.

None of the three approaches relies on a trusted third party.

**Requiring-Explicit-Consent.** The authentication process cannot be started without the explicit consent of the user.

Explicit user consent is built into the n-Auth device’s user interface. Both Pico concepts require user interaction for logging in, however no explicit consent from the user is required after getting the necessary context (the user consents to log into a website, but it could be any website for which he has an account), hence we only rate them Quasi-achieving this property\(^9\). Given that the current Pico seems to be for one account on one website only, the context is clear and hence we rate it to achieve this property fully.

**Unlinkable.** Colluding verifiers cannot determine, from the authenticator alone, whether the same user is authenticating to both.

Both the original Pico and n-Auth achieve this by generating new keys for each account. The current Pico cannot fully achieve this due to password re-use by users. Note that Pico in ‘native mode’ achieves this property.

7.2 Deployability

**Accessible.** Users who can use passwords are not prevented from using the scheme by disabilities or other physical (not cognitive) conditions.

None of the three approaches is accessible as these require alignment of the device with the screen monitor to scan a visual code.

**Negligible-Cost-per-User.** The total cost per user of the scheme, adding up the costs at both the prover’s end (any devices required) and the verifier’s end (any share of the equipment and software required), is negligible.

The original Pico concept with a dedicated hardware token clearly does not achieve this property. Both the current Pico and n-Auth are mobile apps and can thus achieve this property.

**Server-Compatible.** At the verifier’s end, the scheme is compatible with text-based passwords. Providers don’t have to change their existing authentication setup to support the scheme.

This is only achieved by the current Pico, as both the original Pico and n-Auth rely on public key authentication.

**Browser-Compatible.** Users don’t have to change their client to support the scheme and can expect the scheme to work when using other machines with an up-to-date, standards-compliant web browser and no additional software.

This is achieved by both the current Pico and n-Auth. For the original Pico, the user must install a browser plugin and possibly a hardware dongle for setting up a secure short range communication between the browser and the Pico.

**Mature.** The scheme has been implemented and deployed on a large scale for actual authentication purposes beyond research.

As the original Pico is but a concept it does not achieve this property. We rate the current Pico as Quasi-Mature, given the recent usability study that occurred in the wild in a cooperation with a Alexa top 500 website on an Android prototype, available to registered beta testers. We rate n-Auth as Mature, given that it has reliable, publicly available Android and iOS applications. Furthermore, a preliminary user study was carried out in the wild (see Appendix B) and it is currently being deployed on the central login system of KU Leuven, a major European University.

**Non-Proprietary.** Anyone can implement or use the scheme for any purpose without having to pay royalties to anyone else. The relevant techniques are generally known, published openly and not protected by patents or trade secrets.

This is a design goal of both the original Pico and the current Pico. n-Auth only partially achieves this property.

7.3 Usability

**Memorywise-Effortless.** Users of the scheme do not have to remember any secrets at all.

This is achieved by both Pico approaches, the original by using Pico Siblings, the current one by not having a database locking mechanism. n-Auth only achieves Quasi-Memorywise-Effortless as users have to remember one PIN for the database locking mechanism. In future work, we foresee to also enable fingerprint recognition for the same purpose, thus achieving Memorywise-Effortless.

**Scalable-for-Users.** Using the scheme for hundreds of accounts does not increase the burden on the user.

Both the original Pico and n-Auth are scalable-for-users. The usability study on the current Pico seems to suggest that the Pico prototype was build specifically for one account with this one website, considerably simplifying the user interactions (as there is no general enrolment).

**Nothing-to-Carry.** Users do not need to carry an additional physical object (electronic device, mechanical key, piece of paper) to use the scheme.

The original Pico concept does not achieve this. Both the current Pico and n-Auth Quasi-achieve this as the users need to carry their mobile phones, which is something they would carry everywhere at all time anyway.

**Physically-Effortless.** The authentication process does not require physical (as opposed to cognitive) user effort beyond, say, pressing a button.

This is achieved by all three approaches.

**Easy-to-Learn.** Users who don’t know the scheme can figure it out and learn it without too much trouble, and then easily recall how to use it.

The original Pico is possibly not Easy-to-Learn owing to the complexity of the Pico Siblings management. For both the current Pico and n-Auth, results suggest that these are Easy-to-Learn.

**Efficient-to-Use.** The time the user must spend for each authentication is acceptably short. The time required for setting up a new association with a verifier, although possibly longer than that for authentication, is also reasonable.

**Infrequent-Errors.** The task that users must perform to log in usually succeeds when performed by a legitimate and honest user.
For the scoring of the above two properties at least a prototype is necessary, as such the original Pico is not scored\(^{10}\). With respect to efficient to use, we score both the current Pico and n-Auth as Quasi-Efficient-to-Use with average user timings for the former of 35 seconds (42 authentication events over 11 participants) and the latter of 33 seconds (167 authentication events over 24 participants). For the original Pico, the user has to select one of the two dedicated buttons (login, create an account) to start the scanning of the QR code. This approach is prone to errors, as the user might accidentally hit the wrong button when scanning the QR code and does not need to confirm his choice. The current Pico, not having a general enrolment procedure, does not have this problem (as it can only login after credentials were entered). For both the current Pico and n-Auth, results suggest that these achieve Infrequent-Errors.

**Easy-Recovery-from-Loss.** A user can conveniently regain the ability to authenticate if the token is lost or the credentials forgotten.

None of the three approaches achieves this property. From a security point of view, Easy-Recovery-from-Loss usually implies a security vulnerability that might get exploited by an attacker.

## 8 RELATED WORK

Since user authentication is researched from many different perspectives such as, e.g., security, privacy and usability, many proposals can be seen as related to our work. Therefore, we only focus on user authentication systems that (can) make use of mobile phones.

**Phooaproof Phishing Prevention** [18] is one of the earliest works on bringing authentication to a cellphone (back in 2006, far from the current smart phones). A shared key is established between the server and the app in an out-of-band manner. Later on when the user wants to authenticate, a browser plugin outsources the TLS client authentication (based on a pre-shared secret) to the app over Bluetooth. The authors developed a prototype, which they used to conduct a usability study.

In [26] a QR-code scheme is proposed to confirm a (banking) transaction through an OTP (similar to [13] discussed further on). Snap2Pass/Snap2Pay [9] authenticates the user through a challenge presented in the QR code on which a MAC is computed using a symmetric key (shared between the app and server). At registration this shared key is transferred into the app by scanning a QR code, making it vulnerable to physical observation at this time. As it is the web service that generates these shared keys, these can be assumed to be hard-to-guess and unlinkable. Note that there is no protection against theft. Public key authentication is presented as an option, and a logout from the app feature is discussed as a security extension. They implemented a demonstrator Android app and an OpenID provider. A 2015 usability study comparing seven different web authentication systems [19] found that both Snap2Pass and Google OAuth 2.0 are favorite by users, where the former benefits from a coolness factor (participants loved that they could use their smartphones and obviate the need for passwords) and the latter from reputation (“I trust Google with my passwords”). In the case of Snap2Pass, the study also revealed that users are concerned with theft and have some concerns about not being able to login if they forget their smartphone.

\(^{10}\)Contrary to Bonneau et al. [6] that scores it Quasi- and remarks that in order to give it the full score, they first need to see it before they believe it.

tiqr[7] provides an open source (both apps and server) user authentication system. They published robust Android and iOS apps in the respective official stores. tiqr makes use of the OAuth Challenge Response Algorithm (OCRA) [17], which generates a one-time password based on the challenge and a shared secret (between client and server, which is transferred from the client to the server over TLS at enrolment). On the client, the shared secret is in turn protected under a 4 digit PIN. The challenge is transferred from the browser to the client in a QR code that also contains a server identifier. Next the user is shown the URL and asked if he wants to proceed (explicit consent), then prompted for the PIN, after which the client authenticates by sending the response to the server over TLS. tiqr does not have any kind of session management on the app (one-shot authentication, no possibility to logout from the device) and relies on symmetric key cryptography making the server more vulnerable to attacks, even though no symmetric keys are directly transmitted outside the enrolment procedure. It also misses some flexibility with respect to the PIN.

QR-codes have been proposed as a method in physical access control in [13]. This system is a simple extension of a RSA-based one-time password (OTP) scheme to QR-codes. The paper does not describe an implementation.

SQRL [10] derives a private key from a master secret and the server’s domain name, which the client uses to sign a challenge and log in to the server. The client can be either the browser itself or a mobile app which obtains the challenge and domain name by scanning a QR code. The whole system relies on TLS for secure communication and server authentication. Several proof-of-concept implementations for both server and client are available, but there does not appear to be any production-ready software.

Loxin [28] makes use of a central Loxin server and central PKI that certifies Loxin apps. After entering his identifier (username) into the browser, a cryptographic hash value is shown in the browser. The central Loxin server then sends a push message with that hash to the mobile phone. After confirming in the app, the app signs a random nonce, coming from the web service, using its private key. After the signature is verified, the user gets logged in. There exists an open source implementation of the Loxin Android app (EasyChip) and backend server. While users in principle do not have to scan a QR code, this approach still requires the user to enter his username into the client. This also implies that anyone that knows the username can trigger the Loxin app on the user’s phone. The major security disadvantages of this approach are its use of trusted third party, it is not resilient to theft and it is linkable across different service providers as there is only one public key.

More recently, a comprehensive system (Knock x Knock) for web authentication was proposed [11]. This work centers on managing the user’s different accounts on an iPhone and basing security decisions on some contextual information e.g. the location of the user. This work does not consider the privacy issues of authentication and is more like a general purpose password manager.

In the area of standardization the FIDO Alliance has proposed the Universal Authentication Framework (UAF) standard [15] for a passwordless experience. The standard specifies the interface that authenticators offer, the authentication itself and also the resolving process, in which an appropriate authenticator with the right
security capabilities is selected. Contrary to our approach, where we try to rely as little as possible on the client, UAF is heavily focused on the client device. For example, for a website, the authentication would be started from the browser through a javascript API, after which the browser would try to resolve an appropriate authentication on the client machine. UAF also lacks proper mutual authentication, continuous authentication and log out functionality as it relies on TLS and signatures from the authenticator.

9 CONCLUSION

We described n-Auth, a mobile authentication solution done right, which is inspired by the original Pico concept. We provided a detailed specification of the user interface and cryptographic protocols with provisions for future security upgrades. Our robust Android implementation demonstrated that our design is feasible, deployable and allows to evaluate its usability at negligible cost. The results of our preliminary usability study are encouraging and show a positive trend towards n-Auth being a more secure and convenient alternative for passwords.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their constructive comments. Pieter Maene is an SB PhD fellow at Research Foundation - Flanders (FWO).

REFERENCES


[20] Pieter Maene is an SB PhD fellow at Research Foundation - Flanders (FWO).
• Fingerprints
• Voice recognition
• Face recognition
• Token, e.g., YubiKey, RSA security token
• Soft tokens (mobile)
• Other, please specify in comments

Day 3.
• Which login methods do you feel are suitable for Banking services/Social media/Discussion forums? [check boxes authentication methods]
• Comments [open-ended question]

Day 4.
• Which login methods do you like to use for logging in daily? [check boxes authentication methods]
• Comments [open-ended question]

Day 5.
• How easy is it to use the n-Auth method? [Likert scale from very difficult(1) to very easy(5)]
• Comments [open-ended question]

Day 6.
• Compared to passwords, logging in using n-Auth is …
  - [Likert scale from much less safe(1) to much safer(5)]
  - [Likert scale from much less convenient(1) to much more convenient(5)]
  - [Likert scale from much slower(1) to much faster(5)]
• Comments [open-ended question]

Day 7.
• With which of the following services could you imagine using n-Auth to log in? [check boxes applications]
• Would you prefer to use n-Auth in combination with a PIN code? [yes/no]
• Please explain [open-ended question]

Participants could choose from the following applications:
• Banking services
• Health services (including access to medical records)
• Governmental services (e.g., social insurance, taxes, police)
• Email
• Online stores
• Social media
• Discussion forums
• News sites
• Other, please specify below

Day 8.
• Do you know any similar applications to n-Auth? [open-ended question]
• How does n-Auth compare to these? [open-ended question]

Day 9.
• How safe do you feel using n-Auth to log in? [Likert scale from very unsafe(1) to very safe(5)]
• Comments [open-ended question]

Day 10.
• How easy is it to use the n-Auth method? [Likert scale from very difficult(1) to very easy(5)]
• What do you think n-Auth should improve on? [open-ended question]
• Would you recommend the n-Auth method to others? [yes/no/perhaps]

B RESULTS OF THE USER STUDY
B.1 Participants

Figure 6 gives an overview of the participation in our user study. In total, the sign up button was pressed 65 times, for which we identified 39 unique users. Of those 39 participants, 5 never succeeded in registering n-Auth and 2 more succeeded in registering n-Auth but did not answer any questions. Of the 32 participants in our user study, 24 logged in the second day to answer the next set of questions. As the study progressed, more participants dropped out. Having only 4 participants answering the last day’s questions can be partially explained by:

1. the cut-off day for being eligible to win the reward being on day 6; and
2. we only sent email reminders during the two weeks the study took place, even though participants could fill in the questions for one more week (for those that missed a day or even a couple of days).

Taking a closer look at the 24 participants that logged in the second day. The users represented different age groups well (see Figure 7), with the youngest users being from the age group 18-24 and the oldest user from the age group 65+. Roughly 2 out of 3 participants used the Android n-Auth app, while the remaining 1/3 used the iOS n-Auth app. During registration participants were assigned at random to make use of the n-Auth authentication method with or without the PIN database locking mechanism (see Section 3.4). For 40% of the participants, the PIN database locking mechanism was enabled.
B.2 Getting started

As can be deduced from the number of failed initializations and the drop-out rate of the users, the system proved to be somewhat hard to use. As multiple registrations for the same email addresses were observed from the server logs, some users were contacted directly in order to determine whether they had problems accessing the system. Using this method we were able to solve some issues people had when trying to access the system.

One issue we identified was the availability of the application only in English. Although Finnish people are usually quite fluent in English, having the application available in their own native language might have helped some of the users. Some also experienced problems when installing the n-Auth app, some due to security measures on their own (company issued) phones when downloading new applications. Other participants were not that experienced in installing new applications on their smart phones, a set of instructions for installing and finding your new applications on your phone could have also been helpful.

A key element that some participants were missing was a more thorough set of instructions for use. For the purposes of this study, a very compact check list of instructions for installing and using the application was written on the Owela page associated with this trial. The instructions weren’t clear enough in presenting the difference between creating a new account and logging in when using an existing one. Although a feedback page was provided in Owela, few users reported their problems on the site. The instructions detailed two different use cases: using the survey with a computer after scanning the code with a mobile phone; and using the survey with a mobile phone by clicking on the code. It would seem that these instructions could have been clearer, as some users had trouble figuring out what and when to do with their mobile phone.

Some users criticized the inadequate guidelines and visual look and feel on the actual survey page. One of the main issues was that on the registration page, there were no instructions available on the actual use of the application. The registration form and the code being visible simultaneously on the same page caused confusion, and the only instructions available were about the structure of the study.

B.3 Previous experience

Participants were asked what authentication methods they are currently using. All of the 24 participants who responded were used to using PIN codes. Login methods based on passwords (19 users), social media login, security codes on paper and fingerprint were also commonly used among test users. One user used face recognition, but none were using voice recognition for logging into services. Figure 8 illustrates the current login methods used by our participants.

B.4 Findings on n-Auth

On the first day, a 5-point Likert scale (ranging from 1 very difficult to 5 very easy) and a related open-ended question were used to collect data about the ease of adoption and first impressions regarding n-Auth. 32 participants responded to the Likert scale question and the average value of the responses was 3.4 (neutral,
Figure 10: Willingness of users to use different authentication methods on a daily basis

leaning towards easy). 8 participants (25%) considered the adoption of n-Auth either very difficult or rather difficult and 19 participants (60%) perceived the adoption as very easy or rather easy. Three participants (10%) considered the adoption of the n-Auth method as neither difficult nor easy. Many of the participants who considered the adoption process to be difficult, thought it was confusing. Participants with more positive Likert scale responses described the adoption of n-Auth as quick and simple.

Figure 11 illustrates the distribution of the participants’ responses on a 5-point Likert scale. In addition to the perceptions of the participants as a whole, the figure also presents the difference between participants using n-Auth with and without a PIN code. The average value of the Likert scale responses concerning the ease of adoption was 3.3 among PIN code users and 3.6 among those users who did not use a PIN code.

Figure 11: Perceived ease of adoption

In the fifth day of the user study participants were asked to evaluate the ease of use with a 5-point Likert scale. 12 out of 20 (60%) respondents considered the use of n-Auth rather easy or very easy. 6 respondents (30%) perceived the use rather or very difficult and 2 users (10%) neither difficult nor easy. The average of score for the responses was 3.7. Figure 12 presents the distribution of all responses regarding the perceived ease of use of n-Auth from three perspectives: all participants, the participants using n-Auth with a PIN code and the participants using n-Auth without a PIN code. The users who did not use the PIN code, found the use of n-Auth to be easier.

Figure 12: Perceived ease of use of n-Auth

Users were also asked to evaluate the ease of use through a similar response option scale on the tenth and final day of the study. The average value of responses on the final day was 3.6 - almost the same value as on day 5. On the final day the amount of responses was lower as only 4 participants persisted until the very end of the study.

Participants were also asked to compare passwords and the n-Auth method from three perspectives: security, convenience and speed. The participants evaluated these aspects on a 5-point Likert scale (ranging from 1 being much less secure/much less convenient/much slower to 5 being much more secure/much more convenient/much faster than passwords). Figure 13 presents the average of the responses from all users combined, separately for PIN code users and users without PIN code.

Figure 13: Comparing n-Auth to passwords

The users without a PIN code thought the n-Auth method was slightly faster, more convenient and secure than a traditional password. Conversely, the users with PIN codes enabled felt the n-Auth method was slower, less convenient and only as secure as a password.
B.5 Timings

Figure 14 shows the box plot of the times (24 participants) needed to register n-Auth (the time from the website displaying the QR code, that one needs to scan on the n-Auth device, to the actual registration of n-Auth at the server). Overall, it takes the participants that needed to put in a PIN code only 50% longer. Given that it was probably their first account protected with a PIN, they had to enter a non-trivial PIN code and confirm it before being able to register. One also has to account for users not having their smart phone at hand or still having to install the n-Auth app.

![Figure 14: Time required to register n-Auth.](image)

Figure 14 shows the average times each individual participant (167 logins by 24 participants) needed to login using n-Auth (the time from the website displaying the QR code, that one needs to scan on the n-Auth device, to the actual login of n-Auth at the server). The process varied greatly, ranging from 5 to 7 seconds to up to 2 minutes. This probably depends on the participants having their smart phone at hand or not.

![Figure 15: Time required to login using n-Auth.](image)

B.6 Other findings

On day 7, the users were asked to estimate which services they would be willing to use with n-Auth authentication. The answer choices given were banking services, health services, governmental services (such as taxes), email, online stores, social media, discussion forums and news sites. Out of the 19 respondents on day 7, as many as 15 were willing to use n-Auth when shopping online. Another popular use possibility was email (12 respondents) or banking (9 respondents).

When commenting on their choices, some felt that they would be ready to use the n-Auth method if they could opt to use the PIN code with it to provide an extra layer of security. On the other hand, the use of several different passwords and PIN codes puts a load on people’s memory. As some users, who were given PIN codes to use with n-Auth, commented in the beginning of the study, it’s “yet another PIN code to remember”. In case one authentication method can substitute many others and be multi-functional, it could provide users with a welcome relief from remembering a large selection of different access codes. This is often the reason why people use social media authentication, as it provides a quick and easy way to log in and register to new services.

On day 8, users were asked whether they knew any services like the n-Auth. Most users didn’t have experience on similar applications, although some mentioned mobile phone certificates, NFC tags and one user provided a link to tiqr.

Unfortunately, the response choices for day 9 were confused with another day, and the question and its multiple choice options did not correspond with each other. Thus, the results from day 9 were not usable.

The main problems users had with the application were related to feelings of confusion that could have been alleviated with a more thorough set of instructions, and written separately for people using the survey page on a mobile device and a computer. Especially the registration process was difficult to understand, and logging in the first time proved to be complicated. After the first hurdle, however, the process became much clearer. Users wanted a clearer layout and user interface and felt it seemed a bit unpolished.

Unfortunately there were some errors during the trial period, and some users reported getting error messages. Despite this, most users persisted and their answers were logged into the system. For some who were using the system on a computer, it seemed a bit cumbersome to dig out their phone, start the application, find the link to the survey from their email, scan the code and fill in the survey. Of course this was partly due to the structure of the study, as the survey site was not a service they would have normally visited and required a separate link.

B.7 Conclusion

Users were happy with how quick the method was to deploy. It felt quite safe to most, especially if coupled with the additional PIN code. If the method was accompanied with a set of comprehensive instructions and a clear quick start guide, it might have a more positive impact. On the positive side is also the fact that using a method like this reduces the mental load for people, especially if used without a PIN code. If using n-Auth would allow people to log into several services without an extra set of credentials, it would please many people who find passwords and PIN codes frustrating and difficult to remember.

B.8 Lessons learnt

While we focused on simplicity, it is clear that we also need to provide better instructions and documentation to the user (as is also one of the design principles of Molich and Nielsen [16]).