Technical Report

UCAM-CL-TR-817
ISSN 1476-2986

Number 817

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

The quest to replace passwords: a framework for comparative evaluation of Web authentication schemes

Joseph Bonneau, Cormac Herley,
Paul C. van Oorschot, Frank Stajano

March 2012

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

Technical reports published by the University of Cambridge
Computer Laboratory are freely available via the Internet:

http://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes

*

[FULL LENGTH TECHNICAL REPORT]

Joseph Bonneau
University of Cambridge
Cambridge, UK
jcb82@cl.cam.ac.uk

Cormac Herley
Microsoft Research
Redmond, WA, USA
cormac@microsoft.com

Paul C. van Oorschot
Carleton University
Ottawa, ON, Canada
paulv@scs.carleton.ca

Frank Stajano†
University of Cambridge
Cambridge, UK
frank.stajano@cl.cam.ac.uk

***Abstract*—We evaluate two decades of proposals to replace text passwords for general-purpose user authentication on the web using a broad set of twenty-five usability, deployability and security benefits that an ideal scheme might provide. The scope of proposals we survey is also extensive, including password management software, federated login protocols, graphical password schemes, cognitive authentication schemes, one-time passwords, hardware tokens, phone-aided schemes and biometrics. Our comprehensive approach leads to key insights about the difficulty of replacing passwords. Not only does no known scheme come close to providing all desired benefits: none even retains the full set of benefits that legacy passwords already provide. In particular, there is a wide range from schemes offering minor security benefits beyond legacy passwords, to those offering significant security benefits in return for being more costly to deploy or more difficult to use. We conclude that many academic proposals have failed to gain traction because researchers rarely consider a sufficiently wide range of real-world constraints. Beyond our analysis of current schemes, our framework provides an evaluation methodology and benchmark for future web authentication proposals.**
This report is an extended version of the peer-reviewed paper by the same name. In about twice as many pages it gives full ratings for 35 authentication schemes rather than just 9.

***Keywords*-authentication; computer security; human computer interaction; security and usability; deployability; economics; software engineering.**

*A shorter, peer-reviewed version of this report appeared at the 2012 IEEE Symposium on Security and Privacy [1], which should be cited rather than this report wherever appropriate. In addition to the text common to both versions, this report includes, in Section IV, explicit discussion of the ratings of *all* the schemes in Table I.

†Frank Stajano was the lead author who conceived the project and assembled the team. All authors contributed equally thereafter.

CONTENTS

I. Introduction

The continued domination of passwords over all other methods of end-user authentication is a major embarrassment to security researchers. As web technology moves ahead by leaps and bounds in other areas, passwords stubbornly survive and reproduce with every new web site. Extensive discussions of alternative authentication schemes have produced no definitive answers.

Over forty years of research have demonstrated that passwords are plagued by security problems [2] and openly hated by users [3]. We believe that, to make progress, the community must better systematize the knowledge that we have regarding both passwords and their alternatives [4]. However, among other challenges, unbiased evaluation of password replacement schemes is complicated by the diverse interests of various communities. In our experience, security experts focus more on security but less on usability and practical issues related to deployment; biometrics experts focus on analysis of false negatives and naturally-occurring false positives rather than on attacks by an intelligent, adaptive adversary; usability experts tend to be optimistic about security; and originators of a scheme, whatever their background, downplay or ignore benefits that their scheme doesn't attempt to provide, thus overlooking dimensions on which it fares poorly. As proponents assert the superiority of their schemes, their objective functions are often not explicitly stated and differ substantially from those of potential adopters. Targeting different authentication problems using different criteria, some address very specific environments and narrow scenarios; others silently seek generic solutions that fit all environments at once, assuming a single choice is mandatory. As such, consensus is unlikely.

These and other factors have contributed to a long-standing lack of progress on how best to evaluate and compare authentication proposals intended for practical use. In response, we propose a standard benchmark and framework allowing schemes to be rated across a common, broad spectrum of criteria chosen objectively for relevance in wide-ranging scenarios, without hidden agenda.[1] We suggest and define 25 properties framed as a diverse set of benefits, and a methodology for comparative evaluation, demonstrated and tested by rating 35 password-replacement schemes on the same criteria, as summarized in a carefully constructed comparative table.

Both the rating criteria and their definitions were iteratively refined over the evaluation of these schemes. Discussion of evaluation details for passwords and all the examined alternatives is provided herein to demonstrate the process, and to provide evidence that the list of benefits suffices to illuminate the strengths and weaknesses of a wide universe of schemes. Though not cast in stone, we believe that the list of benefits and their specific definitions provide an excellent basis from which to work; the framework and evaluation process that we define are independent of them, although our comparative results naturally are not. From our analysis and comparative summary table, we look for clues to help explain why passwords remain so dominant, despite frequent claims of superior alternatives.

In the past decade our community has recognized a tension between security and usability: it is generally easy to provide more of one by offering less of the other. But the situation is much more complex than simply a linear trade-off: we seek to capture the multi-faceted, rather than one-dimensional, nature of both usability and security in our benefits. We further suggest that "deployability", for lack of a better word, is an important third dimension that deserves consideration. We choose to examine all three explicitly, complementing earlier comparative surveys (e.g., [9]–[11]).

Our usability-deployability-security ("UDS") evaluation framework and process may be referred to as *semi-structured evaluation of user authentication schemes*. We take inspiration from inspection methods for evaluating user interface design, including *feature inspections* and Nielsen's *heuristic analysis* based on usability principles [12].

Each co-author acted as a domain expert, familiar with both the rating framework and a subset of the schemes. For each scheme rated, the evaluation process involved one co-author studying the scheme and rating it on the defined benefits; additional co-authors reviewing each rating score; and iteratively refining the ratings as necessary through discussion, as noted in Section V-D.

Our focus is *user authentication on the web*, specifically from unsupervised end-user client devices (e.g., a personal computer) to remote verifiers. Some schemes examined involve mobile phones as auxiliary devices, but logging in directly from such constrained devices, which involves different usability challenges among other things, is not a main focus. Our present work does not directly examine schemes designed exclusively for machine-to-machine authentication, e.g., cryptographic protocols or infrastructure such as client public-key certificates. Many of the schemes we examine, however, are the technologies proposed for the human-to-machine component that may precede machine-to-machine authentication. Our choice of web authentication as target application also has significant implications for specific schemes, as noted in our results.

II. Benefits

The benefits we consider encompass three categories: usability, deployability and security, the latter including privacy aspects. The benefits in our list have been refined to a set we believe highlights important evaluation dimensions, with an eye to limiting overlap between benefits.

Throughout the paper, for brevity and consistency, each benefit is referred to with an italicized mnemonic title. This title should not be interpreted too literally; refer instead to our actual definitions below, which are informally worded to aid use. Each scheme is rated as either offering or not offering the benefit; if a scheme *almost* offers the benefit, but not quite, we indicate this with the *Quasi-* prefix. Section V-D discusses pros and cons of finer-grained scoring.

[1] The present authors contributed to the definition of the following schemes: URRSA [5], MP-Auth [6], PCCP [7] and Pico [8]. We invite readers to verify that we have rated them impartially.

Sometimes a particular benefit (e.g., *Resilient-to-Theft*) just doesn't apply to a particular scheme (e.g., there is nothing physical to steal in a scheme where the user must memorize a secret squiggle). To simplify analysis, instead of introducing a "not applicable" value, we rate the scheme as offering the benefit—in the sense that nothing can go wrong, for that scheme, with respect to the corresponding problem.

When rating password-related schemes we assume that implementers use best practice such as salting and hashing (even though we know they often don't [13]), because we assess what the scheme's design can potentially offer: a poor implementation could otherwise kill any scheme. On the other hand, we assume that ordinary users won't necessarily follow the often unreasonably inconvenient directives of security engineers, such as never recycling passwords, or using randomly-generated ones.

A. Usability benefits

U1 *Memorywise-Effortless*: Users of the scheme do not have to remember *any* secrets at all. We grant a *Quasi-Memorywise-Effortless* if users have to remember *one* secret for everything (as opposed to one per verifier).

U2 *Scalable-for-Users*: Using the scheme for hundreds of accounts does not increase the burden on the user. As the mnemonic suggests, we mean "scalable" only from the user's perspective, looking at the cognitive load, not from a system deployment perspective, looking at allocation of technical resources.

U3 *Nothing-to-Carry*: Users do not need to carry an additional physical object (electronic device, mechanical key, piece of paper) to use the scheme. *Quasi-Nothing-to-Carry* is awarded if the object is one that they'd carry everywhere all the time anyway, such as their mobile phone, but not if it's their computer (including tablets).

U4 *Physically-Effortless*: The authentication process does not require physical (as opposed to cognitive) user effort beyond, say, pressing a button. Schemes that don't offer this benefit include those that require typing, scribbling or performing a set of motions. We grant *Quasi-Physically-Effortless* if the user's effort is limited to speaking, on the basis that even illiterate people find that natural to do.

U5 *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.

U6 *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.

U7 *Infrequent-Errors*: The task that users must perform to log in usually succeeds when performed by a legitimate and honest user. In other words, the scheme isn't so hard to use or unreliable that genuine users are routinely rejected.[2]

U8 *Easy-Recovery-from-Loss*: A user can conveniently regain the ability to authenticate if the token is lost or the credentials forgotten. This combines usability aspects such as: low latency before restored ability; low user inconvenience in recovery (e.g., no requirement for physically standing in line); and assurance that recovery will be possible, for example via built-in backups or secondary recovery schemes. If recovery requires some form of re-enrollment, this benefit rates its convenience.

B. Deployability benefits

D1 *Accessible*: Users who can use passwords[3] are not prevented from using the scheme by disabilities or other physical (not cognitive) conditions.

D2 *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 scheme is plausible for startups with no per-user revenue.

D3 *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.

D4 *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. In 2012, this would mean an HTML5-compliant browser with JavaScript enabled. Schemes fail to provide this benefit if they require the installation of plugins or any kind of software whose installation requires administrative rights. Schemes offer *Quasi-Browser-Compatible* if they rely on non-standard but very common plugins, e.g., Flash.

D5 *Mature*: The scheme has been implemented and deployed on a large scale for actual authentication purposes beyond research. Indicators to consider for granting the full benefit may also include whether the scheme has undergone user testing, whether the standards community has published related documents, whether open-source projects implementing the scheme exist, whether anyone other than the implementers has

[2]We could view this benefit as "low false reject rate". In many cases the scheme designer could make the false reject rate lower by making the false accept rate higher. If this is taken to an extreme we count it as cheating, and penalize it through a low score in some of the security-related benefits.

[3]Ideally a scheme would be usable by everyone, regardless of disabilities like zero-vision (blindness) or low motor control. However, for any given scheme, it is always possible to identify a disability or physical condition that would exclude a category of people and then no scheme would be granted this benefit. We therefore choose to award the benefit to schemes that do at least as well as the incumbent that is de facto accepted today, despite the fact that it too isn't perfect. An alternative to this text password baseline could be to base the metric on the ability to serve a defined percentage of the population of potential users.

adopted the scheme, the amount of literature on the scheme and so forth.

D6 *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.

C. Security benefits

S1 *Resilient-to-Physical-Observation*: An attacker cannot impersonate a user after observing them authenticate one or more times. We grant *Quasi-Resilient-to-Physical-Observation* if the scheme could be broken only by repeating the observation more than, say, 10–20 times. Attacks include shoulder surfing, filming the keyboard, recording keystroke sounds, or thermal imaging of keypad.

S2 *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.). Personal knowledge questions are the canonical scheme that fails on this point.

S3 *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. The verifier-imposed constraint might be enforced by an online server, a tamper-resistant chip or any other mechanism capable of throttling repeated requests. To give a quantitative example, we might grant this benefit if an attacker constrained to, say, 10 guesses per account per day, could compromise at most 1% of accounts in a year. Lack of this benefit is meant to penalize schemes in which it is frequent for user-chosen secrets to be selected from a small and well-known subset (low min-entropy [14]).

S4 *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. We might for example grant this benefit if an attacker capable of attempting up to 2^{40} or even 2^{64} guesses per account could still only reach fewer than 1% of accounts. Lack of this benefit is meant to penalize schemes where the space of credentials is not large enough to withstand brute force search (including dictionary attacks, rainbow tables and related brute force methods smarter than raw exhaustive search, if credentials are user-chosen secrets).

S5 *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 it is used, perhaps through the CA). As with *Resilient-to-Physical-Observation* above, we grant *Quasi-Resilient-to-Internal-Observation* if the scheme could be broken only by intercepting input or eavesdropping cleartext more than, say, 10–20 times. This penalizes schemes that are not replay-resistant, whether because they send a static response or because their dynamic response countermeasure can be cracked with a few observations. This benefit assumes that general-purpose devices like software-updatable personal computers and mobile phones may contain malware, but that hardware devices dedicated exclusively to the scheme can be made malware-free. We grant *Quasi-Resilient-to-Internal-Observation* to two-factor schemes where both factors must be malware-infected for the attack to work. If infecting only one factor breaks the scheme, we don't grant the benefit.

S6 *Resilient-to-Leaks-from-Other-Verifiers*: Nothing that a verifier could possibly leak can help an attacker impersonate the user to another verifier. This penalizes schemes where insider fraud at one provider, or a successful attack on one backend, endangers the user's accounts at other sites.

S7 *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. This penalizes schemes allowing phishers to get victims to authenticate to lookalike sites and later use the harvested credentials against the genuine sites. It is not meant to penalize schemes vulnerable to more sophisticated real-time man-in-the-middle or relay attacks, in which the attackers have one connection to the victim prover (pretending to be the verifier) and simultaneously another connection to the victim verifier (pretending to be the prover).

S8 *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. We still grant *Quasi-Resilient-to-Theft* if the protection is achieved with the modest strength of a PIN, even if attempts are not rate-controlled, because the attack doesn't easily scale to many victims.

S9 *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.

S10 *Requiring-Explicit-Consent*: The authentication process cannot be started without the explicit consent of the user. This is both a security and a privacy feature (a rogue wireless RFID-based credit card reader embedded in a sofa might charge a card without user knowledge or consent).

S11 *Unlinkable*: Colluding verifiers cannot determine, from the authenticator alone, whether the

same user is authenticating to both. This is a privacy feature. To rate this benefit we disregard linkability introduced by other mechanisms (same user ID, same IP address, etc).

We emphasize that it would be simple-minded to rank competing schemes simply by counting how many benefits each offers. Clearly some benefits deserve more weight than others—but which ones? *Scalable-for-Users*, for example, is a heavy-weight benefit if the goal is to adopt a single scheme as a universal replacement; it is less important if one is seeking a password alternative for only a single account. Providing appropriate weights thus depends strongly on the specific goal for which the schemes are being compared, which is one of the reasons we don't offer any.

Having said that, readers wanting to use weights might use our framework as follows. First, examine and score each individual scheme on each benefit; next, compare (groups of) competing schemes to identify precisely which benefits each offers over the other; finally, with weights that take into account the relative importance of the benefits, determine an overall ranking by rating scheme i as $S_i = \sum_j W_j \cdot b_{i,j}$. Weights W_j are constants across all schemes in a particular comparison exercise, and $b_{i,j} \in [0, 1]$ is the real-valued benefit rating for scheme i on benefit j. For different solution environments (scenarios k), the relative importance of benefits will differ, with weights W_j replaced by $W_j^{(k)}$.

In this paper we choose a more qualitative approach: we do not suggest any weights $W_j^{(k)}$ and the $b_{i,j}$ ratings we assign are not continuous but coarsely quantized. In Section V-D we discuss why. In our experience, *"the journey* (the rating exercise) *is the reward"*: the important technical insights we gained about schemes by discussing whether our ratings were fair and consistent were worth much more to us than the actual scores produced. As a take-home message for the value of this exercise, bringing a team of experts to a shared understanding of the relevant technical issues is much more valuable than ranking the schemes linearly or reaching unanimous agreement over scoring.

III. Evaluating legacy passwords

We expect that the reader is familiar with text passwords and their shortcomings, so evaluating them is good exercise for our framework. It's also useful to have a baseline standard to refer to. While we consider "legacy passwords" as a single scheme, surveys of password deployment on the web have found substantial variation in implemention. A study of 150 sites in 2010 [13], for example, found a unique set of design choices at nearly every site. Other studies have focused on implementations of cookie semantics [15], password composition policies [16], or use of TLS to protect passwords [17]. Every study has found both considerable inconsistency and frequent serious implementation errors in practical deployments on the web.

We remind readers of our Section II assumption of best practice by implementers—thus in our ratings we do not hold against passwords the many weak implementations that their widespread deployment includes, unless due to inherent weaknesses; while on the other hand, our ratings of passwords and other schemes do assume that poor user behavior is an inherent aspect of fielded systems.

The difficulty of guessing passwords was studied over three decades ago [2] with researchers able to guess over 75% of users' passwords; follow-up studies over the years have consistently compromised a substantial fraction of accounts with dictionary attacks. A survey [3] of corporate password users found them flustered by password requirements and coping by writing passwords down on post-it notes. On the web, users are typically overwhelmed by the number of passwords they have registered. One study [18] found most users have many accounts for which they've forgotten their passwords and even accounts they can't remember registering. Another [19] used a browser extension to observe thousands of users' password habits, finding on average 25 accounts and 6 unique passwords per user.

Thus, passwords, as a purely memory-based scheme, clearly aren't *Memorywise-Effortless* or *Scalable-for-Users* as they must be remembered and chosen for each site. While they are *Nothing-to-Carry*, they aren't *Physically-Effortless* as they must be typed. Usability is otherwise good, as passwords are de facto *Easy-to-Learn* due to years of user experience and *Efficient-to-Use* as most users type only a few characters, though typos downgrade passwords to *Quasi-Infrequent-Errors*. Passwords can be easily reset, giving them *Easy-Recovery-from-Loss*.

Their highest scores are in deployability, where they receive full credit for every benefit—in part because many of our criteria are defined based on passwords. For example, passwords are *Accessible* because we defined the benefit with respect to them and accommodations already exist for most groups due to the importance of passwords. Passwords are *Negligible-Cost-per-User* due to their simplicity, and are *Server-Compatible* and *Browser-Compatible* due to their incumbent status. Passwords are *Mature* and *Non-Proprietary*, with turnkey packages implementing password authentication for many popular web development platforms, albeit not well-standardized despite their ubiquity.

Passwords score relatively poorly on security. They aren't *Resilient-to-Physical-Observation* because even if typed quickly they can be automatically recovered from high-quality video of the keyboard [20]. Perhaps generously, we rate passwords as *Quasi-Resilient-to-Targeted-Impersonation* in the absence of user studies establishing acquaintances' ability to guess passwords, though many users undermine this by keeping passwords written down in plain sight [3]. Similarly, users' well-established poor track record in selection means passwords are neither *Resilient-to-Throttled-Guessing* nor *Resilient-to-Unthrottled-Guessing*.

As static tokens, passwords aren't *Resilient-to-Internal-Observation*. The fact that users reuse them across sites means they also aren't *Resilient-to-Leaks-from-Other-Verifiers*, as even a properly salted and strengthened hash function [21] can't protect many passwords from dedicated cracking software. (Up to 50% of websites don't appear to

hash passwords at all [13].) Passwords aren't *Resilient-to-Phishing* as phishing remains an open problem in practice.

Finally, their simplicity facilitates several security benefits. They are *Resilient-to-Theft* as they require no hardware. There is *No-Trusted-Third-Party*; having to type makes them *Requiring-Explicit-Consent*; and, assuming that sites add salt independently, even weak passwords are *Unlinkable*.

IV. Sample evaluation of replacement schemes

We now use our criteria to evaluate a representative sample of proposed password replacement schemes. Table I visually summarizes all the schemes we explored. In the companion peer-reviewed paper [1], due to space constraints, we only explain our ratings in detail for a subset of the schemes. Here, instead, we offer a full write-up for each scheme in the table.

We introduce categories to highlight general trends, but stress that any scheme must be rated individually. Contrary to what the table layout suggests, schemes are not uniquely partitioned by the categories; several schemes belong to multiple categories, and different groupings of the schemes are possible with these same categories. For example, GrIDsure is both cognitive and graphical; and, though several of the schemes we examine use some form of underlying "one-time-passwords", we did not group them into a common category and indeed have no formal category of that name.

We emphasize that, in selecting a particular scheme for inclusion in the table, we do not necessarily endorse it as better than the ones we have not included—merely that it is reasonably representative, or illuminates in some way what the category can achieve.

A. Encrypted Password managers

1) Mozilla Firefox: The Firefox web browser [22] automatically offers to remember passwords entered into web pages, optionally encrypting them with a master password. (Our rating assumes that this option is used; use without the password has different properties.) It then pre-fills the username and password fields when the user revisits the same site. With its Sync facility the passwords can be stored, encrypted, in the cloud. After a once-per-machine authentication ritual, they are updated automatically on all designated machines.

This scheme is *Quasi-Memorywise-Effortless* (because of the master password) and *Scalable-for-Users*: it can remember arbitrarily many passwords. Without Sync, the solution would have required carrying a specific computer; with Sync, the passwords can be accessed from any of the user's computers. However it's not more than *Quasi-Nothing-to-Carry* because a travelling user will have to carry at least a smartphone: it would be quite insecure to sync one's passwords with a browser found in a cybercafé. It is *Quasi-Physically-Effortless*, as no typing is required during authentication except for the master password once per session, and *Easy-to-Learn*. It is *Efficient-to-Use* (much more so than what it replaces) and has *Infrequent-Errors* (hardly any, except when entering the master password). It does not have *Easy-Recovery-from-Loss*: losing the master password is catastrophic.

The scheme is backwards-compatible by design and thus scores quite highly on deployability: it fully provides all the deployability benefits except for *Browser-Compatible*, unavoidably because it requires a specific browser.

It is *Quasi-Resilient-to-Physical-Observation* and *Quasi-Resilient-to-Targeted-Impersonation* because an attacker could still target the infrequently-typed master password (but would also need access to the browser). It is not *Resilient-to-Throttled-Guessing* nor *Resilient-to-Unthrottled-Guessing*: even if the master password is safe from such attacks, the original web passwords remain as vulnerable as before.[4] It is not *Resilient-to-Internal-Observation* because, even if TLS is used, it's replayable static passwords that flow in the tunnel and malware could also capture the master password. It's not *Resilient-to-Leaks-from-Other-Verifiers*, because what happens at the back-end is the same as with passwords. It's *Resilient-to-Phishing* because we assume that sites follow best practice, which includes using TLS for the login page. It is *Resilient-to-Theft*, at least under our assumption that a master password is being used. It offers *No-Trusted-Third-Party* because the Sync data is pre-encrypted locally before being stored on Mozilla's servers. It offers *Requiring-Explicit-Consent* because it pre-fills the username and password fields but the user still has to press enter to submit. Finally, it is as *Unlinkable* as passwords.

2) LastPass: LastPass [23] is a commercial and proprietary password manager that integrates with a variety of web browsers (through plug-ins) and provides cloud storage and syncing of encrypted passwords. Saved passwords are protected by a master password, as with Firefox, but LastPass also allows cross-browser syncing, even with browsers on smartphones. The program also generates strong passwords.

In May 2011, anomalous traffic was detected on the LastPass servers, prompting the suspicion that the database of salted encrypted passwords was being downloaded [24]. The administrators promptly rebuilt the database and asked users to change their passwords. No evidence was provided either way as to whether a leak effectively occurred or not.

Since it requires only one master password, this scheme is *Quasi-Memorywise-Effortless* and *Scalable-for-Users*. It has *Quasi-Nothing-to-Carry* (the user may sync the passwords to any of her devices but, if she is mobile, she must still carry one of them around). It is *Quasi-Physically-Effortless* because, other than the master password once per session, no typing of passwords is required. It's *Easy-to-Learn*, *Efficient-to-Use* and has *Infrequent-Errors*, as the program enters the passwords on behalf of the user. Losing the master password is irreversible but there is an

[4] Security-conscious users might adopt truly random unguessable passwords, as they need no longer remember them, but most users won't. If the scheme pre-generated random passwords it would score more highly here, disregarding pre-existing passwords. Similarly, for *Resilient-to-Leaks-from-Other-Verifiers* below, this scheme makes it easier for careful users to use a different password for every site; if it forced this behaviour (vs. just allowing it), it would get a higher score on this particular benefit.

account recovery facility based on a locally stored one-time password: we rate this *Quasi-Easy-Recovery-from-Loss*.

LastPass is as *Accessible* as passwords. It has *Quasi-Negligible-Cost-per-User* (both free and paid subscription versions exist). It's *Server-Compatible* (no changes required) but not *Browser-Compatible* (plugin required). It's *Mature* (widely deployed) but not *Non-Proprietary* (closed source).

It's *Quasi-Resilient-to-Physical-Observation* and *Quasi-Resilient-to-Targeted-Impersonation* because an attacker could still target the master password. We rate it *Quasi-Resilient-to-Throttled-Guessing* and *Quasi-Resilient-to-Unthrottled-Guessing*: to get the full benefit, users would have always to let the program generate random passwords on their behalf, which the program allows but does not enforce. It's not *Resilient-to-Internal-Observation*: the static passwords sent to web sites could be observed. It's only *Quasi-Resilient-to-Leaks-from-Other-Verifiers* as users might keep their old passwords rather than using the program's strong random ones. It's *Resilient-to-Phishing* because in our threat model we assume, as we did with Firefox in Section IV-A1, that web sites follow best practice, which includes using TLS for the login page. This allows the LastPass client to refuse to send a password to a phishing website which won't have the correct certificate for the domain it is imitating. It's *Resilient-to-Theft* thanks to the master password. It's unclear whether LastPass Inc should be considered a TTP: closed source means they might leak your master password without anyone knowing; therefore, erring on the side of caution, and considering the 2011 incident, we rate the scheme as not *No-Trusted-Third-Party*. It's *Requiring-Explicit-Consent* (the user must press Enter to send the password) and can't be worse than passwords on *Unlinkable* (besides the added bonus that randomly generated passwords will be unrelated).

B. Proxy-based

1) URRSA: Proxy-based schemes place a man-in-the-middle between the user's machine and the server. One reason for doing so, employed by Impostor [25] and URRSA [5] is to enable secure logins despite malware-infected clients.

URRSA has users authenticate to the end server using one-time codes carried on a sheet of paper. At registration the user enters the password, P_j, for each account, j, to be visited; this is encrypted at the proxy with thirty different keys, K_i, giving $C_i = E_{K_i}(P_j)$. The C_i act as one-time codes which the user prints and carries. The codes are generally 8-10 characters long; thirty codes for each of six accounts fit on a two-sided sheet. The keys, but not the passwords, are stored at the proxy. At login the user visits the proxy, indicates which site is desired, and is asked for the next unused code. When he enters the code it is decrypted and passed to the end login server: $E^{-1}_{K_i}(C_i) = P_j$. The proxy never authenticates the user, it merely decrypts with an agreed-upon key, the code delivered by the user.

Since it requires carrying one-time codes URRSA is *Memorywise-Effortless*, but not *Scalable-for-Users* or *Nothing-to-Carry*. It is not *Physically-Effortless* but is *Easy-to-Learn*. In common with all of the schemes that involve transcribing codes from a device or sheet it is not *Efficient-to-Use*. However, we do consider it to have *Quasi-Infrequent-Errors*, since the codes are generally 8-10 characters. It does not have *Easy-Recovery-from-Loss*: a revocation procedure is required if the code sheet is lost or stolen. Since no passwords are stored at the proxy the entire registration must be repeated if this happens.

In common with other paper token schemes it is not *Accessible*. URRSA has *Negligible-Cost-per-User*. Rather than have a user change browser settings, URRSA relies on a link-translating proxy that intermediates traffic between the user and the server; this translation is not flawless and some functionality may fail on complex sites, thus we consider it only *Quasi-Server-Compatible*. It is, however, *Browser-Compatible*. It is neither *Mature* nor *Non-Proprietary*.

In common with other one-time code schemes it is not *Resilient-to-Physical-Observation*, since a camera might capture all of the codes on the sheet. Since it merely inserts a proxy it inherits many security weaknesses from the legacy password system it serves: it is *Quasi-Resilient-to-Targeted-Impersonation* and is not *Resilient-to-Throttled-Guessing* or *Resilient-to-Unthrottled-Guessing*. It is *Quasi-Resilient-to-Internal-Observation* as observing the client during authentication does not allow passwords to be captured, but breaking the proxy-to-server TLS connection does. It inherits from passwords the fact that it is not *Resilient-to-Leaks-from-Other-Verifiers*, but the fact that it is *Resilient-to-Phishing* from other one-time schemes. It is not *Resilient-to-Theft* nor *No-Trusted-Third-Party*: the proxy must be trusted. It offers *Requiring-Explicit-Consent* and is *Unlinkable*.

2) Impostor: Impostor is a proxy-based single sign-on system to enable logins from malware-infected clients proposed by Pashalidis and Mitchell [25]. The proxy intermediates all traffic between the client and the end-server.

At registration the user enters passwords for the accounts to be visited which are stored at the proxy. The user also establishes a phrase which will act as the shared secret to authenticate access to the proxy. To access accounts from a potentially malware-infected client, the user must change the proxy settings on the browser to direct traffic to the Impostor proxy. Before allowing access the proxy challenges the user for a randomly selected subset of characters from the secret phrase (e.g., characters 7, 19 and 34 of a 50-character phrase). This enables the user to login several times from the same machine while making replay attacks hard.

Impostor requires that the user either memorize or carry the secret phrase. We assume the user memorizes the secret phrase which then works for all verifiers meaning Impostor is *Quasi-Memorywise-Effortless* by our definition, as well as *Nothing-to-Carry* and *Scalable-for-Users*. In common with all of the schemes that involve transcribing codes from a device or sheet it is *Easy-to-Learn*, but not *Efficient-to-Use*: looking up and entering challenge characters takes significantly longer than entering a memorized password. We consider it not to have *Infrequent-Errors*:

the user must accurately calculate character positions on a memorized phrase. It is *Easy-Recovery-from-Loss*: a new shared secret phrase can be established if the old one is forgotten.

We consider the scheme *Accessible* as few users should be excluded from using it. Impostor has *Negligible-Cost-per-User*. Impostor is *Server-Compatible*: there is never an issue of functionality failing as occasionally happens with URRSA. However, it is only *Quasi-Browser-Compatible*: users must change proxy settings, a task that they may not have permissions to do on the client, which thus fails to satisfy our definition. It is not *Mature* but it is *Non-Proprietary*.

It is *Resilient-to-Physical-Observation* if the secret is not written down. Impostor achieves the fact that it is *Server-Compatible* by leaving the existing password system unchanged: passwords still work at the legacy server. Thus it shares the security weaknesses of attempts on the server: it is *Quasi-Resilient-to-Targeted-Impersonation* and is not *Resilient-to-Unthrottled-Guessing* or *Resilient-to-Throttled-Guessing*. For example, and attacker can entirely ignore the Impostor proxy and achieve exactly the same success with a throttled or unthrottled guessing attack as he would against any password server.

The system is *Quasi-Resilient-to-Internal-Observation* as nothing the attacker can gain by observing the client during authentication will allow the passwords to be captured, but breaking the proxy-to-server connection will. Impostor inherits the fact that it is *Resilient-to-Phishing* from other password managers. It clearly is *Resilient-to-Theft* and obviously not *No-Trusted-Third-Party*. It inherits from passwords the fact that it is not *Resilient-to-Leaks-from-Other-Verifiers*, but is *Unlinkable*.

C. Federated Single Sign-On

1) OpenID: Federated single sign-on enables web sites to authenticate a user by redirecting them to a trusted identity server which attests the users' identity. This has been considered a "holy grail" as it could eliminate the problem of remembering different passwords for different sites. The concept of federated authentication dates at least to the 1978 Needham-Schroeder key agreement protocol [26] which formed the basis for Kerberos [27]. Kerberos has inspired dozens of proposals for federated authentication on the Internet; Pashalidis and Mitchell provided a complete survey [28]. A well-known representative is OpenID,[5] a protocol which allows any web server to act as an "identity provider" [29] to any server desiring authentication (a "relying party"). OpenID has an enthusiastic group of followers both in and out of academia, but it has seen only patchy adoption with many sites willing to act as identity providers but few willing to accept it as relying parties [30].

In evaluating OpenID, we note that in practice identity providers will continue to use text passwords to authenticate users in the forseeable future, although the protocol itself allows passwords to be replaced by a stronger mechanism. Thus, we rate the scheme *Quasi-Memorywise-Effortless* in that most users will still have to remember one master password, but *Scalable-for-Users* as this password can work for multiple sites. OpenID is *Nothing-to-Carry* like passwords and *Quasi-Physically-Effortless* because passwords only need to be typed at the identity provider. Similarly, we rate it *Efficient-to-Use* and *Infrequent-Errors* in that it is either a password authentication or can occur automatically in a browser with cached login cookies for the identity provider. However, OpenID has found that selecting an opaque "identity URL" can be a significant usability challenge without a good interface at the relying party, making the scheme only *Quasi-Easy-to-Learn*. OpenID is *Easy-Recovery-from-Loss*, equivalent to a password reset.

OpenID is favorable from a deployment standpoint, providing all benefits except for *Server-Compatible*, including *Mature* as it has detailed standards and many open-source implementations. We do note however that it requires identity providers yield some control over trust decisions and possibly weaken their own brand [30], a deployment drawback not currently captured in our criteria.

Security-wise, OpenID reduces most attacks to only the password authentication between a user and his or her identity provider. This makes it somewhat difficult to rate; we consider it *Quasi-Resilient-to-Throttled-Guessing*, *Quasi-Resilient-to-Unthrottled-Guessing*, *Quasi-Resilient-to-Targeted-Impersonation*, *Quasi-Resilient-to-Physical-Observation* as these attacks are possible but only against the single identity provider (typically cached in a cookie) and not for each login to all verifiers. However, it is not *Resilient-to-Internal-Observation* as malware can either steal persistent login cookies or record the master password. OpenID is also believed to be badly non-*Resilient-to-Phishing* since it involves re-direction to an identity provider from a relying party [31]. OpenID is *Resilient-to-Leaks-from-Other-Verifiers*, as relying parties don't store users passwords. Federated schemes have been criticized on privacy grounds and, while OpenID does enable technically savvy users to operate their own identity provider, we rate OpenID as non-*Unlinkable* and non-*No-Trusted-Third-Party* as the vast majority of users aren't capable of doing so.

2) Microsoft Passport: Microsoft Passport was a prominent early proposal for web single-sign on in 1997 [32]. Passport was a close analog to Kerberos for the web, with Microsoft running the only trusted authentication servers and relying parties paying an annual licensing fee to utilize the service. Many academics voiced criticism of Passport on privacy and openness grounds [33]; there were also several bugs found in the protocol [34] and the scheme required all relying servers to pay expensive license fees. As a result, Passport saw little deployment and the system has been converted into a standard OpenID identity provider.

The usability of Passport is identical to OpenID, with the positive exception of being *Easy-to-Learn* as the complicated step of OpenID, choosing one's identity provider, can be eliminated as Microsoft was the only available option. Deployability is worse than OpenID (or text

[5]OpenID is often confused with OAuth, a technically unrelated protocol for delegating access to one's accounts to third parties. The recent OpenID Connect proposal merges the two. We consider the OpenID 2.0 standard here, though all current versions score identically in our framework.

passwords) in that Passport is *Browser-Compatible* and *Accessible* but not *Server-Compatible*. Passport's licensing fees also make the scheme *Negligible-Cost-per-User* nor *Non-Proprietary*.

Finally, the security of Passport is, like OpenID, largely based on the security of a single password. Thus Passport is *Quasi-Resilient-to-Physical-Observation*, *Quasi-Resilient-to-Targeted-Impersonation*, *Quasi-Resilient-to-Throttled-Guessing*, *Quasi-Resilient-to-Unthrottled-Guessing*, non-*Resilient-to-Internal-Observation* and non-*Resilient-to-Phishing* but *Resilient-to-Leaks-from-Other-Verifiers*. Passport is strongly not *No-Trusted-Third-Party* nor *Unlinkable* as all users needed to rely on Microsoft.

3) Facebook Connect: Facebook Connect [35], launched in 2008, is a single sign-on scheme with Facebook as the only identity provider (similar to Passport in that regard). To the user, the scheme looks very similar to OpenID or Passport, with login requiring a redirection to Facebook and typing credentials there (or clicking a button if the user is already logged in). Under the hood it is loosely based on OAuth [36], which enables relying parties to read or write user data after receiveing permission which is a major added benefit for relying parties who can gain access to a user's social data. It is free for relying parties. It has grown quickly and in 2010 it was found to be accepted by more relying parties than OpenID [13].

Facebook Connect scores similarly to OpenID and almost identically to Microsoft Passport, also being *Easy-to-Learn* by requiring just a single button click and no selection of an identity provider. It has a slight advantage from a deployability standpoint as it is *Negligible-Cost-per-User* though still not *Non-Proprietary*.

Finally, the security of Facebook Connect is also *Quasi-Resilient-to-Physical-Observation*, *Quasi-Resilient-to-Targeted-Impersonation*, *Quasi-Resilient-to-Throttled-Guessing*, *Quasi-Resilient-to-Unthrottled-Guessing*, non-*Resilient-to-Internal-Observation* and non-*Resilient-to-Phishing* but *Resilient-to-Leaks-from-Other-Verifiers* by reducing security to a single trusted password. Like Passport it is strongly not *No-Trusted-Third-Party* nor *Unlinkable* as all users need to rely on Facebook and can't run their own servers. Facebook Connect also isn't *Requiring-Explicit-Consent*, as users can be logged in automatically when logged into Facebook. It may have further privacy drawbacks not captured in our criteria as the protocol is designed to enable users to share their social networking data from Facebook to relying websites.

4) Mozilla BrowserID: A very recent competing proposal is the Mozilla Foundation's BrowserID [37], previously called Verified Email, to be built into future releases of the Mozilla browser. In contrast to OpenID, which uses opaque URLs as identifiers, BrowserID uses email addresses. Using email addresses as identifiers has been suggested to be advantageous to OpenID-style identify URLs because email addresses are global identifiers that users are already familiar with [38]. It is designed to be browser-centric in that identity providers can provide a signed, time-limited certificate asserting that "this user owns email address x" that the browser caches. The browser then controls authentication using this certificate. Users can request very short-term assertion certificates for browsers they don't frequently use. The scheme is also designed to be backwards-compatible with non-supporting browsers, albeit through running an online delegated protocol requiring Mozilla to act as a trusted third party.

From the user's perspective, the process is similar to OpenID, as they must initially be redirected to their identity provider for a (presumably) password-based authentication before automatic authentication to relying parties (with a dialog box to ensure consent). Thus, the scheme is also *Quasi-Memorywise-Effortless* in that most users will still have to remember one master password, but *Scalable-for-Users*. BrowserID is also *Nothing-to-Carry* like passwords and *Quasi-Physically-Effortless* because passwords only need to be typed at the identity provider as well as *Efficient-to-Use* and *Infrequent-Errors* in that the scheme typically happens with only a single mouse click. Usability has not been studied in practice, but the scheme is designed to be *Easy-to-Learn* as users must simply supply their email address, which they often do already for registration. It is also *Easy-Recovery-from-Loss* as recovery is equivalent to normal password reset.

From a deployment standpoint, BrowserID is comparable to OpenID except not currently *Browser-Compatible*. Because there are plans to include browser support in the future, and a backward compatibility plan in place currently, we rate it as *Quasi-Browser-Compatible*. Similarly, we rate the scheme as *Quasi-Mature* in that, while not widely deployed yet, it is based on open standards and being supported by the Mozilla Foundation which has credibility as a large developer.

Security-wise, BrowserID is identical to OpenID, being *Quasi-Resilient-to-Throttled-Guessing*, *Quasi-Resilient-to-Unthrottled-Guessing*, *Quasi-Resilient-to-Targeted-Impersonation*, *Quasi-Resilient-to-Physical-Observation* in reducing security to a single password, though not *Resilient-to-Internal-Observation* as malware can steal the master password or an identity certificate. We still consider BrowserID to be not *Resilient-to-Phishing* even though the authentication process to the server can be protected with special browser chrome because this approach has not proven effective against traditional phishing. Finally, while we rate BrowserID as non-*No-Trusted-Third-Party* because most users will not run their own identity server and rely on a third-party which can then impersonate them to any website. However, it does have a tangible privacy advantage over OpenID in that the user's identity server is not made aware of every authentication request; in the common case they are handled by the browser using an identity certificate and the identity provider doesn't learn which relying party the user is authenticating to. Still, BrowserID is not *Unlinkable* by our definition because different verifiers can easily check if the same user (represented by an email address) is authenticating to both.

5) SAW (one-time passwords over email): Federated authentication can be bootstrapped using the existing SMTP email system. Email providers effectively attest to a user's ownership of a specific email address within their domain by enabling the user to retrieve incoming mail

at this address. If a server sends a one-time token to an email address and a user is able to retrieve it they are thus completing a federated authentication with the email provider acting as an identity provider. This protocol has arisen organically since the early days of the web as a recovery scheme when passwords are forgotten. Bonneau and Preibusch [13] found in 2010 that 92% of web sites surveyed used email for password reset. A 2009 user study by Karlof et al. [39] strongly favored the use of email-based reset over personal knowledge questions.

Garfinkel [40] first analyzed the possibility of using one-time tokens sent over email as a primary authentication means in 2003, conjecturing that it had many usability advantages over other attempts to establish PKI. As noted for BrowserID,[6] email addresses are ideal identifiers for federated authentication that users are already familiar with [38].

Van der Horst and Seamons [41] proposed "Simple Authentication for the Web" (SAW) as a specific protocol for email-based login. In SAW, users enters their username to initiate authentication. They then receive an email with a link to complete the authentication, combining a secret token transmitted over SMTP with one stored in JavaScript to bind the browser session requesting identification with the one completing it. SAW's authors also propose an optional browser extension to automate authentication, but we rate the basic version.

Like other federated schemes on the web, an email-based approach is *Quasi-Memorywise-Effortless* in practice as it requires only remembering a single password for one's webmail account and inherently *Scalable-for-Users* as one email address can be the credential for arbitrarily many web accounts. However, SAW is not *Physically-Effortless*, and furthermore is non-*Efficient-to-Use* as it involves the latency of an email as well as switching browser tabs and using copy/paste or clicking a link (though it is *Infrequent-Errors* as nothing needs to be typed.) SAW's authors suggested that browser automation for their protocol would be possible, but the latency of SMTP fundamentally prevents the scheme from being efficient for users. We consider SAW to be *Easy-to-Learn*, as it is equivalent for the user to common web re-authentication over email. It is also *Easy-Recovery-from-Loss*, as it is equivalent to a basic password reset.

Deploying a scheme like SAW is relatively straightforward because it relies mostly on existing web infrastructure, making it *Accessible*, *Browser-Compatible* and *Negligible-Cost-per-User*, though not *Server-Compatible* as servers must send out one-time email tokens. While the concept of using emails for authentication is old, SAW isn't *Mature* as it remains an academic proposal, though it is *Non-Proprietary*.

Security-wise, SAW authentication itself is strong as only one-time tokens are manipulated by the user. However, like other federated schemes it relies on password authentication with the identity provider and thus we can rate it only as *Quasi-Resilient-to-Physical-Observation*, *Quasi-Resilient-to-Targeted-Impersonation*, *Quasi-Resilient-to-Unthrottled-Guessing* and *Quasi-Resilient-to-Throttled-Guessing* because attacks against the relying party aren't possible, but passwords can still be attacked at the user's email provider. Also similar to other federated schemes, we rate it as non-*Resilient-to-Internal-Observation* because malware can easily log a user's main email password. In contrast to OpenID, SAW is *Resilient-to-Phishing* as users are not redirected to their email provider, they are expected to already have it open in another window and simply told to go to it. However, email-based schemes cannot provide *No-Trusted-Third-Party* as most users are incapable of running a mail server. Similarly, while it is technically possible for users to set up email aliases, we consider this impossible for most users so the scheme is non-*Unlinkable*.

[6]Note that while the BrowserID scheme was previously called Verified Email, it does not actually test users's ability to receive email using SMTP, it only provides assertions of owning an email address.

D. Graphical passwords

1) Persuasive Cued Clickpoints (PCCP): Graphical passwords schemes attempt to leverage natural human ability to remember images, which is believed to exceed memory for text. We consider as a representative PCCP [7] (Persuasive Cued Click-Points), a cued-recall scheme. Users are sequentially presented with five images on each of which they select one point, determining the next image displayed. To log in, all selected points must be correctly re-entered within a defined tolerance. To flatten the password distribution, during password creation a randomly-positioned portal covers a portion of each image; users must select their point from therein (the rest of each image is shaded slightly). Users may hit a "shuffle" button to randomly reposition the portal to a different region—but doing so consumes time, thus persuading otherwise. The portal is absent on regular login. Published security analysis and testing report reasonable usability and improved security over earlier schemes, specifically in terms of resistance to both *hotspots* and *pattern-based* attacks [11].

While not *Memorywise-Effortless*, nor *Scalable-for-Users* due to extra cognitive load for each account password, PCCP offers advantages over text passwords (and other uncued schemes) due to per-account image cues reducing password interference. It is *Easy-to-Learn* (usage and mental models match web passwords, but interface details differ), but only *Quasi-Efficient-to-Use* (login times on the order of 5s to 20s exceed text passwords) and at best *Quasi-Infrequent-Errors*.

PCCP is not *Accessible* (consider blind users) and has *Negligible-Cost-per-User*. It is not *Server-Compatible*; though it might be made so by having a proxy act as intermediary (much as URRSA does). It is *Browser-Compatible*. It is not *Mature*, but apparently *Non-Proprietary*.

PCCP is not *Resilient-to-Physical-Observation* (due to video-camera shoulder surfing), but is *Resilient-to-Targeted-Impersonation* (personal knowledge of a target user does not help attacks). We rate it *Quasi-Resilient-to-Throttled-Guessing* due to portal persuasion increasing password randomness, but note individual users may repeatedly bypass portal recommendations. Although the persuasion is also intended to mitigate offline attacks,

we rate it not *Resilient-to-Unthrottled-Guessing* as studies to date have been limited to full password spaces of 2^{43} (which are within reach of offline dictionary attack, especially for users choosing more predictable passwords, assuming verifier-stored hashes are available). It is not *Resilient-to-Internal-Observation* (static passwords are replayable). It is *Resilient-to-Leaks-from-Other-Verifiers* (distinct sites can insist on distinct image sets). PCCP is *Resilient-to-Phishing* per our strict definition of that benefit; to obtain the proper per-user images, a phishing site must interact (e.g., by MITM) with a legitimate server. PCCP matches text passwords on being *Unlinkable*.

2) PassGo: Graphical passwords involve images of some form. The memory recall modes leveraged suggest a natural categorization as pure recall, cued-recall and recognition-based schemes [11]. Current mainstream use, e.g., in Android smartphones [42] and a grid-pattern scheme in Windows 8,[7] is more for access control than remote login. PCCP, discussed above, is a cued-recall scheme.

PassGo [43] is a pure recall scheme. In an example implementation, given a 9×9 grid of dots, a mouse, stylus or finger is used to enter password "doodles" as a sequence of strokes (each a dot sequence along one direction) separated by "pen-ups". Each stroke of one or more dots includes all dots from its start to end. A password is the concatenation of encoded strokes $xyzs0$ indicating start point (x, y) continuing s additional dots in direction $1 \leq z \leq 8$ (left, right, up, down plus diagonals) with 0 indicating pen-up. Users initially create such a password and reconstruct it to log in. PassGo may be viewed [44] as a discretized version of the DAS (Draw-A-Secret) scheme [45]. The Android 9-dot screen-lock pattern [42] is a scaled-down variant providing PIN-level security. Detailed security analyses have been done on DAS and PassGo, plus a user study on the latter. The main security weakness found to date exploits predictability in user choice; user-chosen passwords are more predictable than those randomly chosen from the theoretical password space, a situation analogous to text passwords.

PassGo scores similarly to PCCP on the rated usability and deployability benefits, though we rate it *Quasi-Mature* (ahead of PCCP's not-*Mature*) on the basis of the aforementioned Android phone version (scaled down to 3×3 grid). Neither is *Scalable-for-Users*; but PassGo fares slightly worse, having no per-account image cues, and has disadvantages on the security benefits. Attackers may exploit PassGo's greater password predictability due to user choice, making it not *Resilient-to-Throttled-Guessing* and enabling dictionary attacks, making it similarly not *Resilient-to-Unthrottled-Guessing*. It is not *Resilient-to-Leaks-from-Other-Verifiers* since stored verification hashes leaked from one site may compromise passwords reused at other sites. As a (pure) recall scheme, PassGo is not *Resilient-to-Phishing*; phishing sites can present all users with the same blank PassGo grid.

The discussion above, as summarized in our coarse ratings table, fails to capture all relevant benefits and aspects, e.g., the convenience and suitability of PassGo for touch-screen mobile phones and tablets. However, for the 25 benefits in focus, PassGo in comparison to web passwords rates one green (better) vs. four red (worse) cells, facing challenges in Deployability while offering fewer Security advantages than most competing alternatives to passwords.

3) PassFaces: PassFaces [11][8] is a well-known recognition-based graphical scheme. Users are presented with, e.g., four panels of 9 faces each, having 8 decoys plus one face they must select as a portfolio face designated in a password creation phase. This offers security best compared to PIN-level schemes (cf. Android screen-lock, above); increasing parameters such as the number of panels and/or faces per panel implies greater login time and increased cognitive burden. An early version of PassFaces allowing user selection of the portfolio faces had severe security weaknesses due to the predictability of user choices, as found by the Davis et al. [46] study of a version called "Face"; thus more recent versions use system-assigned faces. While PassFaces has been commercially promoted, disadvantages compared to PassGo and PCCP include that the most common PIN-level versions are neither *Resilient-to-Unthrottled-Guessing* nor *Resilient-to-Throttled-Guessing* due to the small password space. PassFaces implementations can be *Resilient-to-Leaks-from-Other-Verifiers* by using different image datasets across sites. Hlywa et al. [47] recently found no evidence that the use of faces provides user performance advantages over object images; indeed, related schemes use objects other than faces, e.g., GPI/GPIS [48] (Graphical Password with Icons/with Icons suggested by System) has users select 6 ordered icons from a panel of 150 (for 2^{43} possible passwords), while Déjà Vu [49] has users recognize 5 random art images from a panel of 25 (for 2^{16} possible passwords). Dunphy et al. [50] explore other recognition-based schemes tailored for smartphones.

In summary comparison of graphical passwords to text passwords, they fall within current password verification frameworks and usage models, albeit less mature and requiring server-side changes. If different servers use different images as is possible in cued-recall and recognition-based schemes, they may offer advantages of slightly better scalability (in terms of reduced password interference), leak-resilience and resilience to phishing; and selected schemes may offer reduced susceptibility to targeted impersonation and throttled-guessing attacks. However among many characteristics shared by graphical and text passwords are two disadvantages: replayability inherent in static passwords, and the requirement to memorize secrets, in the worst case different for each account—and thus graphical passwords remain challenged to scale to hundreds of accounts absent helper mechanisms such as password managers.

E. Cognitive authentication

1) GrIDsure: Challenge-Response schemes attempt to address the replay attack on passwords by having the user deliver proof that he knows the secret without divulging

[7] http://www.forumswindows8.com/info/graphical-password-logon-windows-8-205.html.

[8] Table I contains no row for this scheme, despite its discussion here.

the secret itself. If memorization and computation were no barrier then the server might challenge the user to return a cryptographic hash of the user's secret combined with a server-selected nonce. However, it is unclear if a scheme within the means of human memory and calculating ability is achievable. We examine the commercial offering GrIDsure (a variant of which is described in a paper [51] by other authors) as representative of the class.

At registration the user is presented with a grid (e.g., 5×5) and selects a pattern, or sequence of cells. There are 25^4 possible length-4 patterns, for example. At login the user is again presented with the grid, but now populated with digits. To authenticate he transcribes the digits in the cells corresponding to his pattern. Since the association of digits to cells is randomized the string typed by the user is different from login to login. Thus he reveals knowledge of his secret without typing the secret itself.

This scheme is similar to passwords in terms of usability and we (perhaps generously) rate it identically in terms of many usability benefits. An exception is that it's only *Quasi-Efficient-to-Use*: unlike passwords, which can often be typed from muscle memory, transcribing digits from the grid cells requires effort and attention and is likely to be slower.

We consider the scheme as not *Accessible* as the two-dimensional layout seems unusable for blind users. The scheme has *Negligible-Cost-per-User*, in terms of technology. It is not *Server-Compatible* but is *Browser-Compatible*. It is not *Mature*. We rate it not *Non-Proprietary*, as the intellectual property status is unknown.

The security properties are, again, similar to passwords in many respects. It is not *Resilient-to-Physical-Observation*, as a camera that captures both the grid and user input quickly learns the secret. It is an improvement on passwords in that it is *Resilient-to-Targeted-Impersonation*: we assume that an attacker is more likely to guess secret strings than secret patterns based on knowledge of the user. However, its small space of choices prevents it from being *Resilient-to-Throttled-Guessing* or *Resilient-to-Unthrottled-Guessing*. In spite of the one-time nature of what the user types the scheme is not *Resilient-to-Internal-Observation*: too many possible patterns are eliminated at each login for the secret to withstand more than three or four observations. It shares the remaining security benefits with passwords.

2) Weinshall: Weinshall [52] proposed a challenge-response scheme. The original claim that the scheme would withstand observation attacks was refuted by Golle and Wagner [53], who showed that as few as six or seven observations would suffice for an attacker to learn the whole secret.

The scheme requires a user to memorize a set of thirty assigned images. To login the user is presented with a series of screens. Each screen contains a grid of 8×10 images, some of which are the images previously memorized. Starting at the top-left corner the user calculates a path that advances down and to the right, but varies when it encounters one of the assigned images. On reaching the edge of the screen the user enters the two bit number written on the margin at the exit point. This procedure is repeated eleven times per login.

The scheme is clearly not *Memorywise-Effortless* or *Scalable-for-Users*. It is *Nothing-to-Carry*, but is not *Physically-Effortless*. Not only is it not *Easy-to-Learn*, *Efficient-to-Use* or *Infrequent-Errors* it is so difficult to use along each of those criteria that any one of them might be considered a fatal weakness. Login times of users in trials took about three minutes, and errors are likely to be very frequent for such a complex and demanding scheme. It does not have *Easy-Recovery-from-Loss* as the effort of memorizing the assigned images must be repeated if, for any reason, the original secret images are compromised.

The scheme is not *Accessible*: a successful authentication requires viewing the graphical layout of the secret images which isn't possible for visually impaired users. As a memory-only scheme it has *Negligible-Cost-per-User*. It is not *Server-Compatible* or *Mature* but is *Browser-Compatible* and *Non-Proprietary*.

The scheme is *Quasi-Resilient-to-Physical-Observation*: although 6-7 observations suffice to learn the secret, these must be perfect and unobstructed. It seems hard to get a large number of such observations. The scheme is *Resilient-to-Targeted-Impersonation*: the images are assigned, so knowledge of the user does not help in guessing the shared secret. The scheme is not *Resilient-to-Throttled-Guessing* or *Resilient-to-Unthrottled-Guessing*: the set of images memorized by user is large enough to provide only a PIN-level secret. As shown by Golle and Wagner [53], the scheme is not *Resilient-to-Internal-Observation*. The scheme is *Resilient-to-Phishing*: a malicious site that lures the user into authenticating learns the secret only if the user can be convinced to authenticate 6-7 times. Given the three minutes a successful authentication takes, this seems improbable. The scheme is *Resilient-to-Theft*: there's nothing to steal. It is *No-Trusted-Third-Party*, *Requiring-Explicit-Consent*, and *Unlinkable*.

3) Hopper-Blum: Hopper and Blum [54] proposed a challenge-response scheme that involves user calculation on a shared secret. The scheme requires that user and server share an N-bit secret. They recommend on the order of $N = 120$ but the secret is non-zero in only 20 or so locations. When the user wishes to authenticate he is sent an N-bit challenge and must calculate the 1-bit inner product between secret and challenge. Rather than faithfully return the calculated bit, the user returns the correct answer only with probability $1 - \nu$ for some small ν (also shared with the server). To give even PIN-level (e.g., 20-bit) security this must be repeated about 20 times. Hopper and Blum explicitly state that their protocol is not a practical solution to the problem, and acknowledge that the protocol and memorization are unreasonable for humans. The scheme is similar to the Weinshall [52] in many ways and shares many security and usability properties.

The scheme is not *Memorywise-Effortless* nor *Scalable-for-Users*. It is *Nothing-to-Carry*, but not *Physically-Effortless*. As we said for Weinshall above: not only is it not *Easy-to-Learn*, *Efficient-to-Use* or *Infrequent-Errors*, but it is so difficult to use that its performance on any one of those criteria might be considered a fatal weakness. Login times will be long, and errors are likely to be very frequent for such a complex and demanding scheme. It is

not *Easy-Recovery-from-Loss* as memorizing a new secret is taxing.

While the scheme is very challenging mentally, we rate it as *Accessible* because users shouldn't be prevented from using it due to physical disabilities. While it seems probable that this excludes a significant fraction of users, we consider that a usability and not an accessibility problem. It is *Negligible-Cost-per-User*. It is not *Server-Compatible* nor *Mature* but is *Browser-Compatible* and *Non-Proprietary*.

The scheme is *Quasi-Resilient-to-Physical-Observation*: it will withstand a few, but certainly not twenty observations (unless the task for humans is made infeasible). The scheme is *Resilient-to-Targeted-Impersonation*: the shared secret is random, so knowledge of the user doesn't help. The scheme is not *Resilient-to-Throttled-Guessing* or *Resilient-to-Unthrottled-Guessing*: the protocol is so cumbersome that PIN level strength seems the highest attainable. The scheme is not *Resilient-to-Internal-Observation* (again, unless the memorized secret is extended unreasonably). The scheme is *Resilient-to-Phishing*: a malicious site that lures the user into authenticating learns the secret only if the user can be convinced to authenticate several times at the malicious site. Given how difficult a successful authentication is, this seems improbable. The scheme is *Resilient-to-Theft*: there's nothing to steal. It is *No-Trusted-Third-Party*, *Requiring-Explicit-Consent*, and *Unlinkable*.

4) Word association: Word association is a scheme proposed by Smith [55] that involves giving answers to challenge questions. The user must first register a series of twenty or so word pairs. The first word of each pair will be the challenge and the second a response word that the user associates with the challenge. A very obvious example might be "yellow" for challenge word, paired with "submarine" for response. The word pairs form the shared secret between the user and the server. To authenticate the user must give the correct word association for one or more challenges. As with cognitive schemes such as those of Weinshall [52] and Hopper-Blum [54] this approach has a challenge-response element. However, the response is based on something the user has memorized, or is easy for him to recall, rather than something he calculates. A clear weakness of the scheme is that the word association pairs must be non-obvious. Pairs such as "black" and "white" or "sesame" and "street" offer little security. The scheme can be viewed as an ancestor of preference-based authentication [56].

The scheme is obviously not *Memorywise-Effortless*; however, it does has an advantage over passwords in that it is cued rather than uncued recall. It is not *Scalable-for-Users*: registering many word pairs at dozens of sites is a non-trivial burden. It clearly is *Nothing-to-Carry* but not *Physically-Effortless*. It is *Easy-to-Learn* and *Efficient-to-Use*: the scheme is very simple and involves entering only a small number of challenges. The scheme has *Quasi-Infrequent-Errors*: if users choose their own word pairs errors should be infrequent, but not unknown. The scheme has *Quasi-Easy-Recovery-from-Loss*: re-registering requires choosing new word associations, which is less onerous than learning a new set of images in the Weinshall scheme, for example.

The scheme is *Accessible* and *Negligible-Cost-per-User*. The scheme is not *Server-Compatible* but is *Browser-Compatible*: a simple web browser is all that is needed, but existing servers do not support the scheme. The scheme is not *Mature*: we have little idea how predictable the answers might be if this were deployed at scale. The scheme is *Non-Proprietary*.

The scheme is not *Resilient-to-Physical-Observation*: to be confident of resisting 10–20 observations the user would have to register an infeasibly large number of pairs. The scheme is not *Resilient-to-Targeted-Impersonation*: early user studies found that users can often predict their spouse's word associations [57]. The scheme is not *Resilient-to-Throttled-Guessing*, *Resilient-to-Unthrottled-Guessing* or *Resilient-to-Internal-Observation*: it affords PIN-level strength and withstands only a small number of observations. It is not *Resilient-to-Leaks-from-Other-Verifiers* or *Resilient-to-Phishing*. The scheme is *Resilient-to-Theft*, *No-Trusted-Third-Party* and *Requiring-Explicit-Consent*. The scheme is *Unlinkable*.

F. Paper tokens

1) OTPW: Using paper to store long secrets is the cheapest form of a physical login token. The concept is related to military codebooks used throughout history, but interest in using possession of paper tokens to authenticate humans was spurred in the early 1980's by Lamport's hash-chaining scheme [58], later developed into S/KEY [59]. OTPW is a later refinement, developed by Kuhn in 1998 [60], in which the server stores a larger set of independent hash values, consisting of about 4 kB per user. The user carries the hash pre-images, printed as 8-character values like `IZdB bqyH`. Logging in requires typing a "prefix password" as well as one randomly-queried hash-preimage.

OTPW rates poorly for usability: the prefix password means the scheme isn't *Memorywise-Effortless* or *Scalable-for-Users*; it also isn't *Nothing-to-Carry* because of the paper token. The typing of random passwords means the scheme also isn't *Physically-Effortless*, *Efficient-to-Use* or *Infrequent-Errors*. We do expect that the scheme is *Easy-to-Learn*, as typing in a numbered password upon request is only marginally more difficult than using text passwords. It is also *Easy-Recovery-from-Loss* as we expect most users can easily print a new sheet if needed.

Paper-based tokens are cheap and easy to deploy. We rate OTPW as non-*Accessible* because plain printing may be insufficient for visually-impaired users, though alternatives (e.g. braille) may be available. We consider the price of printing to be *Negligible-Cost-per-User*. While not *Server-Compatible*, the scheme is *Browser-Compatible*. Finally, OTPW has a mature open-source implementation, making it *Mature* and *Non-Proprietary*.

Though OTPW is designed to resist human observation compared to S/KEY, it isn't *Resilient-to-Physical-Observation* because the printed sheet of one-time codes can be completely captured by a camera. Otherwise, OTPW achieves all other security benefits. Because login codes are used only once and ran-

domly generated, the scheme is *Resilient-to-Throttled-Guessing*, *Resilient-to-Unthrottled-Guessing* and *Resilient-to-Internal-Observation*. It is *Resilient-to-Phishing* as it is impractical for a user to enter all of their secrets into a phishing website even if asked, and *Resilient-to-Theft* thanks to the prefix password. As a one-to-one scheme with different secrets for each server, it is *Resilient-to-Leaks-from-Other-Verifiers*, *No-Trusted-Third-Party* and *Unlinkable*. Finally, the typing required makes it *Requiring-Explicit-Consent*.

2) S/KEY: Haller et al. developed S/KEY [59], a predecessor to OTPW, in the 1980s. It is based directly on Lamport's hash-chaining scheme [58], in which a sequence of eventual one-time codes $F(x)$, $F^1(x)$, ... , $F^{N-1}(x)$ are printed onto a sheet of paper, so that the server need only store $F^N(x)$ initially and then store each login code as it is successfully entered, requiring less server storage than OTPW. S/KEY is thus limited to querying one code at a time (OTPW may query several). S/KEY also represents each one-time code as a sequence of common English words such as `ROY HURT SKI FAIL GRIM KNEE`. The scheme is designed to be used standalone, it has no prefix password like OTPW. The S/KEY scheme has been standardized by several RFCs [59], [61], implemented as a pluggable login module for Unix-like systems, and widely deployed.

S/KEY rates very closely to OTPW, with poor usability and good deployability and security. It is *Memorywise-Effortless*, unlike OTPW, because no memorized password is required to authenticate. It is also *Quasi-Infrequent-Errors* because the one-time codes represented as word sequences are more reliable to type (especially on mobile devices). It is otherwise identical from a usability standpoint to OTPW: not *Scalable-for-Users* and not *Nothing-to-Carry* because of the paper token, not *Physically-Effortless* or *Efficient-to-Use* because of the typing and paper retrieval, but *Easy-to-Learn* and *Easy-Recovery-from-Loss* because of its simplicity.

From a deployment standpoint it is identical to OTPW: non-*Accessible* because visually-impaired users can't use plain paper tokens, but *Negligible-Cost-per-User*, *Browser-Compatible*, *Mature* and *Non-Proprietary*. Being non-*Server-Compatible* is the main deployment issue, though it is relatively simple for servers to implement.

Finally, S/KEY is slightly less secure than OTPW (the price paid for its increased usability). S/KEY is non-*Resilient-to-Physical-Observation* in a stronger way because a human attacker only needs to memorise the last one-time code on the sheet, which enables re-deriving all other codes due to the hash-chaining sequence. Similarly, the scheme is only *Quasi-Resilient-to-Phishing* because if phishing website can dupe users into entering the final code it can then re-derive all others. S/KEY is not *Resilient-to-Theft* because, unlike OTPW which requires a prefix password, a stolen token sheet can be used freely. Otherwise, S/KEY achieves all other security properties, like OTPW.

3) PIN + TAN: A very closely related scheme to OTPW is Transaction Authentication Number (TAN codes), deployed by many German and Austrian banks to verify banking transactions [62]. The scheme is typically called PIN+TAN, as a user must type in both a PIN number and a requested TAN code at random from a sheet. TAN codes come in the form of long numeric sequences. In contrast to OTPW, the sheet of printed codes is generated by the bank and mailed to the user instead of being printed upon initial enrollment. Like OTPW, a user can be queried for a random set of codes in order to authenticate. The scheme may be easier to break with a stolen sheet of codes than OTPW, since instead of using a full password it only user a PIN [63], hence *Quasi-Resilient-to-Theft*, and the numeric codes may be slightly easier to type than the mixed-case OTPW codes making it *Quasi-Infrequent-Errors*, but we otherwise grade the scheme identically to OTPW. Two further differences: the centralized printing and mailing cost downgrades the scheme to *Quasi-Negligible-Cost-per-User*, and also *Quasi-Easy-Recovery-from-Loss* as new sheets must be mailed for recovery.

G. Visual crypto tokens

1) PassWindow: Tokens printed onto a transparency are only marginally more expensive than paper tokens but enable increased security. Moni Naor and Adi Shamir launched the field of visual cryptography in 1994 [64], using transparencies to transmit a secret image from one user to another with security provably equivalent to a one-time pad. Of course, since basic visual cryptography schemes are equivalent to one-time pads, an observer can eventually learn the user's secret by repeated observation. Kobara and Matsumoto proposed resisting physical attack in 1996 paper using the physical details of transparencies with limited alignment to prevent shoulder-surfing by an adversary viewing the screen at a different angle than the valid user [65]. Naor and Pinkas developed a challenge-response protocol specifically for authenticating humans in 1997. In this scheme, the verifier displays (or sends to a web browser) one share of an image displaying a one-time login code, which can then be read by a human prover by overlaying their own secret transparency and sent back to the verifier [66].

Recently, a commercial scheme called PassWindow [67] has appeared which uses visual cryptography (though substantially different from Naor and Pinkas' scheme). In PassWindow, a small transparency (which might embedded into the corner of a payment card or ID card) is overlaid by the user on screen, which includes 14 digits of seven-segment display, each digit overlapping the next in one of its virtual column (for a total of 72 segments). A small number of segments (15 in a provided example) are shaded in the user's secret transparency. The verifying website cycles through a sequence of 6 challenge images, 4 of which reveal a full digit in the seven-segment display when overlain with the user's transparency. The user must type all 4 correctly in order to prove possession of his or her secret transparency.

PassWindow has similar usability drawbacks to paper-token schemes. Though it is *Memorywise-Effortless*, it isn't *Scalable-for-Users* or *Nothing-to-Carry* as one token is needed for each verifier. The usability of the scheme appears challenging as it requires spotting a fully-formed decimal digit from a row of noisy line segments and ignoring some images which don't present a

fully-formed image, meaning the scheme isn't *Physically-Effortless*, *Easy-to-Learn*, *Efficient-to-Use*, nor *Infrequent-Errors*. Few users can print their own transparencies, making it non-*Easy-Recovery-from-Loss*.

Visual cryptography is inherently inaccessible for visually impaired users with no obvious remedy, making the scheme non-*Accessible*. While the cost of transparencies is "near-zero" in bulk according to PassWindow, they must be physically distributed so we rate the scheme *Quasi-Negligible-Cost-per-User*. It is not *Server-Compatible*, as challenges must be generated and served, though it is *Browser-Compatible*. PassWindow is based on a *Mature* commercial offering, but isn't *Non-Proprietary*.

The security of the scheme is comparable to S/KEY, with a slight upgrade because it uses limited-visibility techniques introduced by Kobara and Matsumoto [65] and a printing method claimed to prevent photographic capture of the user's secret transparency. These only make the scheme *Quasi-Resilient-to-Physical-Observation* though because a perfectly placed camera can still record the user's secret. Each transparency is a randomly-chosen secret by the server, making the scheme *Resilient-to-Targeted-Impersonation*, *Resilient-to-Leaks-from-Other-Verifiers*, *No-Trusted-Third-Party* and *Unlinkable*. The space of possible secrets is large enough to make the scheme *Resilient-to-Throttled-Guessing* and *Resilient-to-Unthrottled-Guessing*. An independent analysis commissioned by PassWindow claims that 20–30 challenge/response pairs must be observed before the user's secret is leaked [68] making the scheme *Quasi-Resilient-to-Internal-Observation*. Similar to OTPW, it is difficult for a phisher to request the full transparency secret, making the scheme *Resilient-to-Phishing*. The scheme isn't *Resilient-to-Theft* as a stolen token can be used by anyone. Finally, the active typing makes the scheme *Requiring-Explicit-Consent*.

H. Hardware tokens

1) RSA SecurID: Hardware tokens store secrets in a dedicated tamper-resistant module carried by the user; the RSA SecurID [69] family of tokens is the long-established market leader. Here we refer to the simplest dedicated-hardware version, which has only a display and no buttons or I/O ports. Each instance of the device holds a secret "seed" known to the back-end. A cryptographically strong transform generates a new 6-digit code from this secret every 60 seconds. The current code is shown on the device's display. On enrollment, the user connects to the administrative back-end through a web interface, where he selects a PIN and where the pairing between username and token is confirmed. From then on, for authenticating, instead of username and password the user shall type username and "passcode" (concatenation of a static 4-digit PIN and the dynamic 6-digit code). RSA offers an SSO facility to grant access to several corporate resources with the same token; but we rate this scheme assuming there won't be a single SSO spanning all verifiers.

In March 2011 attackers compromised RSA's back-end database of seeds [70], which allowed them to predict the codes issued by any token. This reduced the security of each account to that of its PIN until the corresponding token was recalled and reissued.

The scheme is not *Memorywise-Effortless* nor *Scalable-for-Users* (it needs a new token and PIN per verifier). It's not *Physically-Effortless*, because the user must transcribe the passcode. It's simple enough to be *Easy-to-Learn*, but *Quasi-Efficient-to-Use* because of the transcription. We rate it as having *Quasi-Infrequent-Errors*, like passwords, though it might be slightly worse. It is not *Easy-Recovery-from-Loss*: the token must be revoked and a new one reissued.

The scheme is not *Accessible*: blind users cannot read the code off the token. No token-based scheme can offer *Negligible-Cost-per-User*. The scheme is not *Server-Compatible* (a new back-end is required) but it is *Browser-Compatible*. It is definitely *Mature*, but not *Non-Proprietary*.

As for security, because the code changes every minute, SecurID is *Resilient-to-Physical-Observation*, *Resilient-to-Targeted-Impersonation*, *Resilient-to-Throttled-Guessing* and *Resilient-to-Unthrottled-Guessing* (unless we also assume that the attacker broke into the server and stole the seeds). It is *Resilient-to-Internal-Observation*: we assume that dedicated devices can resist malware infiltration. It's *Resilient-to-Leaks-from-Other-Verifiers*, as different verifiers would have their own seeds; *Resilient-to-Phishing*, because captured passcodes expire after one minute; and *Resilient-to-Theft*, because the PIN is checked at the verifier, so guesses could be rate-limited. It's not *No-Trusted-Third-Party*, as demonstrated by the March 2011 attack, since RSA keeps the seed of each token. It's *Requiring-Explicit-Consent*, as the user must transcribe the passcode, and *Unlinkable* if each verifier requires its own token.

2) YubiKey: The YubiKey [71] by Yubico, shaped like a USB flash drive, is another very cheap authentication token that generates one-time codes. It does not even have a display: instead, it simulates a USB keyboard, saving the user from having to transcribe the code. In its default mode, after the user types a PIN or password (necessary to offer *Resilient-to-Theft*) and positions the cursor in the "YubiKey" entry field, pressing the token's only button causes the YubiKey to type a string of printable characters—the concatenation of a fixed "identity string" (replacing the username) and a one-time code (replacing the password). It also offers other modes: it can generate static passwords (not *Resilient-to-Internal-Observation* on the host computer but *Memorywise-Effortless*, *Physically-Effortless*, *Server-Compatible* and *Browser-Compatible*), it can generate HOTP codes (HMAC-based One-Time Password [72]) and it can "sign" a challenge. There is also a model that supports the popular MIFARE protocol for access control over RFID/NFC but *not* as an alternative physical layer instead of USB: the two systems are separate and it's a bit like having glued a regular Yubikey to a MIFARE card as a single physical token.

Yubikey could be coupled with an SSO system, which is not mandated by the design but seems a natural complement, but we rate the scheme assuming there isn't a global SSO that all verifiers have adopted. We also rate YubiKey with reference to its default mode (user-typed password

+ YubiKey-"typed" identity string). To clarify our ratings: we assume that, if Big Organizations A, B, C all adopt YubiKey in default mode, with Yubico running the back-end on their behalf, they all receive their own batches of company-dedicated and non-interoperable YubiKeys. Other arrangements, enabling clients of both A and C to authenticate with the same YubiKey, are certainly possible; but they would require a degree of inter-organization trust and cooperation that we can't merely assume.

The scheme is not *Memorywise-Effortless* nor *Scalable-for-Users* (different password, and token, for each verifier), not *Nothing-to-Carry* (hardware token) and not *Physically-Effortless* (typed password). It seems to be *Easy-to-Learn*. Without having seen user studies we rate it *Quasi-Efficient-to-Use* (typing password plus fiddling with token) though arguably it might even deserve the full benefit. It is no better than passwords for *Quasi-Infrequent-Errors*. It's not *Easy-Recovery-from-Loss* because the lost token must be revoked and a new one issued.

On deployability it's *Accessible*, like passwords, but not *Negligible-Cost-per-User* because one or more hardware tokens per user are required. It's not *Server-Compatible* in default mode, though it is *Browser-Compatible*. It's *Mature* and already commercialized in several countries. Some support software is released as open source, though it is unclear whether this allows full usage in default mode without relying on Yubico's servers. However, the hardware design is kept secret, preventing the scheme from being *Non-Proprietary*.

On security: it's *Resilient-to-Physical-Observation*, *Resilient-to-Targeted-Impersonation*, *Resilient-to-Throttled-Guessing*, *Resilient-to-Unthrottled-Guessing* thanks to its one-time codes. It's *Resilient-to-Internal-Observation* as a dedicated device. It's *Resilient-to-Leaks-from-Other-Verifiers* and *Resilient-to-Phishing* thanks to one-time codes. It's *Resilient-to-Theft* thanks to the use of a password as second factor. In default mode every verifier relies on Yubico's servers, so it's not *No-Trusted-Third-Party*. It offers *Requiring-Explicit-Consent* as you must press the button. It's *Unlinkable* since the user has different tokens for different verifiers.

3) IronKey: The IronKey [73] is a conceptually different kind of token: rather than a generator of one-time passwords, it is essentially a secure USB flash drive. Its storage is encrypted with a user-supplied password and the on-board firmware wipes the drive after a certain number of failed attempts. The hardware is certified tamper-resistant to FIPS 140-2 Level 3. A bootable version of the device is marketed as a secure end-point that can run a secure browser, pre-loaded with the URL of the issuing bank. This is claimed to make the system keylogging-resistant, under the assumption that malware on the host computer won't run if you boot from the IronKey; but a hardware keylogger would still capture the password that the user must type to unlock the IronKey, since the device has no user interface.[9] In the following evaluation we assume that one IronKey can be preloaded with the URLs of all the verifiers of interest, together with the relevant userids/passwords (à la Firefox Encrypted Password Manager, see Section IV-A1). It's unclear whether the user is expected to work in the environment booted from the IronKey all the time (unpleasant user experience) or only when authenticating to a site (inconvenient multiple reboots).

On usability, we rate the IronKey scheme as *Quasi-Memorywise-Effortless* (one single master password), *Scalable-for-Users* (the device remembers arbitrarily many credentials), not *Nothing-to-Carry* (it's a physical token) and *Quasi-Physically-Effortless* (user must type the master password but only once per session, not at every authentication). The scheme is possibly *Quasi-Easy-to-Learn*, depending on usability of environment of bootable drive, and *Quasi-Efficient-to-Use* (it requires rebooting, so it can't earn the full benefit). It's similar to passwords for *Quasi-Infrequent-Errors* but it's not *Easy-Recovery-from-Loss* because the lost device, besides needing replacement, contains the user's authentication credentials.

On deployability, it's similar to passwords for *Accessible*. It's not *Negligible-Cost-per-User* because it's a physical token. It's *Server-Compatible* and *Browser-Compatible* because nothing changes with respect to passwords. It's *Mature* and already being sold commercially. The hardware is not *Non-Proprietary*, though arguably this has little relevance to the authentication system used on the secure flash drive.

On security we consider it *Resilient-to-Physical-Observation* because, although the master password could be captured from the unprotected host computer, it can't be used without also stealing the IronKey. The scheme is just as bad as passwords for *Quasi-Resilient-to-Targeted-Impersonation*, not *Resilient-to-Throttled-Guessing*, not *Resilient-to-Unthrottled-Guessing*. It's only *Quasi-Resilient-to-Internal-Observation* (if even that) because we are sceptical of the claim that the "secure browser" can't possibly be infected by malware. Like passwords, it's not *Resilient-to-Leaks-from-Other-Verifiers*. It's *Resilient-to-Phishing* because only pre-loaded whitelisted verifiers can be reached. It's *Resilient-to-Theft* thanks to master password. It's *No-Trusted-Third-Party* because no third party is involved. It's *Requiring-Explicit-Consent* because the user must choose to visit the site. Finally, it's as *Unlinkable* as passwords.

4) CAP reader: The CAP (Chip Authentication Program) reader is a stateless calculator-like device with a slot that accepts an EMV bank card. It is part of a system specified by MasterCard to secure online banking transactions using the bank card's chip [74]. In the mode used for authenticating to the bank's web site, the user types the card's PIN into the CAP reader (bypassing keyloggers) which, after talking to the card's chip, displays a one-time 8-digit code that the user must transcribe onto the web form as a substitute for the password. In another mode that does not concern us here the device can authenticate banking transactions (such as sending money to a new payee) by letting the user type in the values of specific fields (such as amount and account number of the payee) and then generating a MAC over the data, which the user then has to transcribe into the browser to prove that the

[9] It's true that this attack requires physical access to the host computer; but it means that using the IronKey's "secure browser" doesn't make it safe to use from a cybercafé.

card and its PIN were used to authorize the transaction.

The scheme is not *Memorywise-Effortless* nor *Scalable-for-Users*, as each user needs a new card (and PIN) per verifier. It's not *Nothing-to-Carry* because users must carry at least their bank cards (plus their own CAP reader if they want "trusted path"). It's not *Physically-Effortless*, as the user must transcribe the 8-digit code. It's probably *Easy-to-Learn* (perhaps a generous rating), *Quasi-Efficient-to-Use* (it's somewhat fiddly but not too bad) and *Quasi-Infrequent-Errors* (similar to typing a password, perhaps slightly harder). It's not *Easy-Recovery-from-Loss* (even though the reader itself is stateless) if we also consider loss of the chip cards.

On deployability it's not *Accessible* (blind users can't read the code off the reader's display), not *Negligible-Cost-per-User*, not *Server-Compatible*, definitely *Browser-Compatible*, definitely *Mature* and, finally, not *Non-Proprietary*.

On security, the scheme is *Resilient-to-Physical-Observation*, *Resilient-to-Targeted-Impersonation*, *Resilient-to-Throttled-Guessing* and *Resilient-to-Unthrottled-Guessing* thanks to the one-time codes; *Resilient-to-Internal-Observation* as a dedicated device; *Resilient-to-Leaks-from-Other-Verifiers* and *Resilient-to-Phishing* thanks to the one-time codes; *Resilient-to-Theft* thanks to the PIN, with guesses rate-limited by the card's chip; *No-Trusted-Third-Party* because each web site is its own verifier;[10] *Requiring-Explicit-Consent* because users must transcribe the code; and finally *Unlinkable* because each web site is its own verifier.

Note that, while this scheme scores fully on all the security benefits examined in our table, the system clearly isn't perfect: it is still affected by several known security flaws [74], [75].

5) Pico: Pico [8] is a design for a dedicated password-replacement token whose primary goals are explicitly *Memorywise-Effortless* and *Scalable-for-Users*. It aims to improve on passwords with respect to both usability and security, but explicitly ignores deployability, to avoid premature optimization: it does not attempt to be cheap or backwards-compatible and it doesn't even exist yet, let alone have a mature implementation. This makes it one of the hardest schemes to rate fairly, among those in our table: we err on the side of caution and take the author's claims with some skepticism, particularly with respect to usability.

The Pico runs a multi-channel protocol [76] with the verifying back-end: over the optical channel, the verifier offers a visual code with a hash of its public key, which the user acquires with the Pico's camera to signify her intent to authenticate to that application; over the wireless channel, instead, the Pico and the verifier run a TLS-like protocol that provides mutual authentication and establishes a confidentiality-and-integrity-protected channel between the two entities. The Pico stores in its memory a different key pair for each verifier, so as to be *Unlinkable*. In order to be mitigate theft while remaining fully *Memorywise-Effortless*, unlike most other tokens the Pico is not unlocked by a PIN but by a k-out-of-n secret sharing scheme that involves the proximity of a swarm of radio-based "Picosiblings", as well as sensing a biometric input from the owner.

Pico is *Memorywise-Effortless* and *Scalable-for-Users* by design: the same Pico will pair with thousands of independent verifiers. As a physical token it's not *Nothing-to-Carry*. It's *Physically-Effortless*, as the credential is transferred wirelessly rather than being typed. It's possibly not *Easy-to-Learn* owing to the complexity of the Picosiblings management. We rate it *Quasi-Efficient-to-Use* and *Quasi-Infrequent-Errors* (visual code acquisition and no typing suggest we might grant the full benefit in both cases; but without any user testing "we'll believe it when we see it"). We consider it not *Easy-Recovery-from-Loss*, despite its automatic backup at every recharge, because one needs to go and buy a blank Pico into which to restore the backup.

Unsurprisingly given its design goals, Pico scores quite poorly on deployability. It's not *Accessible* (it requires coordinated use of camera, display and buttons) and it doesn't even try to be *Negligible-Cost-per-User*, *Server-Compatible* or *Browser-Compatible*. It can't be *Mature* without an implementation. Its only redeeming quality in this department is that it is *Non-Proprietary*.

It does better on security: it is *Resilient-to-Physical-Observation*, *Resilient-to-Targeted-Impersonation*, *Resilient-to-Throttled-Guessing* and *Resilient-to-Unthrottled-Guessing* thanks to its use of a TLS-like[11] public-key-based mutual authentication challenge-response protocol. It's *Resilient-to-Internal-Observation* as a dedicated hardware device. It's *Resilient-to-Leaks-from-Other-Verifiers* because it uses a different credential for every verifier. The multi-channel protocol and TLS-like construction make it *Resilient-to-Phishing*. It is *Quasi-Resilient-to-Theft* if we consider the Picosiblings at least as secure as a non-rate-limited PIN. It offers *No-Trusted-Third-Party* because all pairings are end-to-end, with no reliance on any PKI. It offers *Requiring-Explicit-Consent*, because the user must acquire the visual code to initiate the authentication protocol; and *Unlinkable* because it uses a different key pair for each verifier.

6) Nebuchadnezzar: The Nebuchadnezzar [77], or Neb, is another thought-experiment hardware token, like the Pico which it predates. Here, as with CAP (Section IV-H4), Cronto (Section IV-I2) and MP-Auth (Section IV-I3), the emphasis is more on protecting entire transactions than merely authenticating a user to a verifier. The central observations of the authors are that it is not possible to secure a general-purpose operating system (the attack surface is too large) but also that it is not possible to make a bulletproof operating system easy and pleasant to use. The proposed solution is therefore to use an insecure host computer for most operations, thus offering a nice user experience, but to confirm the crucial elements of a transaction (e.g. authenticating, authorizing, deciding what to sign) on the trusted-path interface of the Neb, designed

[10] comparing this rating to that of the RSA SecurID for the same benefit, one might argue whether the card manufacturers should count as a trusted third party in this context; on balance we feel the situation is somewhat different here, but the topic makes for an interesting debate

[11] While Pico uses a TLS-like protocol, it does not rely on the existing CA system for certifying public keys.

to be absolutely secure even if too spartan to be usable for everyday tasks such as browsing the web or editing files.

This high-level position paper does not give details on how the Neb gets paired to its user and how it gets locked and unlocked, so it's hard to assess how it would score on most of our usability benefits, and on others like *Resilient-to-Theft*. For this reason we do not give scores for Neb in the table. We expect however that it would be rated highly on most security benefits.

I. Mobile-Phone-based

1) Phoolproof: Phoolproof Phishing Prevention [78] is another token-based design, but one in which the token is a mobile phone with special code and crypto keys. It uses public key cryptography and an SSL-like authentication protocol and was designed to be as compatible as possible with existing systems.

Phoolproof was conceived as a system to secure banking transactions against phishing, not as a password replacement. The user selects a desired site from the whitelist on the phone; the phone talks wirelessly to the browser, causing the site to be visited; an end-to-end TLS-based mutual authentication ensues between the phone and the bank's site; the user must still type the banking website password into the browser. Thus the scheme is not *Memorywise-Effortless*, nor *Scalable-for-Users*. It has *Quasi-Nothing-to-Carry* (the mobile phone). It's not *Physically-Effortless* as one must type a password. We rate it *Easy-to-Learn*, perhaps generously, and *Quasi-Efficient-to-Use* as it requires both typing a password and fiddling with a phone. It's no better than passwords on *Quasi-Infrequent-Errors*, since it still uses one. The only recovery mechanism is revocation and reissue, so it doesn't have *Easy-Recovery-from-Loss*.

On deployability: it's *Quasi-Accessible* insofar as most disabled users, including blind people, can use a mobile phone too (note the user doesn't need to transcribe codes from the phone). We assume most users will already have a phone, though perhaps not one of the right type (with Java, Bluetooth etc), hence it has *Quasi-Negligible-Cost-per-User*. The scheme requires changes, albeit minor, to both ends, so it's *Quasi-Server-Compatible* but, by our definitions, not *Browser-Compatible* because it uses a browser plugin. It's not really *Mature* (only a research prototype), but it is *Non-Proprietary*.

On security: it's *Resilient-to-Physical-Observation*, *Resilient-to-Targeted-Impersonation*, *Resilient-to-Throttled-Guessing*, *Resilient-to-Unthrottled-Guessing* because, even after observing or guessing the correct password, the attacker can't authenticate unless he also steals the user's phone, which holds the cryptographic keys. It's *Quasi-Resilient-to-Internal-Observation* because malware must compromise both the phone (to capture the private keys) and the computer (to keylog the password). It's *Resilient-to-Leaks-from-Other-Verifiers* because the phone has a key pair per verifier, so credentials are not recycled. It's definitely *Resilient-to-Phishing*, the main design requirement of the scheme. It's *Resilient-to-Theft* because possession of the phone is insufficient: the user still needs to type user ID and password in the browser (for additional protection against theft, the authors envisage an additional PIN or biometric to authenticate the user to the device; we are not rating this). The scheme is *No-Trusted-Third-Party* if we disregard the CA that certifies the TLS certificate of the bank. It's *Requiring-Explicit-Consent* because the user must type user ID and password. Finally it's *Unlinkable* because the phone has a different key pair for each verifier.

2) Cronto: Cronto [79] is a commercial and proprietary transaction authentication system to protect online banking transactions against malware on the user's browser. Cronto's argument is that authenticating the user is insufficient if a valid user is tricked into authorizing a fraudulent transaction in which the beneficiary and amount are different from what is shown on the screen of the malware-ridden host. It uses a camera phone to acquire a visual code. The bank's web site generates an encrypted version of the transaction details, including a nonce, and displays it as a visual code on the web page. The mobile phone Cronto application acquires the code, decrypts it with the per-device key it shares with the bank, and displays the transaction details on the phone's screen. It also computes a MAC over the details and renders it as a one-time textual password for that transaction. The user must check that the transaction details are the intended ones and, if yes, transcribe the one-time password back into the web page to authenticate the transaction.

For secure login, rather than for transaction authorization, the web site displays a cryptogram; then the user acquires it with the camera and the phone displays a one-time password that the user must copy into the web page (in addition to the usual password) in order to log in. We rate the phone variant here.

Cronto is not *Memorywise-Effortless* nor *Scalable-for-Users* (a standard password for each verifier is still required). It's *Quasi-Nothing-to-Carry* (the user probably already carries a phone, though maybe not one with Java and a camera). It's not *Physically-Effortless* (the user must transcribe the one-time password and type the regular password). Perhaps generously, we consider it *Easy-to-Learn*. It's only *Quasi-Efficient-to-Use* (and perhaps that's generous too) owing to the extra fiddling with the phone besides typing the password. It's *Quasi-Infrequent-Errors*, like passwords, though the rating might arguably be lower because there's also the one-time code to transcribe. It's not *Easy-Recovery-from-Loss* because you must replace the phone and it contained a pre-shared key with each verifier.

For deployability, having to acquire the visual code and transcribe the one-time password makes Cronto not *Accessible*. It's *Quasi-Negligible-Cost-per-User* because the user might already have a suitable phone. It's not *Server-Compatible* (the back-end needs changes) but it's *Browser-Compatible* and *Mature*, being currently trialled by several banks. It's not *Non-Proprietary*.

As for security, the use of the one-time code in addition to the password makes it *Resilient-to-Physical-Observation*, *Resilient-to-Targeted-Impersonation*, *Resilient-to-Throttled-Guessing* and *Resilient-to-Unthrottled-Guessing*. It's *Quasi-Resilient-to-Internal-Observation* because malware must compromise both

phone and computer. It's *Resilient-to-Leaks-from-Other-Verifiers* because, even if account passwords were reused across verifiers, the attacker would still need to steal the phone to get the pre-shared keys. It's *Resilient-to-Phishing* because the visual challenge-response doesn't reveal the secret in the user's phone. It's *Resilient-to-Theft* because the standard password acts as a second factor. It's *No-Trusted-Third-Party* because the only parties involved are the user and the bank. It's *Requiring-Explicit-Consent* because the user must acquire the visual code and transcribe the one-time password. Finally it's *Unlinkable*, assuming a different key per verifier.

3) MP-Auth: MP-Auth [6] is another phone-based authenticator. The phone is used as the allegedly malware-free trusted endpoint[12] that performs cryptographic operations and into which the user enters the password. The intention is to protect the user's password entry, assuming that her PC might be compromised by malware. The password (used as the seed for a challenge-response that is then performed automatically between the phone and the web server, communicating through the web browser on the user's PC) is not stored on the phone but is pre-shared between the human user and the back-end verifier. The main motivation for MP-Auth was to address the problem of entering a user password on an untrusted PC such as in a cybercafe (rather than to replace passwords); a secondary motivation was to provide "transaction integrity", though we do not discuss or rate that aspect of its functionality herein.

Ratings change substantially if the same password is used with all verifiers or if a different one is used on every site. In scoring MP-Auth we adopt the same assumptions as for regular passwords: users will be instructed never to reuse passwords (hence not *Memorywise-Effortless* and not *Scalable-for-Users*) but some of the time they will anyway (hence not *Resilient-to-Leaks-from-Other-Verifiers*).

The scheme is *Quasi-Nothing-to-Carry* (the user probably carries a suitable mobile phone already). It's not *Physically-Effortless* (a password must be typed into the phone). Perhaps generously, we consider the scheme *Easy-to-Learn*; but having to type a password into a phone means it can't be more than *Quasi-Efficient-to-Use* (if that) and that it can't offer *Infrequent-Errors*. It's *Quasi-Easy-Recovery-from-Loss* because the phone does not hold authentication secrets and can thus be replaced relatively painlessly.

It's *Quasi-Accessible* as it requires typing a password into the phone but not transcribing anything. It's *Quasi-Negligible-Cost-per-User* because the user might already have a suitable phone. It's not *Server-Compatible* because it requires custom code. It's not *Browser-Compatible* because the browser needs a plugin to accept Bluetooth data from the mobile phone. It's not *Mature* (just a prototype implementation) but it is *Non-Proprietary* (openly published).

On security, the scheme is not *Resilient-to-Physical-Observation* as the attacker could observe the password entered into the phone. It's no better than passwords concerning *Quasi-Resilient-to-Targeted-Impersonation*, not *Resilient-to-Throttled-Guessing*, not *Resilient-to-Unthrottled-Guessing*. It's not *Resilient-to-Internal-Observation* because malware on just the phone can break the scheme. Like passwords, it's not *Resilient-to-Leaks-from-Other-Verifiers*. It's *Resilient-to-Phishing* (though possibly not to active MITM) thanks to the challenge-response. It's *Resilient-to-Theft* because the phone does not store any secrets. It's *No-Trusted-Third-Party* because there are no intermediaries. It's *Requiring-Explicit-Consent* because the user must type her password into phone and it's *Unlinkable* to the extent that passwords are.

4) OTP over SMS: An interesting class of approaches provides authentication assurances by using externally-observable *capabilities*, e.g., the ability to alter or access a resource. Suppose a web site or world-readable public database (e.g., DNS record) is writable only by authorized entities; if an entity purportedly having write access is requested to post a challenge string to that resource, appearance of the update suggests ownership or control. The ability to retrieve and/or respond to an email link sent to an account at a designated domain is used by digital certificate authorities issuing *domain-validated certificates*. Email providers use analogous methods for automated password resets, avoiding the expense of human service agents. Service providers may send to a (physical) postal mail address on record, an authorization code (PIN or password) to initialize login access by web or phone to an account. In such *cross-channel* methods and related *multi-channel protocols* (see [76], [80]), security derives from independent corroboration or control of multiple independent channels.

OTP over SMS is a specific such authentication method. The base idea is so generic that we shouldn't expect an academic reference on it (hence the hole in the table). A one-time-password (OTP) is sent as a conventional text message (SMS) to a cellphone number on record, requiring no specialized software application. The OTP may be used for account login, transaction authorization, or as a second factor in a two-factor scheme. The act of a user initiating authentication to a remote site triggers the site to send an OTP via SMS to the user's mobile phone (or via a voice message to a wireline number), which the user must then transcribe into the site's login page.

Regarding usability, the approach is *Memorywise-Effortless*, *Scalable-for-Users* (the timing context associates the message to the particular account) and *Quasi-Nothing-to-Carry* (many users already carry SMS-capable phones). It is not *Physically-Effortless* (OTP must be transcribed), is *Easy-to-Learn*, but not *Efficient-to-Use* (the user must wait to receive the message, sometimes for minutes, and then enter it, e.g., on a keyboard). We rate it *Quasi-Infrequent-Errors* as occasional errors may occur on user entry of the OTP. A lost mobile phone takes time to replace, but here the phone stores no authentication secrets; hence *Quasi-Easy-Recovery-from-Loss*.

On deployability, we rate OTP-over-SMS as *Quasi-Accessible*, assuming that blind people have ways of reading SMS messages, but not *Negligible-Cost-per-User* due to the total system-side cost of sending SMS mes-

[12] In contrast to the authors' viewpoint, we assume the phone may be infected by malware, per our definition for *Resilient-to-Internal-Observation*.

sages to all users. The scheme is not *Server-Compatible* (some verifier-side changes are necessary) but is *Browser-Compatible*, *Mature* (already in use by banks), and *Non-Proprietary*.

On security: it's *Resilient-to-Physical-Observation*, *Resilient-to-Targeted-Impersonation*, *Resilient-to-Throttled-Guessing*, *Resilient-to-Unthrottled-Guessing* thanks to the inherent properties of OTPs. We rate it only *Quasi-Resilient-to-Internal-Observation* considering that an attacker might intercept the SMS or phone call and race against the user to enter the OTP. It's *Resilient-to-Leaks-from-Other-Verifiers* and *Resilient-to-Phishing* thanks again to the properties of OTP. It's *Quasi-Resilient-to-Theft*, assuming the stolen phone is PIN-protected. It's not *No-Trusted-Third-Party* (the wireless provider must be trusted). It's *Requiring-Explicit-Consent* (OTP must be retyped) and *Unlinkable* (OTPs are all different).

5) Google 2-Step: Introduced in 2011, Google 2-Step Verification [81] is a commercial offering that combines a user's traditional, memorized password with one-time codes, which are either sent over SMS or voice to a registered phone or can be generated as time-dependent passwords by a dedicated mobile phone application called Google Authenticator that maintains a secret key. The scheme supports long static passwords for applications (such as mail readers making IMAP mailbox requests) that lack access to the one-time codes; these passwords can be managed and revoked by the user. The scheme also supports long, printable backup secrets for override in the case of a lost phone. Using cookies, the scheme optionally remembers any browser from which a user has successfully authenticated; it then won't require that user to use the phone again when authenticating from that browser within a 30-day window. Below we rate the variant without Google Authenticator, which works with more phones, and in which users accept the long-term cookies, for higher usability.

The scheme is not *Memorywise-Effortless* or *Scalable-for-Users* because it still uses a traditional password during authentication. It is *Quasi-Nothing-to-Carry*: while it requires a phone, it works with feature phones as well as smart phones. It's not *Physically-Effortless* as the user must still type the password. It is seemingly *Easy-to-Learn*. We rate it as *Quasi-Efficient-to-Use* and *Quasi-Infrequent-Errors* because the additional second factor (phone) is only required every 30 days. We rate it as *Quasi-Easy-Recovery-from-Loss* because the phone contains no secrets (but the scheme loses this benefit if Google Authenticator is used, because the inconvenience of having to replace the phone is then compounded by the fact that the lost phone also holds a secret key). Note that users who have safely stored paper backups can easily disable the requirement of using their phone if they lose it; but then they may fail to save those backups in the first place.

From a deployment standpoint, we rate the scheme as *Quasi-Accessible*, in line with simple OTP-over-SMS. It isn't *Negligible-Cost-per-User* because of the reliance on SMS messages, which Google must provide for users unable to install the Authenticator phone application. The scheme is not *Server-Compatible*, as it requires much more complicated server behavior, but it is *Browser-Compatible*. While Google has developed the scheme sufficiently to rate it as *Mature*, it is not *Non-Proprietary* as many details, particularly of the one-time code generation application, are not publicly disclosed and the intellectual property status is unclear.

Because its use requires both a regular password and a one-time secret from the phone, the scheme is generally resistant to guessing and observation attacks; however we rate it only *Quasi-Resilient-to-Physical-Observation* and *Quasi-Resilient-to-Targeted-Impersonation* because the 30-day cookie means the one-time secret is usually not requested, so a local attacker observing or guessing the password and having temporary access to the browser can get in. Remote guessing attackers won't have access to the cookie so we still grant a full *Resilient-to-Throttled-Guessing* and *Resilient-to-Unthrottled-Guessing* thanks to the second factor. However, we rate the scheme as not *Resilient-to-Internal-Observation* because the long-term cookie gives malware a long, if finite, time to use a stolen cookie and password. As is, the scheme works only for one verifier and is thus implicitly *Resilient-to-Leaks-from-Other-Verifiers*; it would retain this benefits if the scheme were adopted independently by several other verifiers, and lose them if the other verifiers all relied on Google to provide the back-end. The second factor (SMS or cookie) makes it *Resilient-to-Phishing* as a user can't easily turn this credential over to a phishing site. The scheme is *Resilient-to-Theft* because it requires a password in addition to possession of the phone. As above, the scheme is currently *No-Trusted-Third-Party* as Google is the only verifier, though Google could become a trusted third-party if the scheme were used for single-sign-on. It's *Requiring-Explicit-Consent* because the password must be typed. Finally, the scheme is *Unlinkable* as Google is currently the only verifier, though this could also change.

J. Biometrics

1) Fingerprint recognition: Biometrics [82] are the "what you are" means of authentication, leveraging the uniqueness of physical or behavioral characteristics across individuals. We discuss in detail *fingerprint* biometrics [83]; our summary table also rates *iris recognition* [84] and voiceprint biometrics [85]. In rating for our remote authentication application, and *biometric verification* ("Is this individual asserted to be Jane Doe really Jane Doe?"), we assume unsupervised biometric hardware as might be built into client devices, vs. verifier-provided hardware, e.g., at an airport supervised by officials.

Fingerprint biometrics offer usability advantages *Memorywise-Effortless*, *Scalable-for-Users*, *Easy-to-Learn*, and *Nothing-to-Carry* (no secrets need be carried; we charge elsewhere for client-side fingerprint readers not being currently universal). Current products are at best *Quasi-Physically-Effortless* and *Quasi-Efficient-to-Use* due to user experience of not *Infrequent-Errors* (the latter two worse than web passwords) and fail to offer *Easy-Recovery-from-Loss* (here equated with requiring an alternate scheme in case of compromise, or users becoming unable to provide the biometric for physical reasons).

Deployability is poor—we rate it at best *Quasi-Accessible* due to common failure-to-register biometric issues; not *Negligible-Cost-per-User* (fingerprint reader has a cost); neither *Server-Compatible* nor *Browser-Compatible*, needing both client and server changes; at best *Quasi-Mature* for unsupervised remote authentication; and not *Non-Proprietary*, typically involving proprietary hardware and/or software.

We rate the fingerprint biometric *Resilient-to-Physical-Observation* but serious concerns include easily fooling COTS devices, e.g., by lifting fingerprints from glass surfaces with gelatin-like substances [86], which we charge by rating not *Resilient-to-Targeted-Impersonation*. It is *Resilient-to-Throttled-Guessing*, but not *Resilient-to-Unthrottled-Guessing* for typical precisions used; estimated "effective equivalent key spaces" [9, page 2032] for fingerprint, iris and voice are 13.3 bits, 19.9 bits and 11.7 bits respectively. It is not *Resilient-to-Internal-Observation* (captured samples of static physical biometrics are subject to replay in unsupervised environments), not *Resilient-to-Leaks-from-Other-Verifiers*, not *Resilient-to-Phishing* (a serious concern as biometrics are by design supposed to be hard to change), and not *Resilient-to-Theft* (see above re: targeted impersonation). As a plus, it needs *No-Trusted-Third-Party* and is *Requiring-Explicit-Consent*. Physical biometrics are also a canonical example of schemes that are not *Unlinkable*.

2) Iris recognition: In rating the physical biometric of iris recognition [84], [87] for our target application of remote authentication, we consider unsupervised client devices with low-priced COTS sampling hardware (e.g., a built-in camera atop screens as in many laptops and desktop computers), to match our rating of fingerprint biometrics with built-in readers (as found on laptops or specialized mice). This results in iris recognition being similarly rated as *Memorywise-Effortless*, *Scalable-for-Users*, at best *Nothing-to-Carry*, *Quasi-Physically-Effortless*, *Easy-to-Learn* and *Quasi-Efficient-to-Use* but not *Easy-Recovery-from-Loss*. (For *Nothing-to-Carry* there is indeed nothing-to-carry regarding secrets, but we note that client-side iris-suitable cameras are not currently universal; a combination computer mouse/iris camera is available, which users can position to take an iris image.) We rate iris not *Infrequent-Errors* (due to lack of data) and *Quasi-Requiring-Explicit-Consent* (a built-in camera may not require user consent), the latter slightly lower than fingerprint. Overall, perhaps surprising experts who might consider the iris biometric as a superior modality, it is roughly comparable to fingerprint for our target application, within the coarseness of our metrics.

For the six deployability benefits and all security benefits (see below) other than explicit consent as noted above, iris rates the same as fingerprint recognition. It requires changes to both client- and system ends, typically with proprietary hardware and/or software, and is only *Quasi-Accessible* due to common failure-to-register biometric issues.

On security, we rate iris *Resilient-to-Physical-Observation* with attacks related to relatively easily obtaining high-quality images of a user's iris, sufficient to fool COTS systems, charged instead against ratings as not *Resilient-to-Targeted-Impersonation*, not *Resilient-to-Internal-Observation*, and not *Resilient-to-Theft*; the static nature of physical biometrics makes captured samples subject to replay in unsupervised environments. We grant *Resilient-to-Throttled-Guessing* but (as explained in the fingerprint section) not *Resilient-to-Unthrottled-Guessing* and not *Unlinkable* nor *Resilient-to-Leaks-from-Other-Verifiers* nor *Resilient-to-Phishing*—the latter being serious concerns, as for fingerprint biometrics.

3) Voice recognition: Voice recognition [85] is a physical-behavioral biometric depending in part on physiological properties that generate voice. Our focus on remote authentication assumes technology such as a mobile phone or consumer PC with built-in microphone. To preclude trivial record-and-replay attacks in this unsupervised application, we assume time-variant challenge phrases are used.

Voice biometrics benefit ratings have much in common with the fingerprint and iris recognition physical biometrics discussed above, with largely similar justifications. On usability benefits, voice shares the serious concern of lack of *Easy-Recovery-from-Loss*, is only *Quasi-Physically-Effortless* (directly from the definition), and even assuming that viable deployments would adjust the error rate, ambient noise and practical challenges may result in not *Infrequent-Errors* and thus at best *Quasi-Efficient-to-Use*. On usability benefits, voice thus rates the same as fingerprint biometrics.

On deployability, voice is similarly rated *Quasi-Accessible*, but improves upon both fingerprint and iris, as we grant it *Quasi-Negligible-Cost-per-User* (charging minorly for commonly-deployed built-in microphones), and *Quasi-Browser-Compatible* again assuming commonly deployed (but not universal) microphones, here with built-in voice processing support. We also withhold awarding full client-end compatibility due to lack of ubiquitous trusted computing base and trusted path from input to remote verifier. As with the other biometrics discussed, for unsupervised remote authentication, we rate voice only *Quasi-Mature*.

On the security grouping, voice biometrics suffer the same serious concerns as fingerprint, being not *Resilient-to-Leaks-from-Other-Verifiers*, not *Resilient-to-Phishing*, not *Resilient-to-Theft*, and not *Unlinkable*. We rate voice (perhaps generously) *Quasi-Resilient-to-Throttled-Guessing* but (based on statistics cited in the fingerprint section) not *Resilient-to-Unthrottled-Guessing*; the dynamic nature of behavioral biometrics raises interesting research questions on the efficacy of targeted impersonation by generative attacks [88], a topic not aggressively pursued in the conventional biometrics literature. We assume a targeted generative attack will render voice biometrics neither *Resilient-to-Targeted-Impersonation* nor *Resilient-to-Internal-Observation* since capture of recorded samples likely allows generative attacks even for time-variant challenge phrases.

K. Recovery

1) Traditional personal knowledge questions: Questions based on personal knowledge, or PKQs, such "what

is your mother's maiden name?" are hoped to be more memorable than passwords remembered specifically for authentication. This concept is deployed on a much larger scale than most other authentication mechanisms (being observed at 17% of websites in Bonneau and Preibusch's survey [13]), usually as a fallback for password authentication.

The usability of PKQs is comparable to passwords. We consider them *Quasi-Memorywise-Effortless*, the only advantage over passwords, because the answers need not be remembered specifically for authentication. Indeed, they have been found to be much more memorable than passwords [89] though users often forget the precise form of their answers. Otherwise, usability is identical: PKQs aren't *Scalable-for-Users*, possibly in a worse way than passswords in that there is only a small set of possible questions which are suitable for use. They aren't *Physically-Effortless* as they must be typed, but *Easy-to-Learn* and *Efficient-to-Use* due to years of user experience. Like passwords, PKQs are only *Quasi-Infrequent-Errors* because users often forget the exact form of their answers. PKQs can be easily reset, making them *Easy-Recovery-from-Loss*.

Deployment is close to passwords, being *Accessible*, *Negligible-Cost-per-User*, not *Server-Compatible* (though many sites already implement them, not as a primary authentication mechanism), *Browser-Compatible*, *Mature*, and *Non-Proprietary*. However, while users will not be prevented from using the system due to physical disability, some users may be disenfranchised—questions often don't apply to a considerable portion of the population, even if several choices are provided.[13]

Security-wise, a wealth of academic research has focused on the vulnerabilities of PKQs. User studies have demonstrated the ability of friends, family, and acquaintances to guess answers correctly [57], [91], [92], for example, Schechter et al.'s 2009 study [91] found that 17% of acquaintances who weren't deemed trustworthy enough to share a password with were able to guess correct answers to personal knowledge questions within 5 guesses. Many questions used in practice have a tiny set of possible answers [93], [94] or can be researched in public databases or online social networks [95], [96]. The distribution of feasible answers, such as surnames in the population, is also typically skewed enough to limit the security of most questions to 10 bits [97].

Thus, PKQs are probably the weakest scheme possible from a security standpoint, being non-*Resilient-to-Targeted-Impersonation* due to acquaintance attacks, as well as not *Resilient-to-Throttled-Guessing* or *Resilient-to-Unthrottled-Guessing* due to the weakness of the questions. PKQs also aren't *Resilient-to-Physical-Observation* or *Resilient-to-Internal-Observation* due to their static nature, and not *Resilient-to-Leaks-from-Other-Verifiers* as answers are typically stored un-hashed to enable more liberal string matching (and nearly all sites register the same types of questions [97]). PKQs also aren't *Resilient-to-Phishing*, similar to passwords. However, like passwords their one-to-one nature makes them *No-Trusted-Third-Party*, *Requiring-Explicit-Consent*, and *Unlinkable*.

2) Preference-based authentication: "Preference-based authentication" [56], [98] is a 2008 scheme which seeks to address some of the weaknesses of basic PKQs by having users choose a number (16 is suggested) of items like "rap music" or "vegetarian food" which they strongly like or dislike from a set of possibile items vetted for being liked and disliked by roughly equal numbers of people. To authenticate, the user must then re-classify the specified items, with some error tolerance. A usability study suggests a negligible false negative rate can be achieved while limiting statistical guessing to a 0.5% chance of success [56].

The scheme appears to have the same benefit of being *Quasi-Memorywise-Effortless* like traditional PKQs, though there are no usability studies of this. Otherwise, the scheme is similar to traditional PKQs from a usability standpoint: non-*Scalable-for-Users*, *Nothing-to-Carry*, non-*Physically-Effortless*, and *Easy-to-Learn*. The requirement to choose many preferences means the scheme isn't *Efficient-to-Use*, but this allows some error tolerance, making it *Infrequent-Errors*. Like Word Association, the scheme has a relatively lengthy enrollment process so we rate it as only *Quasi-Easy-Recovery-from-Loss*.

Preference-based authentication is generally similar to PKQs from a deployment standpoint, being *Accessible* (though like with PKQs some items used may not apply to some users), *Negligible-Cost-per-User* and *Browser-Compatible*, though not *Server-Compatible*. Unlike PKQs, preference-based authentication isn't *Mature* as it has not seen widespread usage, and is not *Non-Proprietary*.

The security is comparably poor to traditional PKQs, not being *Resilient-to-Physical-Observation*, *Resilient-to-Internal-Observation*, or *Resilient-to-Leaks-from-Other-Verifiers* due to the static nature of the scheme and not *Resilient-to-Throttled-Guessing* or *Resilient-to-Unthrottled-Guessing* due to the relatively small answer space. The authors claim that preferences are difficult for one's acquaintances to guess, but in the absence of a usability study we rate the scheme only *Quasi-Resilient-to-Targeted-Impersonation*.[14] The scheme is, however, *Resilient-to-Phishing*, as a phishing site won't know which set of items the user has rated during enrollment. Finally, like traditional PKQs the scheme is *No-Trusted-Third-Party*, *Requiring-Explicit-Consent*, and *Unlinkable*.

3) Social Re-authentication: Using somebody you know as a "fourth factor" in authentication was proposed by Brainard et al. [99]. This is not generally regarded as a primary means of authenticating, but as a last resort or emergency authentication when the primary method is unavailable (e.g., a password has been forgotten or a token has been lost or stolen). The user contacts their trusted friend, preferably by telephone or in person. The trusted friend then contacts the server and retrieves a "vouchcode" for the user. The vouchcode is used to enable a one-time authentication, which enables the user to change the password or get a new token. The key part of the protocol

[13] Allowing users to compose their own questions is not an effective solution. Research by Just and Aspinall found that most users choose questions with very small answer spaces [90].

[14] Their publications mentioned an experiment on guessing by acquaintances, but did not report numerical results.

is that the trusted friend verifies that it is indeed the user he is dealing with. Shortcuts on this step, such as acting on a one-line email of the form "I'm locked out can you send a code?" can be fatal to the security of the scheme.

The scheme is not *Memorywise-Effortless*: the user must remember who has been designated as trustee. Neither is it *Scalable-for-Users*: invoking a trusted friend every time, even for re-authentication, appears too large a burden to scale to dozens of sites per user. The scheme is *Nothing-to-Carry*, but is not *Physically-Effortless*. It is *Easy-to-Learn*, the idea is very easy to understand and learn, it is not, however, *Efficient-to-Use*: there is a significant delay, and it requires effort of the trusted friend every time it is used. The scheme is *Infrequent-Errors*: assuming that the correct friend is selected, and co-operates. It is certainly possible that a friend who was nominated as the trustee is not available, or no longer a friend, but we assume that this is rare. The scheme is not *Easy-Recovery-from-Loss*: if, for any reason, the trustee is unable or unwilling to vouch for the user then recovery is hard.

The scheme is *Accessible* and *Negligible-Cost-per-User*: we presume few users would have difficulty nominating a trusted friend, and the marginal cost is low. It is not *Server-Compatible*, but is *Browser-Compatible*. The scheme is *Quasi-Mature*: facebook has been offering a variant as a backup authentication scheme since October of 2011. As the intellectual property state is unclear we rate the scheme not *Non-Proprietary*.

The scheme is *Quasi-Resilient-to-Physical-Observation*: an observer would figure out who the trusted friend is, but this information seems hard to exploit if the friend follows protocol and only vouches after establishing contact with the user. It is not *Resilient-to-Targeted-Impersonation*: there is extreme reliance on the trusted friend getting a vouchcode only after speaking to the user. A third friend, or a person who knows both the user and trustee might be able to manipulate the trustee into releasing the code based solely on an email contact. The scheme is *Resilient-to-Throttled-Guessing*, *Resilient-to-Unthrottled-Guessing*. It is *Quasi-Resilient-to-Internal-Observation*: malware running on a single machine would be insufficient to break the scheme if more than one trustee must vouch for the user. The scheme is *Quasi-Resilient-to-Leaks-from-Other-Verifiers*: the identity of the trusted friend can be leaked, but is difficult for an attacker to exploit if the protocol is followed. The scheme is *Resilient-to-Phishing*: it would appear hard to lure the user into invoking the backup mechanism and then deliver a valid vouchcode to a simulated site. It is *Resilient-to-Theft*: there's nothing to be stolen. It is clearly not *No-Trusted-Third-Party*: the friend must be trusted, not merely not to abuse the responsibility, but to follow the protocol when contacted. The scheme is *Requiring-Explicit-Consent*. It is *Quasi-Unlinkable*: a common trustee suggests, but does not prove that a user has accounts at both sites.

V. Discussion

A clear result of our exercise is that no scheme we examined is perfect—or even comes close to perfect scores. The incumbent (traditional passwords) achieves all benefits on deployability, and one scheme (the CAP reader) achieves all in security, but no scheme achieves all usability benefits. Not a single scheme is dominant over passwords, i.e., does better on one or more benefits and does at least as well on all others. Almost all schemes do better than passwords in some criteria, but all are worse in others: as Table I shows, no row is free of red (horizontal) stripes.

Thus, the current state of the world is a Pareto equilibrium. Replacing passwords with any of the schemes examined is not a question of giving up an inferior technology for something unarguably better, but of giving up one set of compromises and trade-offs in exchange for another. For example, arguing that a hardware token like RSA SecurID is better than passwords implicitly assumes that the security criteria where it does better outweigh the usability and deployability criteria where it does worse. For accounts that require high assurance, security benefits may indeed outweigh the fact that the scheme doesn't offer *Nothing-to-Carry* nor *Negligible-Cost-per-User*, but this argument is less compelling for lower value accounts.

The usability benefits where passwords excel—namely, *Nothing-to-Carry*, *Efficient-to-Use*, *Easy-Recovery-from-Loss*—are where essentially all of the stronger security schemes need improvement. None of the paper token or hardware token schemes achieves even two of these three. In expressing frustration with the continuing dominance of passwords, many security experts presumably view these two classes of schemes to be sufficiently usable to justify a switch from passwords. The web sites that crave user traffic apparently disagree.

Some sets of benefits appear almost incompatible, e.g., the pair (*Memorywise-Effortless*, *Nothing-to-Carry*) is achieved only by biometric schemes. No schemes studied achieve (*Memorywise-Effortless*, *Resilient-to-Theft*) fully, nor (*Server-Compatible*, *Resilient-to-Internal-Observation*) or (*Server-Compatible*, *Resilient-to-Leaks-from-Other-Verifiers*), though several almost do. Note that since compatibility with existing servers almost assures a static replayable secret, to avoid its security implications, many proposals abandon being *Server-Compatible*.

A. Rating categories of schemes

Password managers offer advantages over legacy passwords in selected usability and security aspects without losing much. They could become a staple of users' coping strategies if passwords remain widespread, enabling as a major advantage the management of an ever-increasing number of accounts (*Scalable-for-Users*). However, the underlying technology remains replayable, static (mainly user-chosen) passwords.

Federated schemes are particularly hard to grade. Proponents note that security is good if authentication to the identity provider (IP) is done with a strong scheme (e.g., one-time passwords or tokens). However in this case usability is inherited from that scheme and is generally poor, per Table I. This also reduces federated schemes to be a placeholder for a solution rather than a solution itself. If authentication to the IP relies on passwords, then the resulting security is only a little better than

Category	Scheme	Described in section	Reference	Usability							Deployability						Security										
				Memorywise-Effortless	*Scalable-for-Users*	*Nothing-to-Carry*	*Physically-Effortless*	*Easy-to-Learn*	*Efficient-to-Use*	*Infrequent-Errors*	*Easy-Recovery-from-Loss*	*Accessible*	*Negligible-Cost-per-User*	*Server-Compatible*	*Browser-Compatible*	*Mature*	*Non-Proprietary*	*Resilient-to-Physical-Observation*	*Resilient-to-Targeted-Impersonation*	*Resilient-to-Throttled-Guessing*	*Resilient-to-Unthrottled-Guessing*	*Resilient-to-Internal-Observation*	*Resilient-to-Leaks-from-Other-Verifiers*	*Resilient-to-Phishing*	*Resilient-to-Theft*	*No-Trusted-Third-Party*	*Requiring-Explicit-Consent*	*Unlinkable*
(Incumbent)	Web passwords	III	[13]			●		●	●	○	●	●	●	●	●	●	●		○						●	●	●	●
Password managers	Firefox	IV-A1	[22]	○	●	○	○	●	●	●		●	●	●		●	●	○	○					●	●	●	●	●
	LastPass	IV-A2	[23]	○	●	○	○	●	●	●	○	●	○	●		●		○	○				○	●	●		●	●
Proxy	URRSA	IV-B1	[5]	●				●		○			●	○	●				○			○		●			●	●
	Impostor	IV-B2	[25]	○	●	●		●			●	●	●	●	○		●	●	○			○		●	●			●
Federated	OpenID	IV-C1	[29]	○	●	●	○	○	●	●	●	●	●		●	●	●	○	○	○	○		●		●		●	
	Microsoft Passport	IV-C2	[33]	○	●	●	○	●	●	●	●	●			●	●		○	○	○	○		●		●		●	
	Facebook Connect	IV-C3	[35]	○	●	●	○	●	●	●	●	●	●		●	●		○	○	○	○		●		●			
	BrowserID	IV-C4	[37]	○	●	●	○	●	●	●	●	●	●		○	○	●	○	○	○	○		●		●		●	
	OTP over email	IV-C5	[41]	○	●	●		●		●	●	●	●		●		●	○	○	○	○		●	●	●		●	
Graphical	PCCP	IV-D1	[7]			●		●	○	○	●		●		●		●		●	○			●	●	●	●	●	●
	PassGo	IV-D2	[100]			●		●	○	○	●		●		●	○	●		●						●	●	●	●
Cognitive	GrIDsure (original)	IV-E1	[51]			●		●	○	○	●		●		●				●						●	●	●	●
	Weinshall	IV-E2	[52]			●							●		●		●	○	●				●	●	●	●	●	●
	Hopper Blum	IV-E3	[54]			●						●	●		●		●	○	●				●	●	●	●	●	●
	Word Association	IV-E4	[55]			●		●	●	○	○	●	●		●		●								●	●	●	●
Paper tokens	OTPW	IV-F1	[60]					●			●		●		●	●	●		●	●	●	●	●	●	●	●	●	●
	S/KEY	IV-F2	[59]	●				●		○	●		●		●	●	●		●	●	●	●	●	○	●	●	●	●
	PIN+TAN	IV-F3	[62]					●		○	○		○		●	●	●		●	●	●	●	●	●	○	●	●	●
Visual crypto	PassWindow	IV-G1	[67]	●									○		●	●	●	○	●	●	●	○	●	●		●	●	●
Hardware tokens	RSA SecurID	IV-H1	[69]					●	○	○					●	●	●	●	●	●	●	●	●	●	●		●	●
	YubiKey	IV-H2	[71]					●	○	○		●			●	●	●	●	●	●	●	●	●	●	●		●	●
	IronKey	IV-H3	[73]	○	●		○	○	○	○		●		●	●	●	●	○		○		●	●	●	●	●	●	●
	CAP reader	IV-H4	[74]					●	○	○					●	●	●	●	●	●	●	●	●	●	●	●	●	●
	Pico	IV-H5	[8]	●	●		●		○	○							●	●	●	●	●	●	●	○	●	●	●	●
Phone-based	Phoolproof	IV-I1	[78]			○		●	○	○		○	○	○			●	●	●	●	○	●	●	●	●	●	●	●
	Cronto	IV-I2	[79]			○		●	○	○		○	○		●	●	●	●	●	●	○	●	●	●	●	●	●	●
	MP-Auth	IV-I3	[6]			○		●	○		○	○	○				●	○					●	●	●	●	●	●
	OTP over SMS	IV-I4		●	●	○		●		○	○	○			●	●	●	●	●	●	○	●	●	○		●	●	●
	Google 2-Step	IV-I5	[81]			○		●	○	○	○	○			●	●	●	○	○	●		●	●		●	●	●	●
Biometric	Fingerprint	IV-J1	[83]	●	●	●	○	●	○			○				○	●			●						●	●	
	Iris	IV-J2	[84]	●	●	●	○	●	○			○				○	●			●						●	○	
	Voice	IV-J3	[85]	●	●	●	○	●	○			○	○		○	○	●			○						●	●	
Recovery	Personal knowledge	IV-K1	[91]	○		●		●	●	○	●	●	●		●	●	●								●	●	●	●
	Preference-based	IV-K2	[56]	○		●		●	○	●	○	●	●		●			○						●	●	●	●	●
	Social re-auth.	IV-K3	[99]			●		●		●		○	●		●	○	○		●	●	○	○	●	●	●		●	○

●= offers the benefit; ○= almost offers the benefit; *no circle* = does not offer the benefit.
||||= better than passwords; ||||= worse than passwords; *no background pattern* = no change.
We group related schemes into categories. For space reasons, the peer-reviewed paper [1] describes at most one scheme per category, but this tech report discusses them all.

TABLE I
COMPARATIVE EVALUATION OF THE VARIOUS SCHEMES WE EXAMINED

that of passwords themselves (with fewer password entry instances exposed to attack).

Graphical passwords can approach text passwords on usability criteria, offering some security gain, but static secrets are replayable and not *Resilient-to-Internal-Observation*. Despite adoption for device access-control on some touch-screen mobile devices, for remote web authentication the advantages appear insufficient to generally displace a firmly-entrenched incumbent.

Cognitive schemes show slender improvement on the security of passwords, in return for worse usability. While several schemes attempt to achieve *Resilient-to-Internal-Observation*, to date none succeed: the secret may withstand one observation or two [101], but seldom more than a handful [53]. The apparently inherent limitations [102], [103] of cognitive schemes to date lead one to question if the category can rise above one of purely academic interest.

The hardware token, paper token and phone-based categories of schemes fare very well in security, e.g., most in Table I are *Resilient-to-Internal-Observation*, easily beating other classes. However, that S/KEY and SecurID have been around for decades and have failed to slow down the inexorable rise of passwords suggests that their drawbacks in usability (e.g., not *Scalable-for-Users*, nor *Nothing-to-Carry*, nor *Efficient-to-Use*) and deployability (e.g., hardware tokens are not *Negligible-Cost-per-User*) should not be over-looked. Less usable schemes can always be mandated, but this is more common in situations where a site has a de facto monopoly (e.g., employee accounts or government sites) than where user acceptance matters. Experience shows that the large web-sites that compete for both traffic and users are reluctant to risk bad usability [16]. Schemes that are less usable than passwords face an uphill battle in such environments.

Biometric schemes have mixed scores on our usability metrics, and do poorly in deployability and security. As a major issue, physical biometrics being inherently non-*Resilient-to-Internal-Observation* is seriously compounded by biometrics missing *Easy-Recovery-from-Loss* as well, with re-issuance impossible [9]. Thus, e.g., if malware captures the digital representation of a user's iris, possible replay makes the biometric no longer suitable in unsupervised environments. Hence despite security features appropriate to control access to physical locations under the supervision of suitable personnel, biometrics aren't well suited for unsupervised web authentication where client devices lack a trusted input path and means to verify that samples are live.

B. Extending the benefits list

Our list of benefits is not complete, and indeed, any such list could always be expanded. We did not include resistance to active-man-in-the-middle, which a few examined schemes may provide, or to relay attacks, which probably none of them do. However, tracking all security goals, whether met or not, is important and considering benefits that indicate resistance to these (and additional) attacks is worthwhile.

Continuous authentication (with ongoing assurances rather than just at session start, thereby addressing session hijacking) is a benefit worth considering, although a goal of few current schemes. Positive user affectation (how pleasant users perceive use of a scheme to be) is a standard usability metric we omitted; unfortunately, the literature currently lacks this information for most schemes. The burden on the end-user in migrating from passwords (distinct from the deployability costs of modifying browser and server infrastructure) is another important cost—both the one-time initial setup and per-account transition costs. While ease of resetting and revoking credentials falls within *Easy-Recovery-from-Loss*, the benefit does not include user and system aspects related to ease of renewing credentials that expire within normal operations (excluding loss). Other missing cost-related benefits are low cost for initial setup (including infrastructure changes by all stakeholders); low cost for ongoing administration, support and maintenance; and low overall complexity (how many inter-related "moving parts" a system has). We don't capture continued availability under denial-of-service attack, ease of use on mobile devices, nor the broad category of economic and business effects—e.g., the lack of incentive to be a relying party is cited as a main reason for OpenID's lack of adoption [30].

We have not attempted to capture these and other benefits in the present paper, though all fit into the framework and could be chosen by others using this methodology. Alas, many of these raise a difficulty: assigning ratings might be even more subjective than for existing benefits.

C. Additional nuanced ratings

We considered, but did not use, a "fatal" rating to indicate that a scheme's performance on a benefit is so poor that the scheme should be eliminated from serious consideration. For example, the 2–3 minutes required for authentication using the Weinshall or Hopper-Blum schemes may make them "fatally-non-*Efficient-to-Use*", likely preventing widespread adoption even if virtually all other benefits were provided. We decided against this because for many properties, it isn't clear what level of failure to declare as fatal.

We also considered a "power" rating to indicate that a scheme optionally enables a benefit for power users—e.g., OpenID could be rated "amenable-to-*No-Trusted-Third-Party*" as users can run their own identity servers, in contrast to Facebook Connect or Microsoft Passport. The popularity of webmail-based password reset indicates most users accede to a heavily-trusted third party for their online identities already, so "amenable-to" may suffice for adoption. OpenID is arguably amenable to every security benefit for power users, but doesn't provide them for common users who use text passwords to authenticate to their identity provider. However, as one could argue for an amenable-to rating for many properties of many schemes, we maintained focus on properties provided by default to all users.

D. Weights and finer-grained scoring

We reiterate a caution sounded at the end of Section II: the benefits chosen as metrics are not all of equal weight. The importance of any particular benefit depends on

target use and threat environment. While one could assign weights to each column to compute numerical scores for each scheme, providing exact weights is problematic and no fixed values would suit all scenarios; nonetheless, our framework allows such an endeavour. For finer-grained evaluation, table cell scores like *partially* could also be allowed beyond our very coarse {*no*, *almost*, *yes*} quantization, to further delineate similar schemes. This has merit but brings the danger of being "precisely wrong", and too fine a granularity adds to the difficulty of scoring schemes consistently. There will be the temptation to be unrealistically precise ("If scheme X gets 0.9 for this benefit, then scheme Y should get at most 0.6"), but this demands the ability to maintain a constant level of precision *repeatably* across all cells.

We have resisted the temptation to produce an aggregate score for each scheme (e.g., by counting the number of benefits achieved), or to rank the schemes. As discussed above, fatal failure of a single benefit or combined failure of a pair of benefits (e.g., not being *Resilient-to-Internal-Observation* and fatally failing *Easy-Recovery-from-Loss* for biometrics) may eliminate a scheme from consideration. Thus, seeking schemes purely based on high numbers of benefits could well prove but a distraction.

Beyond divergences of judgement, there will no doubt be errors in judgement in scoring. The table scoring methodology must include redundancy and cross-checks sufficient to catch most such errors. (Our exercise involved one author initially scoring a scheme row, co-authors verifying the scores, and independently, cross-checks within columns to calibrate individual benefit ratings across schemes; useful clarifications of benefit definitions often resulted.) Another danger in being "too precise" arises from scoring on second-hand data inferred from papers. Coarsely-quantized but self-consistent scores are likely better than inconsistent ones.

On one hand, it could be argued that different application domains (e.g., banking vs. gaming) have different requirements and that therefore they ought to assign different weights to the benefits, resulting in a different choice of optimal scheme for each domain. However on the other hand, to users, a proliferation of schemes is in itself a failure: the meta-scheme of "use the best scheme for each application" will score rather poorly on *Scalable-for-Users*, *Easy-to-Learn* and perhaps a few other usability benefits.

E. Combining schemes

Pairs of schemes that complement each other well in a two-factor arrangement might be those where *both* achieve good scores in usability and deployability and *at least one* does so in security—so a combined scheme might be viewed as having the AND of the usability-deployability scores (i.e., the combination does not have a particular usability or deployability benefit unless both of the schemes do) and the OR of the security scores (i.e., the combination has the security benefit if either of the schemes do). An exception would appear to be the usability benefit *Scalable-for-Users* which a combination might inherit from either component.

However, this is necessarily just a starting point for the analysis: it is optimistic to assume that two-component schemes always inherit benefits in this way. Wimberly and Liebrock [104] observed that the presence of a second factor caused users to pick much weaker passwords than if passwords alone were used to protect an account—as predicted by Adams's "risk thermostat" model [105]. Thus, especially where user choice is involved, there can be an erosion of the efficacy of one protection when a second factor is known to be in place. Equally, defeating one security mechanism may also make it materially easier to defeat another. We rated, e.g., Phoolproof *Quasi-Resilient-to-Internal-Observation* because it requires an attacker to compromise both a PC and a mobile device. However, malware has already been observed in the wild which leverages a compromised PC to download further malware onto mobile devices plugged into the PC for a software update [106].

See O'Gorman [9] for suggested two-factor combinations of biometrics, passwords, and tokens, for various applications (e.g., combining a hardware token with a biometric). Another common suggestion is pairing a federated scheme with a higher-security scheme, e.g., a hardware token.

VI. Concluding remarks

The concise overview offered by Table I allows us to see high level patterns that might otherwise be missed. We could at this stage draw a variety of conclusions and note, for example, that graphical and cognitive schemes offer only minor improvements over passwords and thus have little hope of displacing them. Or we could note that most of the schemes with substantial improvements in both usability and security can be seen as incarnations of Single-Sign-On (including in this broad definition not only federated schemes but also "local SSO" systems [28] such as password managers or Pico). Having said that, we expect the long-term scientific value of our contribution will lie not as much in the raw data distilled herein, as in the methodology by which it was assembled. A carefully crafted benefits list and coherent methodology for scoring table entries, despite inevitable (albeit instructive) disagreements over fine points of specific scores, allows principled discussions about high level conclusions.

That a Table I scheme (the CAP reader) scored full marks in security does not at all suggest that its real-world security is perfect—indeed, major issues have been found [74]. This is a loud warning that it would be unwise to read absolute verdicts into these scores. Our ratings are useful and we stand by them, but they are not a substitute for independent critical analysis or for considering aspects we didn't rate, such as vulnerability to active man-in-the-middle attacks.

We note that the ratings implied by scheme authors in original publications are often not only optimistic, but also incomplete. Proponents, perhaps subconsciously, often have a biased and narrow view of what benefits are relevant. Our framework allows a more objective assessment.

In closing we observe that, looking at the green (vertical) and red (horizontal) patterns in Table I, most schemes

do better than passwords on security—as expected, given that inventors of alternatives to passwords tend to come from the security community. Some schemes do better and some worse on usability—suggesting that the community needs to work harder there. But *every* scheme does worse than passwords on deployability. This was to be expected given that the first four deployability benefits are defined with explicit reference to what passwords achieve and the remaining two are natural benefits of a long-term incumbent, but this uneven playing field reflects the reality of a decentralized system like the Internet. Marginal gains are often not sufficient to reach the activation energy necessary to overcome significant transition costs, which may provide the best explanation of why we are likely to live considerably longer before seeing the funeral procession for passwords arrive at the cemetery.

Acknowledgments

The authors thank the anonymous reviewers whose comments helped improve the paper greatly. Joseph Bonneau is supported by the Gates Cambridge Trust. Paul C. van Oorschot is Canada Research Chair in Authentication and Computer Security, and acknowledges NSERC for funding the chair and a Discovery Grant; partial funding from NSERC ISSNet is also acknowledged. This work grew out of the Related Work section of Pico [8].

References

[1] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano, "The quest to replace passwords: A framework for comparative evaluation of web authentication schemes," in *Proc. IEEE Symp. on Security and Privacy*, 2012.

[2] R. Morris and K. Thompson, "Password security: a case history," *Commun. ACM*, vol. 22, no. 11, pp. 594–597, 1979.

[3] A. Adams and M. Sasse, "Users Are Not The Enemy," *Commun. ACM*, vol. 42, no. 12, pp. 41–46, 1999.

[4] C. Herley and P. C. van Oorschot, "A research agenda acknowledging the persistence of passwords," *IEEE Security & Privacy*, vol. 10, no. 1, pp. 28–36, 2012.

[5] D. Florêncio and C. Herley, "One-Time Password Access to Any Server Without Changing the Server," *ISC 2008, Taipei.*

[6] M. Mannan and P. C. van Oorschot, "Leveraging personal devices for stronger password authentication from untrusted computers," *Journal of Computer Security*, vol. 19, no. 4, pp. 703–750, 2011.

[7] S. Chiasson, E. Stobert, A. Forget, R. Biddle, and P. C. van Oorschot, "Persuasive cued click-points: Design, implementation, and evaluation of a knowledge-based authentication mechanism," *IEEE Trans. on Dependable and Secure Computing*, vol. 9, no. 2, pp. 222–235, 2012.

[8] F. Stajano, "Pico: No more passwords!" in *Proc. Sec. Protocols Workshop 2011*, ser. LNCS, vol. 7114. Springer.

[9] L. O'Gorman, "Comparing passwords, tokens, and biometrics for user authentication," *Proceedings of the IEEE*, vol. 91, no. 12, pp. 2019–2040, December 2003.

[10] K. Renaud, "Quantification of authentication mechanisms: a usability perspective," *J. Web Eng.*, vol. 3, no. 2, pp. 95–123, 2004.

[11] R. Biddle, S. Chiasson, and P. C. van Oorschot, "Graphical Passwords: Learning from the First Twelve Years," *ACM Computing Surveys*, vol. 44, no. 4, 2012.

[12] J. Nielsen and R. Mack, *Usability Inspection Methods*. John Wiley & Sons, Inc, 1994.

[13] J. Bonneau and S. Preibusch, "The password thicket: technical and market failures in human authentication on the web," in *Proc. WEIS 2010*, 2010.

[14] J. Bonneau, "The science of guessing: analyzing an anonymized corpus of 70 million passwords," *IEEE Symp. Security and Privacy*, May 2012.

[15] K. Fu, E. Sit, K. Smith, and N. Feamster, "Dos and don'ts of client authentication on the web," in *Proc. USENIX Security Symposium*, 2001.

[16] D. Florêncio and C. Herley, "Where Do Security Policies Come From?" in *ACM SOUPS 2010: Proc. 6th Symp. on Usable Privacy and Security*.

[17] L. Falk, A. Prakash, and K. Borders, "Analyzing websites for user-visible security design flaws," in *ACM SOUPS 2008*, pp. 117–126.

[18] S. Gaw and E. W. Felten, "Password Management Strategies for Online Accounts," in *ACM SOUPS 2006: Proc. 2nd Symp. on Usable Privacy and Security*, pp. 44–55.

[19] D. Florêncio and C. Herley, "A large-scale study of web password habits," in *WWW '07: Proc. 16th International Conf. on the World Wide Web*. ACM, 2007, pp. 657–666.

[20] D. Balzarotti, M. Cova, and G. Vigna, "ClearShot: Eavesdropping on Keyboard Input from Video," in *IEEE Symp. Security and Privacy*, 2008, pp. 170–183.

[21] B. Kaliski, *RFC 2898: PKCS #5: Password-Based Cryptography Specification Version 2.0*, IETF, September 2000.

[22] Mozilla Firefox, ver. 10.0.2, www.mozilla.org/.

[23] LastPass, www.lastpass.com/.

[24] J. Aron, "Data leak in service that stores your passwords," May 2011, www.newscientist.com/blogs/onepercent/2011/05/password-manager-lastpass-suff.html.

[25] A. Pashalidis and C. J. Mitchell, "Impostor: A single sign-on system for use from untrusted devices," *Proc. IEEE Globecom*, 2004.

[26] R. M. Needham and M. D. Schroeder, "Using encryption for authentication in large networks of computers," *Commun. ACM*, vol. 21, pp. 993–999, December 1978.

[27] J. Kohl and C. Neuman, "The Kerberos Network Authentication Service (V5)," United States, 1993.

[28] A. Pashalidis and C. J. Mitchell, "A Taxonomy of Single Sign-On Systems," in *Proc. ACISP 2003, Information Security and Privacy, 8th Australasian Conference*. Springer LNCS 2727, 2003, pp. 249–264.

[29] D. Recordon and D. Reed, "OpenID 2.0: a platform for user-centric identity management," in *DIM '06: Proc. 2nd ACM Workshop on Digital Identity Management*, 2006, pp. 11–16.

[30] S.-T. Sun, Y. Boshmaf, K. Hawkey, and K. Beznosov, "A billion keys, but few locks: the crisis of web single sign-on," *Proc. NSPW 2010*, pp. 61–72.

[31] B. Laurie, "OpenID: Phishing Heaven," January 2007, www.links.org/?p=187.

[32] "Microsoft Passport," Feb 2011, www.passport.net.

[33] D. P. Kormann and A. D. Rubin, "Risks of the Passport single signon protocol," *Computer Networks*, vol. 33, no. 1–6, 2000.

[34] R. Oppliger, "Microsoft .NET Passport: A Security Analysis," *Computer*, vol. 36, no. 7, pp. 29–35, 2003.

[35] "Facebook Connect," 2011, www.facebook.com/advertising/?connect.

[36] D. Recordon and D. Hardt, "The OAuth 2.0 Protocol," April 2010, tools.ietf.org/html/draft-hammer-oauth2-00.

[37] M. Hanson, D. Mills, and B. Adida, "Federated Browser-Based Identity using Email Addresses," *W3C Workshop on Identity in the Browser*, May 2011.

[38] B. Adida, "EmID: Web authentication by email address," *Proceedings of Web 2.0 Security and Privacy Workshop*, 2008.

[39] C. Karlof, J. D. Tygar, and D. Wagner, "Conditioned-safe ceremonies and a user study of an application to web authentication," in *ACM SOUPS 2009: Proc. 5th Symposium on Usable Privacy and Security*.

[40] S. L. Garfinkel, "Email-Based Identification and Authentication: An Alternative to PKI?" *IEEE Security and Privacy*, vol. 1, no. 6, pp. 20–26, 2003.

[41] T. W. van der Horst and K. E. Seamons, "Simple Authentication for the Web," in *Intl. Conf. on Security and Privacy in Communications Networks*, 2007, pp. 473–482.

[42] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, and J. M. Smith, "Smudge Attacks on Smartphone Touch Screens," in *USENIX 4th Workshop on Offensive Technologies*, 2010.

[43] H. Tao and C. Adams, "Pass-Go: A Proposal to Improve the Usability of Graphical Passwords," *International Journal of Network Security*, vol. 7, no. 2, pp. 273–292, 2008.

[44] P. C. van Oorschot and J. Thorpe, "On Predictive Models and User-Drawn Graphical Passwords," *ACM Transactions on Information and System Security*, vol. 10, no. 4, pp. 1–33, 2008.

[45] I. Jermyn, A. Mayer, F. Monrose, M. Reiter, and A. Rubin, "The Design and Analysis of Graphical Passwords," in *8th USENIX Security Symposium*, August 1999.

[46] D. Davis, F. Monrose, and M. Reiter, "On User Choice in Graphical Password Schemes," in *13th USENIX Security Symposium*, 2004.

[47] M. Hlywa, A. Patrick, and R. Biddle, "Facing the Facts about Image Type in Recognition-Based Graphical Passwords," in *Annual Computer Security Applications Conf. (ACSAC)*, 2011.

[48] K. Bicakci, N. B. Atalay, M. Yuceel, H. Gurbaslar, and B. Erdeniz, "Towards Usable Solutions to Graphical Password Hotspot Problem," in *33rd Annual IEEE International Computer Software and Applications Conference*, 2009.

[49] R. Dhamija and A. Perrig, "Déjà Vu: A User Study Using Images for Authentication," in *9th USENIX Security Symposium*, 2000.

[50] P. Dunphy, A. P. Heiner, and N. Asokan, "A Closer Look at Recognition-based Graphical Passwords on Mobile Devices," in *Sixth ACM Symposium on Usable Privacy and Security (SOUPS), Redmond, Washington*, July 2010.

[51] R. Jhawar, P. Inglesant, N. Courtois, and M. A. Sasse, "Make mine a quadruple: Strengthening the security of graphical one-time pin authentication," in *Proc. NSS 2011*, pp. 81–88.

[52] D. Weinshall, "Cognitive Authentication Schemes Safe Against Spyware (Short Paper)," in *IEEE Symposium on Security and Privacy*, May 2006.

[53] P. Golle and D. Wagner, "Cryptanalysis of a Cognitive Authentication Scheme," *IEEE Symp. Security and Privacy*, 2007.

[54] N. Hopper and M. Blum, "Secure human identification protocols," *ASIACRYPT 2001*, pp. 52–66, 2001.

[55] S. Smith, "Authenticating users by word association," *Computers & Security*, vol. 6, no. 6, pp. 464–470, 1987.

[56] M. Jakobsson, L. Yang, and S. Wetzel, "Quantifying the Security of Preference-based Authentication," in *ACM DIM 2008: 4th Workshop on Digital Identity Management*.

[57] W. J. Haga and M. Zviran, "Question-and-answer passwords: an empirical evaluation," *Inf. Syst.*, vol. 16, no. 3, pp. 335–343, 1991.

[58] L. Lamport, "Password authentication with insecure communication," *Commun. ACM*, vol. 24, no. 11, pp. 770–772, 1981.

[59] N. Haller and C. Metz, "RFC 1938: A One-Time Password System," 1998.

[60] M. Kuhn, "OTPW — a one-time password login package," 1998, www.cl.cam.ac.uk/~mgk25/otpw.html.

[61] N. Haller, C. Metz, P. Nesser, and M. Straw, "RFC 2289: A One-Time Password System," 1998.

[62] A. Wiesmaier, M. Fischer, E. G. Karatsiolis, and M. Lippert, "Outflanking and securely using the PIN/TAN-System," *CoRR*, vol. cs.CR/0410025, 2004.

[63] J. Bonneau, S. Preibusch, and R. Anderson, "A birthday present every eleven wallets? The security of customer-chosen banking PINs," *FC '12: Proceedings of the the Sixteenth International Conference on Financial Cryptography*, March 2012.

[64] M. Naor and A. Shamir, "Visual cryptography," in *EUROCRYPT'94*, ser. LNCS 950 (Springer, 1995), pp. 1–12.

[65] K. Kobara and H. Imai, "Limiting the Visible Space Visual Secret Sharing Schemes and Their Application to Human Identification," in *ASIACRYPT 1996*, pp. 185–195.

[66] M. Naor and B. Pinkas, "Visual authentication and identification," in *CRYPTO '97*. Springer LNCS 1294, 1997, pp. 322–336.

[67] "PassWindow," 2011, www.passwindow.com.

[68] S. Nettle, S. O'Neil, and P. Lock, *PassWindow: A New Solution to Providing Second Factor Authentication*. VEST Corporation, 2009.

[69] RSA, "RSA SecurID Two-factor Authentication," 2011, www.rsa.com/products/securid/sb/10695_SIDTFA_SB_0210.pdf.

[70] P. Bright, "RSA finally comes clean: SecurID is compromised," Jun. 2011, arstechnica.com/security/news/2011/06/rsa-finally-comes-clean-securid-is-compromised.ars.

[71] Yubico, "The YubiKey Manual, v. 2.0," 2009, static.yubico.com/var/uploads/YubiKey_manual-2.0.pdf.

[72] D. M'Raihi, M. Bellare, F. Hoornaert, D. Naccache, and O. Ranen, "HOTP: An HMAC-Based One-Time Password Algorithm," IETF, RFC 4226, Dec. 2005, www.ietf.org/rfc/rfc4226.txt.

[73] Ironkey, www.ironkey.com/internet-authentication.

[74] S. Drimer, S. J. Murdoch, and R. Anderson, "Optimised to Fail: Card Readers for Online Banking," in *Financial Cryptography and Data Security*, 2009, pp. 184–200.

[75] S. J. Murdoch, S. Drimer, R. Anderson, and M. Bond, "Chip and PIN is Broken," in *IEEE Symp. Security and Privacy*, 2010, pp. 433–446.

[76] F.-L. Wong and F. Stajano, "Multi-channel Protocols," in *13th International Security Protocols Workshop, Cambridge, UK (LNCS 4631)*. Springer, 2005, pp. 112–127.

[77] B. Laurie and A. Singer, "Choose the red pill and the blue pill: a position paper," in *Proc. New Security Paradigms Workshop 2008*. ACM, 2008, pp. 127–133.

[78] B. Parno, C. Kuo, and A. Perrig, "Phoolproof Phishing Prevention," in *Proc. Fin. Crypt. 2006*, pp. 1–19.

[79] Cronto, www.cronto.com/.

[80] P. van Oorschot, "Message Authentication by Integrity with Public Corroboration," in *New Security Paradigms Workshop (NSPW), Lake Arrowhead, California*, September 2005.

[81] Google Inc., "2-step verification: how it works," 2012, www.google.com/accounts.

[82] A. K. Jain, A. Ross, and S. Pankanti, "Biometrics: a tool for information security," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 2, pp. 125–143, 2006.

[83] A. Ross, J. Shah, and A. K. Jain, "From Template to Image: Reconstructing Fingerprints from Minutiae Points," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 4, pp. 544–560, 2007.

[84] J. Daugman, "How iris recognition works," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 14, no. 1, pp. 21–30, 2004.

[85] P. S. Aleksic and A. K. Katsaggelos, "Audio-Visual Biometrics," *Proc. of the IEEE*, vol. 94, no. 11, pp. 2025–2044, 2006.

[86] T. Matsumoto, H. Matsumoto, K. Yamada, and S. Hoshino, "Impact of artificial "gummy" fingers on fingerprint systems," in *SPIE Conf. Series*, vol. 4677, Apr. 2002, pp. 275–289.

[87] J. Daugman, "New Methods in Iris Recognition," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 37, no. 5, pp. 1167–1175, 2007.

[88] L. Ballard, F. Monrose, and D. Lopresti, "Biometric authentication revisited: Understanding the impact of wolves in sheep's clothing," in *Proc. 15th Annual Usenix Security Symp.*, 2006, pp. 29–41.

[89] M. Zviran and W. J. Haga, "A Comparison of Password Techniques for Multilevel Authentication Mechanisms," *Comput. J.*, vol. 36, no. 3, pp. 227–237, 1993.

[90] M. Just and D. Aspinall, "Personal Choice and Challenge Questions: A Security and Usability Assessment," in *SOUPS '09: Proceedings of the Fifth Symposium on Usable Privacy and Security*, 2009.

[91] S. Schechter, A. J. B. Brush, and S. Egelman, "It's no secret: Measuring the security and reliability of authentication via 'secret'

questions," in *IEEE Symp. Security and Privacy*, 2009, pp. 375–390.

[92] R. Pond, J. Podd, J. Bunnell, and R. Henderson, "Word Association Computer Passwords: The Effect of Formulation Techniques on Recall and Guessing Rates," *Computers & Security*, vol. 19, no. 7, pp. 645–656, 2000.

[93] M. Just and D. Aspinall, "Challenging Challenge Questions," presented at Trust 2009: International Conference on the Technical and Socio-Economic Aspects of Trusted Computing. Invited for submission to Policy and Internet Journal.

[94] A. Rabkin, "Personal knowledge questions for fallback authentication: Security questions in the era of Facebook," in *ACM SOUPS 2008: Proc. 4th Symposium On Usable Privacy and Security*.

[95] V. Griffith and M. Jakobsson, "Messin' with Texas: Deriving Mother's Maiden Names Using Public Records," *Applied Cryptography and Network Security*, 2005.

[96] J. Lindamood and M. Kantarcioglu, "Inferring Private Information Using Social Network Data," University of Texas at Dallas Computer Science Department, Tech. Rep. UTDCS-21-08, July 2008.

[97] J. Bonneau, M. Just, and G. Matthews, "What's in a Name: Evaluating Statistical Attacks Against Personal Knowledge Questions," *FC 2010: Proc. 14th International Conference on Financial Cryptography and Data Security*.

[98] M. Jakobsson, E. Stolterman, S. Wetzel, and L. Yang, "Love and authentication," in *CHI*, 2008, pp. 197–200.

[99] J. Brainard, A. Juels, R. L. Rivest, M. Szydlo, and M. Yung, "Fourth-factor authentication: somebody you know," in *ACM CCS 2006*, pp. 168–178.

[100] H. Tao, "Pass-Go, a New Graphical Password Scheme," Master's thesis, School of Information Technology and Engineering, University of Ottawa, June 2006.

[101] D. Weinshall, "Cognitive Authentication Schemes Safe Against Spyware," *IEEE Symp. Security and Privacy*, 2006.

[102] B. Coskun and C. Herley, "Can "Something You Know" be Saved?" *ISC 2008, Taipei.*

[103] Q. Yan, J. Han, Y. Li, and H. Deng, "On limitations of designing usable leakage-resilient password systems: Attacks, principles and usability." Proc. NDSS, 2012.

[104] H. Wimberly and L. M. Liebrock, "Using Fingerprint Authentication to Reduce System Security: An Empirical Study," in *IEEE Symp. Security and Privacy*, 2011, pp. 32–46.

[105] J. Adams, "Risk and morality: three framing devices," in *Risk and Morality*, R. Ericson and A. Doyle, Eds. University of Toronto Press, 2003.

[106] A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, "A survey of mobile malware in the wild," in *ACM SPSM 2011: 1st Workshop on Security and Privacy in Smartphones and Mobile Devices*, pp. 3–14.