Bootstrapping adoption of the Pico password replacement system

Frank Stajano, Graeme Jenkinson, Jeunese Payne,
Max Spencer, Quentin Stafford-Fraser, Chris Warrington
{frank.stajano, graeme.jenkinson, jeunese.payne,
max.spencer, quentin.stafford-fraser,
chris.warrington}@cl.cam.ac.uk

University of Cambridge Computer Laboratory, Cambridge, UK

Abstract. In previous work we presented Pico, an authentication system designed to be both more usable and more secure than passwords. One unsolved problem was that Pico, in its quest to explore the whole solution space without being bound by compatibility shackles, requires changes at both the prover and the verifier, which makes it hard to convince anyone to adopt it: users won't buy an authentication gadget that doesn't let them log into anything and service providers won't support a system that no users are equipped to log in with. In this paper we present three measures to break this vicious circle, starting with the "Pico Lens" browser add-on that rewrites websites on the fly so that they appear Pico-enabled. Our add-on offers the user most (though not all) of the usability and security benefits of Pico, thus fostering adoption from users even before service providers are on board. This will enable Pico to build up a user base. We also developed a server-side Wordpress plugin which can serve both as a reference example and as a useful enabler in its own right (as Wordpress is one of the leading content management platforms on the web). Finally, we developed a software version of the Pico client running on a smartphone, the Pico App, so that people can try out Pico (at the price of slightly reduced security) without having to acquire and carry another gadget. Having broken the vicious circle we'll be in a stronger position to persuade providers to offer support for Pico in parallel with passwords.

1 Introduction and motivation

For normal people, passwords are a pain. Their inadequacy in terms of both usability and security has been repeatedly pointed out [1,4]. As people must now handle dozens of accounts, passwords are a solution that can no longer scale. Yet passwords continue to dominate as the well-entrenched incumbent because, from the viewpoint of the verifier, they beat every alternative hands down when it comes to ease of deployment [3].

Pico [11], which we briefly describe in Section 2, is our ambitious long-term project to replace passwords with a more usable and more secure system that

Authors' preprint.
In B. Christianson et al. (eds), *Proceedings of Security Protocols Workshop 2014*, Springer LNCS.

will not require you to memorize any secrets[1]. In its quest to explore the entire solution space for the best possible solution in terms of usability and security, Pico is a clean-slate redesign that explicitly gives up on compatibility with passwords. It is immediately clear that, in the short term, this choice will harm Pico's deployability. Our rationale is that, in the long term, passwords will become so blatantly unacceptable that the world will eventually demand something better; and, by then, Pico will have undergone several cycles of prototyping and testing and will be ready for adoption as a user-friendly, secure and technically sound solution that both users and service providers consider an improvement.

Having said that, in order to be ready for adoption when the time comes, Pico has to be taken seriously by the stakeholders, both on the client side and on the server side. For this reason, while we continue to investigate and develop the architecture without considering ourselves constrained by backwards compatibility, we also intend to provide a plausible migration path from the current password-dominated scenario to a future one in which Pico has replaced passwords. Charting this path is the topic of this paper.

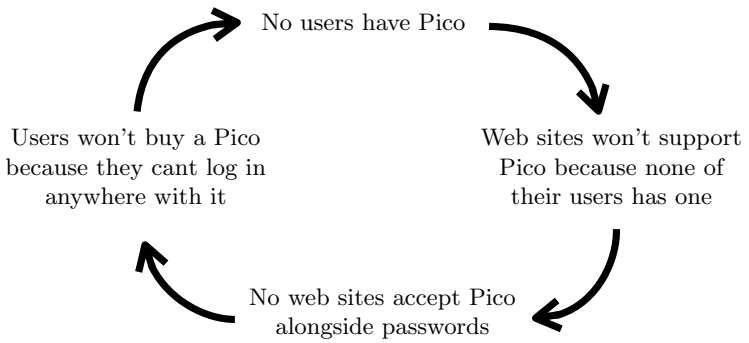

Fig. 1. The vicious circle opposing Pico adoption.

The main obstacle to widespread adoption of Pico is a classic vicious circle (fig 1). Organizations that authenticate their users with passwords are reluctant to change their servers to support an unfamiliar and unproven system, especially if it also requires outfitting every user with a physical gadget at non-zero unit cost. People, on the other hand, will be understandably reluctant to acquire, carry and use a new authentication gadget, even if genuinely easier to use than passwords, if it does not work with the services to which they wish (or need) to authenticate. Our strategy involves disrupting this vicious circle in several places. Where circular dependencies prevent users from adopting Pico before

[1] The project's website, `http://pico.cl.cam.ac.uk/`, contains a brief introductory video, the original paper, a FAQ and other resources.

servers have adopted it and vice versa, in this paper we present software modules we have developed to break such dependencies.

Our first contribution, on the client side, is the "Pico Lens" web browser add-on, described in Section 3: when you view a website through the Pico Lens, it appears Pico-enabled even if it isn't, so that you may authenticate to the website using your Pico device rather than by typing your password. This breaks the vicious circle for users because it allows them to use a Pico and reap most of its usability benefits even before their favourite websites start offering native Pico support. In turn, once enough people adopt Pico for its convenience, website operators have more of an incentive to support the full Pico authentication system alongside traditional password login.

Our second contribution, on the server side, is the "Pico Verifier" plugin for Wordpress[2], described in Section 4. It breaks the vicious circle for content providers because it allows webmasters of Wordpress-based websites to make their site Pico-enabled simply by installing the plugin, without any development effort. This plugin also provides a reference implementation of the server side, for webmasters who might wish to develop a Pico-enabled version of their non-Wordpress website.

Our third contribution, on the client side, is the "Pico Prover" app for Android smartphones, described in Section 5. This software breaks the vicious circle for users because it allows them to use their existing smartphone as a Pico device, without having to buy (or carry) any extra hardware. The Pico Prover app can to authenticate to both natively Pico-enabled verifiers and to non-Pico-aware websites viewed through the Pico Lens, so it lets users reap many of the usability and a few of the security benefits of a real Pico without any significant investment. If they like the user experience (which, by releasing the app early, we can refine and enhance while taking into account the feedback of many users in a crowd-sourced fashion), they may wish to upgrade to a dedicated Pico, which will eventually be smaller, simpler and more secure. The Pico Prover also provides a reference implementation for the client side.

After a brief overview of the Pico system in Section 2 to make the presentation self-contained, the rest of this paper describes each of these three contributions in greater detail.

2 The Pico architecture in brief

The Pico system consists of the Pico device itself (a small, dedicated and tamper-resistant hardware authenticator the size of a pedometer or a car key fob), acting as the prover, and a back-end acting as the verifier. Even though the technical contributions described in this paper focus on the use-case of web authentication,

[2] According to W3techs statistics (`http://w3techs.com/technologies/overview/content_management/all/`), as of February 2014, Wordpress is the most widely used Content Management System on the web, being used by 21.5% of all websites and by 60.0% of all websites that use a content management system.

in which the verifier is a website, in the general case any entity that authenticates its users (whether with or without passwords—think of car keys) could be augmented with a Pico back-end.

Pico relies on a multi-channel authentication protocol [14] in which an additional channel (acquisition of a QR code [7] in the current implementation) conveys the user's intent to authenticate to a designated verifier. The verifier signals that it supports Pico authentication by displaying a Pico visual code, perhaps alongside the conventional login prompt for user name and password. The human prover signals her intent to authenticate to a particular verifier by acquiring the verifier's QR code with her Pico device. This action initiates the execution of the Pico authentication protocol, which mutually authenticates the verifier to the Pico and the Pico to the verifier.

In the current implementation the Pico system uses the SIGMA protocol [8] for mutual authentication and generation of a symmetric session key. The Pico prover and back-end verifier create digital signatures to prove ownership of a public key, which is their long-term public identity. Pico has adopted the "I" variant of the SIGMA protocol [8], in which the verifier must authenticate its identity to the prover, before the prover reveals its identity to the verifier, preventing any privacy loss to verifiers presenting "counterfeit" visual codes. To protect the user's privacy further the Pico uses a different key pair for every account so that colluding verifiers cannot link accounts belonging to the same user. A run of the SIGMA protocol also yields a fresh symmetric session key, which the Pico and verifier use for continuous authentication.

In some cases, for example when logging into a local computer or when opening a Pico-enabled door, the Pico prover device talks directly to the Pico verifier that displays the QR code directly. In other cases, though, most notably when logging into web sites, a third device is involved: when you authenticate to a website with your Pico, you actually access your account for that website through the web browser of your normal computer. In such cases the website provides the authenticated Pico with a "session delegation token" (a cookie) that the Pico then transfers to the web browser to delegate[3] the session it has authenticated[4].

When the verifier is remote, as in the case of a website, the Pico needs a connection to the Internet and a connection to the user's web browser so that the Pico can transfer the session delegation token to the web browser after authenticating. In our implementation as of March 2014 the Pico connects to the user's computer via a Bluetooth Personal Area Network (PAN) and tunnels out to remote services via this interface as well so that Pico doesn't need to have its own Internet connection via WiFi or via a mobile phone network.

[3] Delegation is a process whereby a principal authorizes an agent to act on its behalf by transferring a set of rights.

[4] The session delegation protocol used by Pico is described in further detail in our other paper "I bought a new security token and all I got was this lousy phish—Relay attacks on visual code authentication schemes", also in these proceedings.

Pico offers continuous authentication, whereby the Pico device authenticates to the verifier at regular intervals without user intervention, so long as the Pico remains unlocked and in proximity of the computer running the web browser. The verifier may thus keep the session open for as long as necessary but close it immediately when the user is no longer present, minimizing the window of vulnerability during which another person could hijack the session if the user left the terminal unattended. In the current prototype we detect proximity with a heuristic based on Bluetooth signal strength, though in the future we plan to adopt a more secure distance bounding protocol [5,6].

As with any token-based authentication method, the Pico system must protect the token against misuse by others who might find or steal it. In our design this is achieved through the Picosiblings mechanism [11,12]: the Pico device locks up (with its credentials encrypted), pauses the continuous authentication of any active session, and stops authenticating new sessions whenever it cannot sense the "aura of safety" around its owner. The "aura" is defined by the proximity of a sufficient number of other electronic devices (the Picosiblings) worn by that person. A biometric sample and a connection to a home server also act as special Picosiblings that offer additional protection properties and allow remote revocation. Because this Picosibling-based locking mechanism is independent of the normal operation of the Pico device and, particularly, of the "vicious circle of adoption" alluded to above, it will not be discussed further in this paper.

The credentials stored in the Pico device are backed up automatically, in encrypted form, whenever the Pico is plugged into its docking station for recharging. Backup, too, despite being a fundamental component of the Pico architecture, is independent of the "vicious circle of adoption" and will not be discussed further in this paper.

3 The "Pico Lens" browser add-on

The "Pico Lens" is a web browser add-on that rewrites websites on the fly to make them appear as if they support Pico alongside password authentication. The Pico Lens detects web pages containing login forms and adds a Pico visual code to them, alongside the existing username and password fields, so that Pico users have the option of authenticating with their Pico instead of typing their password.

Although the underlying methods used by the Pico to authenticate to Pico-enabled and only *Lens-enabled* websites are quite different, our aim was to make the user experiences as similar as possible. The end result in both modes of operation is the same: the user's web browser receives a session cookie granting access to the user account. Behind the scenes, however, what happens is rather different.

3.1 Pico authentication

For comparison, authentication to a fully Pico-enabled website (that is, a website that supports the real Pico authentication protocol) is described by the following sequence of events.

1. The user, whose web browser has a Pico add-on installed[5], navigates to a login page.
2. The login page includes a visual code encoding the public key of the service. If the full Pico Lens add-on is installed, it detects that the website is Pico-enabled (for example through a `pico-enabled` HTML `meta` tag.), and refrains from rewriting the page.
3. The user scans the visual code with their Pico.
4. If the user has multiple accounts for the website, they select one from the list displayed by the Pico. If the user only has a single account for the website, as is common, this step is skipped.
5. The Pico and the website mutually authenticate.
6. The website sends the Pico a fresh session delegation token, which for a web authentication takes the form of a set of cookies and a URL.
7. The Pico sends the session delegation token to the local terminal via the Bluetooth PAN.
8. The Pico add-on causes that browser to navigate to the URL contained in the session delegation token.

When a website is not Pico-enabled, the Pico Lens add-on allows the user to follow the same work flow, despite the differences in the underlying mechanism.

1. The user, whose browser has the Pico Lens add-on installed, navigates to a login page.
2. The Pico Lens detects a login form on the page, and displays an authentication visual code containing the domain name of the website (Fig. 2.)
3. The user scans the code with their Pico.
4. If the user only has a single account with the website, as is common, this step is skipped and the Pico proceeds to mutually authenticate the website. Otherwise the user selects an account for the website from the list displayed by the Pico (for each account the Pico device holds the username/password credentials used to authenticate).
5. The Pico internally loads, fills in and submits the login form using its stored username/password credentials for that account, tunnelling an end-to-end HTTPS connection to the website through the Bluetooth PAN established with the local computer.

[5] To perform Pico authentication with a Pico-enabled website, the Pico Lens, which rewrites legacy login pages to add a QR code to them, is clearly not required; however, *some* Pico browser add-on is still needed for receiving session delegation tokens from the Pico device.

6. The Pico receives the website's response to the form submission and creates a session delegation token consisting of any cookies set in the response and the address it was redirected to.
7. The Pico sends the session delegation token to the user's computer via the Bluetooth PAN. In the web browser, the Pico Lens installs the set cookies and follows the redirect, so that the user is logged in.

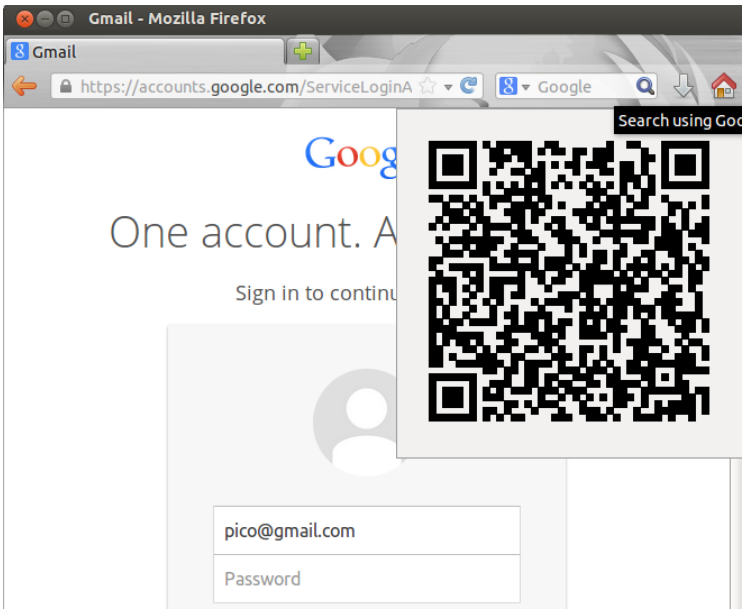

Fig. 2. QR code presented by the Pico Lens add-on for authentication to a website that does not natively support Pico.

3.2 Pico Lens pairing

Before a user can log into a Lens-enabled website with their Pico, they must store on in its encrypted storage their account credentials for that site and the Pico must learn how to fill out the login page on behalf of the user. The interaction is somewhat different to the "pairing" interaction with a Pico enabled site. Assume the user already has a password-based account with the web site and now wants to be able to log in with their Pico when viewing the site through the Pico Lens.

1. The user, whose web browser has the Pico Lens add-on installed, navigates to a login page.

2. The Pico Lens add-on detects a login form on the page and thus rewrites the page[6] to display an "authentication" visual code containing the domain name of the web page (Fig. 2).
3. The user, having not yet stored any credentials to their Pico for this website, ignores this first QR code, types their username and password in the login form and submits it as if without Pico.
4. The Pico Lens add-on detects the form submission, captures the name-value pairs being submitted, and offers to save these credentials on the user's Pico (Fig. 3), much as an in-browser password manager[7] would.
5. The user accepts this offer and the add-on displays a "pairing" visual code containing the submitted credentials[8], as well as the domain name of the website.
6. The user scans the visual code with their Pico.
7. The user confirms the pairing details and the username and password credentials are saved in the Pico's encrypted database, with the website domain as their lookup key.

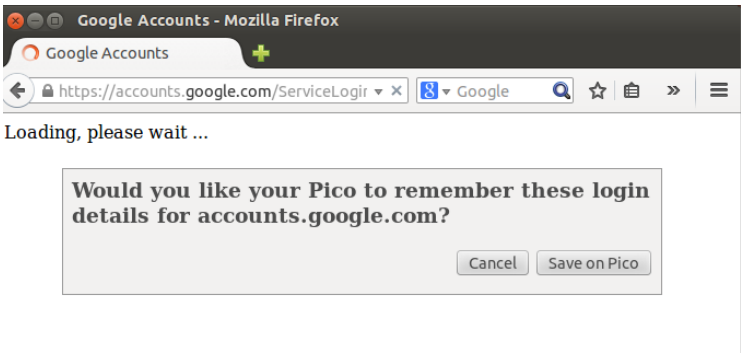

Fig. 3. Dialogue presented by the Pico Lens add-on after a login form is submitted.

[6] The Pico Lens at this stage has no idea that this is going to be a first-time pairing rather than a regular login, so it behaves exactly as in the previous case.

[7] We call "password manager" a piece of software that records username-password pairs on behalf of the user and supplies them to verifiers as appropriate, saving the user from having to remember and retype them. A password manager may be a standalone program or it may be integrated in a web browser. Password managers may store their database locally or in the cloud, and in cleartext or in encrypted form. The latter case provides greater security but requires entering a master password.

[8] The pairing code is currently unencrypted. If the visual code is observed during the account pairing, the attacker gains the user's password for that website.

3.3 Pico Lens design trade-offs and future work

The current implementation largely mimics existing in-browser password managers but is still a work in progress at the time of writing. Design decisions still to be finalized, possibly with the help of user studies, include at least the following.

First, we could do better in step 6, security-wise, by encrypting the username and password to the Pico before encoding them into the QR code; but we'd need some key for encrypting the message from Lens to Pico, and what would be the most usable way of establishing that[9]?

Second, we might consider using radio rather than the visual channel in step 6, sparing the user from having to scan a visual code, although there may be technical difficulties on platforms where the add-on is not allowed to open a socket on the computer where the browser runs.

Third, instead of making it the responsibility of the user to choose a suitably strong password, the Pico itself might choose a random password as hard as the website's policy will allow, to ensure greater randomness than can be expected by letting a human choose a secret [2]; this would also ensure that passwords are different on each site, thus limiting the damage for the user in case of server-side compromise. The practical difficulty is that the Pico would have to be informed about the constraints imposed by the website's password policy (e.g. certain characters disallowed, password length not to exceed some limit, etc.)[10] or discover them by trial and error.

We assume that many web users will already be familiar with the concept of a password manager since most modern web browsers incorporate one that pops up and offers its services when appropriate. An *encrypting* password manager, even though it stores your passwords on your computer thus exposing them to network intruders[11], offers a significant improvement in both security and usability compared to plain passwords because it allows you to use stronger passwords that are all different and that you don't have to remember nor retype. The fact that Pico can be seen as a portable password manager will allow users to mentally associate the former to the latter, making it easier for people to adopt Pico because they won't have to form a totally new mental model for it.

In contrast to a password manager, the username-password pair is stored not on the computer running the web browser but within the encrypted storage of the Pico (just as the private-key cryptographic credentials would be in the fully-Pico-enabled use case [11,12]) making the credential more strongly protected against network attacks than with the in-browser password manager. Although

[9] Bearing in mind scenarios in which one Lens serves several Pico devices, as when several family members use the shared tablet in the living room.

[10] It would be nice if websites published their password policy in a uniform machine-readable form; and even nicer if they imposed no upper bounds on making passwords arbitrarily long and complicated. As argued by Bonneau and Preibusch [4], websites that impose such limits probably do so because they are not hashing their passwords.

[11] A risk that is greatly reduced with Pico, which is a dedicated device not intended to run other software.

the implementation is very different, we are keen to maintain consistency in the user's mental model ("it's my Pico that holds my credentials") between the two cases of visiting a Pico-enabled website and visiting a legacy website through the Pico Lens.

The Pico visits the target website's login page independently from the web browser, rather than sending the password to the web browser; therefore, if the terminal is compromised, only individual session cookies can be compromised, rather than the long-term password. In this respect, too, the Pico Lens is a security improvement on existing password managers.

The Pico Lens add-on cannot provide continuous authentications for non-Pico-aware websites because, in order for continuous authentication to work, the back-end must accept "pause" and "resume" methods for the session, besides the standard "start" and "stop". A session that has been paused is inactive but retains its state; this must be explicitly supported by the back-end verifier. A session with a non-Pico-enabled website can only be either "logged in" or "logged out".

In the implementation we had at the time of the SPW workshop, the Pico connected to the user's computer via a Bluetooth Personal Area Network (PAN) and tunnelled out to remote services via this interface. Setting this up is, whist reasonably straightforward, is non-trivial. Further work is required to ensure that the solution is easily deployable by geeks and grannies alike.

Because Bluetooth isn't universally deployed, especially on desktop machines, we also considered using audio as an alternative channel, on the basis that practically every computer supports audio nowadays. Like radio, audio does not provide data origin authenticity; it has lower capacity, it is more easily eavesdropped on, and it annoys the user. So we thought we might avoid the acoustic transducers and wire up the devices directly through their ubiquitous 3.5 mm phono connectors. This would provide data origin authenticity, higher capacity, lower susceptibility to eavesdropping and no acoustic disturbance. However, many users would find it too cumbersome to have to attach a cable, which they often wouldn't have with them anyway. Speaking of cables, it also felt quite odd that, despite everything having dedicated digital connections such as USB, we had to resort to hacking the audio I/O ports to achieve a simple data transfer in a portable way that did not require installing device drivers and so forth.

Since presenting this work at the SPW in March 2014 we have been working on replacing the Bluetooth PAN with a web-based rendezvous point. Whilst introducing the need to securely pair the Pico and web browser, removing the dependency on Bluetooth significantly simplifies deployment of the solution.

The current Pico Lens add-on implementation is a technology demonstrator of the core insight that we can offer users the Pico experience even before their favourite websites are Pico-enabled. For actual deployment, though, we may have to revisit some of our implementation choices. For example, besides the other

issues previously mentioned in this section, the leading browser is now Chrome[12] rather than Firefox, although their add-on architectures are rather similar.

4 The "Pico Verifier" website plugin

Wordpress, originally a blogging platform, is currently the leading content management system on the web[13]. Our "Pico Verifier" plugin for Wordpress implements the back-end side of Pico for any website running Wordpress. It allows users to log in to the website using the genuine Pico protocol (not a simulation, as would be the case with the Pico Lens) while of course still allowing traditional password-based authentication.

With the Pico Verifier plugin installed on a Wordpress website, the login page is modified to include a Pico visual code, alongside the usual username and password prompt. Unlike the visual codes added by the Pico Lens, the ones the Wordpress plugin adds are in the HTML returned by the web server. The Pico visual code contains the name, address and public key of the website (see Fig. 4).

To authenticate to the Wordpress website, the user scans this QR code with her Pico. Provided the Pico is already paired with her user account on the website, the Pico then initiates the mutual authentication protocol of section 3.1 with the website's Pico verifier (also provided by our Wordpress plugin). If authentication is successful, the Pico verifier returns to the Pico a session delegation token, which the Pico uses to delegate its authority to the web browser as discussed in section 2.

Users can create a Pico-enabled account by scanning another visual code (similar to the one shown in Fig. 4) which the plugin adds to the Wordpress account creation page. The plugin also adds a "Pico" section to the Wordpress account management page. Here users can unlink a Pico which is already linked with their account and a QR code is added to allow the user to link a new Pico. For administrators, new Pico-related settings are added to the site's settings page.

Normally, once mutually authenticated, the Pico and the service execute a continuous authentication protocol over the established secure channel. When the Pico is out-of-range of either its Picosiblings or the terminal, the session is first paused and then eventually terminated. Terminating a session is straightforward. Pausing is more difficult, as Wordpress wasn't designed with pause/resume in mind, and this feature isn't currently provided by our plugin.

To experience a Pico compatible Wordpress blog requires that both websites and web visitors install and configure some software. The demands placed on the website are relatively modest—to install our Pico Verifier Wordpress plugin. In contrast, each user (prover) is required to install and setup multiple pieces of

[12] As of January 2014, Chrome holds 55.7% market share, with Firefox a distant second at 26.9%, according to W3schools statistics (`http://www.w3schools.com/browsers/browsers_stats.asp`).

[13] See footnote 2.


```
{
  "serviceName": "Some Wordpress Blog",
  "serviceUri": "http://someblog.com/pico:8080",
  "servicePublicKey": "MFUw...iC8U",
  "signature": "MEAC...9uGn",
  "TYPE": "KeyAuthenticationVisualCode"
}
```


Fig. 4. Example visual code added to the login page by the Wordpress plugin.

software (browser add-on, Bluetooth device driver etc.) and to provide network connectivity between the Pico, their computer and the website. Our future work includes simplifying these requirements in order to make Pico easier to deploy.

5 The "Pico Prover" smartphone app

It seems prudent that an authentication token holding all your login credentials should not be a general-purpose computing platform, with unfettered networking facilities, on which users merrily install arbitrary code of dubious provenance [9]. For this reason (as well as to be simple and easy to use) the Pico client is ultimately intended to be a dedicated single-purpose hardware device rather than a smartphone app. However, users tend to be extremely reluctant to carry one more device. And a smartphone can already simulate, if not the form factor, at least most of the intended functions of a Pico. It therefore seems reasonable for us to release a smartphone app that will allow users to try out the Pico functionality at no cost—without having to acquire a physical Pico (which we haven't yet built anyway) nor having to carry one around (assuming they'd already carry their smartphone regardless). While they're trying out Pico, users may still retain traditional passwords for the few accounts they consider most valuable and only use Pico for their more numerous lower-value ones. The risk introduced by the possibility of their long term credentials being exposed by malware on the smartphone is therefore limited.

We consider a dedicated tamper-resistant hardware token and the use of a consumer computing device such as smartphone as being at opposite ends of a design spectrum. Within this spectrum there are several other interesting options, such as the use of a Trusted Execution Environment (TEE). A TEE is an isolated execution environment in which sensitive or security enforcing functions can be executed. The Protection Profile for TEE—produced by the Global Platform collaboration—targets the Common Criteria Evaluation Assurance Level EAL2 ("Structurally tested"). Thus an implementation of the Pico Prover App using this technology would offer security benefits over a basic smartphone app.

Despite TEEs based on ARM's TrustZone technology being present in over 100 million handsets, the software running in the isolated environment has, until recently, been tightly controlled by handset manufacturers. With the advent of the Samsung S4 the TEE has been opened up to third party developers and this trend is likely to continue.

Despite the security benefits provided by a TEE, ensuring a trusted path to the user is a significant residual problem: malware can't access the data segment of the TEE-secured Pico Prover app, but it could simulate its screen. Because Pico does not require the user to remember or enter secrets, such concerns are somewhat mitigated. However, careful thought is needed to ensure that a malicious app can't trick the user in some elaborate way or abuse the API between the rich and trusted side to extract sensitive data such as keying material. A detailed analysis of Pico executing within a TEE on a smartphone would make interesting future work.

During the transition phase, the smartphone app will support both the native Pico protocol and also the Pico Lens protocol. Therefore, an important constraint placed on the Pico Lens solution is that it should be as usable as the native Pico. Distributing the Pico Prover as a free app allows us to crowd-source feedback about the usability of the solution and ensure that we are meeting this goal.

6 User acceptance

To date the security and usability benefits of Pico have only been considered by technologists [3]. The Unified Theory of Acceptance and Usage of Technology (UTAUT) [13] applies to technology adoption the ideas of the more general Theory of Planned Behaviour (ToPB) from social psychology. The UTAUT model postulates that a user's intention to adopt a new technology is driven by four constructs:

Performance Expectancy
: The root component of which is *perceived usefulness*.

Effort Expectancy
: The root component of which is *ease of use*.

Social Influence
: Similar to *social norms* in the ToPB model.

Facilitating Conditions
: Objective factors that influence the *ease of adoption* of the technology.

As users often hold inaccurate notions of security and of the importance of security measures [1,10], the stated security benefits need to be reconciled with the benefits that end users perceive Pico to offer. However, without a working implementation and more importantly without compatible services to log in to, validating these benefits would be tricky. In our initial work we have focused on providing "Facilitating Conditions" that allow us to perform large scale user studies to validate Pico's assumed benefits.

The solutions presented in this paper allow Pico to be adopted by a broad user community and used with legacy services based on password authentication. The insights gained from users adopting Pico in a day-to-day setting will highlight what we got wrong and inform changes to our design. Reporting and acting on these findings is a significant and exciting part of the future direction of the Pico project.

7 Conclusions

We still believe that, in our quest to produce a more usable and more secure password replacement, it would be a mistake to limit our horizon to solutions that are compatible with passwords: it may well be that better solutions exist beyond that horizon, and we want the freedom to explore those regions of the design space too.

On the other hand, we fully realize that a realistic solution requires a plausible path to deployment and that, when encouraging major players to adopt Pico, we cannot act as if passwords weren't already a strongly entrenched incumbent. The vicious circle undeniably exists: websites won't have any incentive to support Pico authentication until users already have Pico devices, and users won't have any incentive to get a Pico unless it works with their websites of interest.

Our strategy is therefore to break this vicious circle in several places. We have shown how the Pico Lens browser add-on allows users to reap the benefits of a Pico device even before the websites support it. We have shown how the Pico Verifier website plugin allows webmasters (of Wordpress sites) to support Pico at no development cost. And we have shown how the Pico Prover smartphone app allows users to try out Pico for some of their accounts without having to buy anything or carry any additional gadgets.

A significant advantage of this bootstrapping strategy is that allows us not to compromise on the purity of the Pico design: the clean-slate Pico is incompatible with passwords, but the solutions presented above are stepping stones that bridge this compatibility gap because they interwork with both legacy passwords and Pico, thus allowing for a transition phase.

Our next step will be more organizational than technological: we need to get the website operators on board—especially the big ones. If we can demonstrate that a critical mass of users finds the Pico Prover app and the Pico Lens add-on to be more usable than passwords (and we'll have to work hard at simplifying the installation process) we'll be in a good position to persuade the big players that it's worth supporting Pico as an alternative authentication method. This will in turn attract more users and we'll finally move from a vicious to a *virtuous* circle.

8 Acknowledgements

We gratefully acknowledge the European Research Council for funding this research under grant 307224.

We also thank Roel Peeters et al. for their independent implementation of Pico and for sharing pre-publication drafts of their work "Towards Building the Pico: The Security Perspective" (still in submission at the time of writing), from which we adopted the SIGMA-I protocol for mutual authentication.

References

1. Anne Adams and Martina Angela Sasse. "Users are not the enemy". *Commun. ACM*, **42**(12):40–46, Dec 1999. ISSN 0001-0782. `http://doi.acm.org/10.1145/322796.322806`.
2. Joseph Bonneau. *Guessing human-chosen secrets*. Ph.D. thesis, University of Cambridge, May 2012. `http://www.jbonneau.com/doc/2012-jbonneau-phd_thesis.pdf`.

3. Joseph Bonneau, Cormac Herley, Paul C. van Oorschot and Frank Stajano. "The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes". In "Proceedings of the 2012 IEEE Symposium on Security and Privacy", SP '12, pp. 553–567. IEEE Computer Society, Washington, DC, USA, 2012. ISBN 978-0-7695-4681-0. http://dx.doi.org/10.1109/SP.2012.44.
4. Joseph Bonneau and Sren Preibusch. "The Password Thicket: technical and market failures in human authentication on the web". In "WEIS 2010", 2010.
5. Stefan Brands and David Chaum. "Distance-Bounding Protocols (Extended Abstract)". In "EUROCRYPT'93, Lecture Notes in Computer Science 765", pp. 344–359. Springer-Verlag, 1993.
6. Gerhard P. Hancke and Markus G. Kuhn. "An RFID Distance Bounding Protocol". In "Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks", SECURECOMM '05, pp. 67–73. IEEE Computer Society, Washington, DC, USA, 2005. ISBN 0-7695-2369-2. http://dx.doi.org/10.1109/SECURECOMM.2005.56.
7. ISO. "Information technology—Automatic identification and data capture techniques—QR Code 2005 bar code symbology specification". ISO 18004:2006, International Organization for Standardization, Geneva, Switzerland, 2006.
8. Hugo Krawczyk. "SIGMA: The "SIGn-and-MAC" Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols". In Dan Boneh (ed.), "Advances in Cryptology - CRYPTO 2003", vol. 2729 of *Lecture Notes in Computer Science*, pp. 400–425. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-40674-7. http://dx.doi.org/10.1007/978-3-540-45146-4_24.
9. Ben Laurie and Abe Singer. "Choose the Red Pill and the Blue Pill: A Position Paper". In "Proceedings of the 2008 Workshop on New Security Paradigms", NSPW '08, pp. 127–133. ACM, New York, NY, USA, 2008. ISBN 978-1-60558-341-9. http://doi.acm.org/10.1145/1595676.1595695.
10. M. A. Sasse, S. Brostoff and D. Weirich. "Transforming the 'Weakest Link' — a Human/Computer Interaction Approach to Usable and Effective Security". *BT Technology Journal*, **19**(3):122–131, Jul 2001. ISSN 1358-3948. http://dx.doi.org/10.1023/A:1011902718709.
11. Frank Stajano. "Pico: no more passwords!" In "Proceedings of the 19th international conference on Security Protocols", SP'11, pp. 49–81. Springer-Verlag, Berlin, Heidelberg, 2011. ISBN 978-3-642-25866-4. http://dx.doi.org/10.1007/978-3-642-25867-1_6.
12. Oliver Stannard and Frank Stajano. "Am I in Good Company? A Privacy-Protecting Protocol for Cooperating Ubiquitous Computing Devices". In Bruce Christianson, James Malcolm, Frank Stajano and Jonathan Anderson (eds.), "Security Protocols XX", vol. 7622 of *Lecture Notes in Computer Science*, pp. 223–230. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-35693-3. http://dx.doi.org/10.1007/978-3-642-35694-0_24.
13. Viswanath Venkatesh, Michael G. Morris, Gordon B. Davis and Fred D. Davis. "User Acceptance of Information Technology: Toward a Unified View". *MIS Q.*, **27**(3):425–478, Sep 2003. ISSN 0276-7783. http://dl.acm.org/citation.cfm?id=2017197.2017202.
14. Ford-Long Wong and Frank Stajano. "Multi-channel Protocols". In Christianson et al. (ed.), "Proc. Security Protocols Workshop 2005", vol. 4631 of *LNCS*, pp. 112–127. Springer-Verlag, Apr 2005. ISBN 978-3-540-77155-5. http://www.cl.cam.ac.uk/~fms27/papers/2005-WongSta-multichannel.pdf. Updated version in *IEEE Pervasive Computing* **6**(4):31–39 (2007).