Designing a User-Experience-First, Privacy-Respectful, high-security mutual-multifactor authentication solution

Chris Drake

CryptoPhoto
Suite 2, 3, 4; 1 Eugarie St;
Noosa Heads, QLD 4567
Chris.Drake@CryptoPhoto.com

Dr. Praveen Gauravaram

Tata Consultancy Services Limited
22/69 Ann street, Brisbane
Queensland, Australia
p.gauravaram@tcs.com

ABSTRACT

The rush for improved security, particularly in banking, presents a frightening erosion of privacy. As fraud and theft rise, anti-fraud techniques subject user privacy, identity, and activity to ever-increasing risks. Techniques like behavioral analytics, biometric data exchange, persistent device identifiers, GPS/geo-fencing, knowledge-based authentication, on-line user activity tracking, social mapping and browser fingerprinting secretly share, profile, and feed sensitive user data into backend anti-fraud systems. This is usually invisible, and usually without user consent or awareness. It is also, unfortunately, necessary, partly because contemporary authentication is increasingly ineffective against modern attacks, but mostly because the idea of "usable" is confused with "invisible" most of the time. In the mind of a CISO, "stronger authentication" means a slower, less convenient, and more complicated experience for the user. Security and privacy tend to lose most battles against usability, particularly when friction impacts customer adoption or increases support costs.

This paper presents an innovative mutual authentication solution showing that it is possible to improve both security and user experience at the same time. The striking feature of our solution is that the human is inherently involved during the entire authentication and verification process. By making authentication broadly effective for ordinary users, our solution improves user privacy by obsoleting the need for invasive anti-fraud. We outline further improvements to user privacy made possible through adoption of our authentication methods within identity management solutions.

Our mutual authentication solution contributes a range of novel techniques addressing traditionally "out of scope" topics, including (but not limited to) unmotivated and unsophisticated users, social-engineering against both users and staff, phishing, malware, server-side break-ins, MitM, DoS, bots, enrollment over compromised channels, certificate-substitution (good and bad), strong binding of identity to authentication, and improvements to the speed and ease of enrollment, setup and use.

We informally discuss some private reviews our proof-of-concept implementation received from defense and industry experts.

Symposium on Usable Privacy and Security (SOUPS) 2017, July 12 -- 14, 2017, Santa Clara, California.

1. INTRODUCTION

Passwords have a great many problems, not least being 63% of confirmed data breaches involve weak, default or stolen ones[10]. Augmenting passwords with 2-Factor-Authentication (2FA) is a solution banks & other industries use or consider. Unfortunately, existing 2FA techniques suffer very poor usability[6]. A popular method to avoid usability degradation is to simply not use 2FA as often as practical. To mitigate this risk of only occasionally using 2FA, invisible antifraud techniques or hidden multifactor metrics are often used[1]. These include: recording and monitoring user geo-location for risky places or unexpected changes; device finger-printing to monitor for unexpected use of different user machines; and persistent user tracking to record other websites[1] a visitor browses and activities they engage in to classify their reputation as "good" or "bad". Contemporary improvements in authentication security and usability have come at great expense to user privacy.

Considering the importance on protecting users' privacy placed by regulatory authorities such as General Data Protection Regulation (Regulation (EU) 2016/679) with the primary objective of giving citizens back control of their personal data, it is essential to design security systems that are privacy preserving.

From a security perspective, 2FA technologies (including tokens, apps, biometrics, certificates, USB/smartcard gadgets, other OTP, and SMS) suffer many weaknesses and are often breached. The operation of some also puts user privacy (and sometimes safety) at risk. Many workarounds addressing some 2FA shortcomings pose additional privacy and safety risks as well. Refer appendix A and Table 2 for our detailed discussion of usability problems, security issues, and privacy risks associated with current 2FA.

Our research shows that cyber-attacks on authentication systems through phishing, social-engineering, and similar trickery succeed due to insufficient human involvement in the mechanism. Authentication should ensure that ordinary users (including unmotivated, unsophisticated, careless, and/or inattentive ones) can recognize the legitimacy of sites they log into, in a way they cannot ignore. The key concept is "recognize", thus the following idea was born:-

What if a user's eyes, brain, and hands were incorporated into the authentication protocol itself?

The subject of this paper is essentially an answer to this question. We present a fast, easy, complete, high-assurance authentication

[1] Cross-website activity tracking is made possible through 3rd party antifraud vendor insight among their many clients, and popular analytics platforms being present across many sites[1].

and verification solution mutually protecting both ends of interactions against a comprehensive range of modern attacks, throughout the entirety of the authentication "lifecycle" (i.e. provider deployment, user enrolment, use, maintenance, etc, plus side-channels and exceptions).

Our ImageOTP solution demonstrates that user-provider mutual authentication as well as provider-user mutual verification is essential for broadly securing authentication solutions against attack. The match-the-image-and-tap feature of our authentication solution also significantly improves 2FA login usability, making it faster than contemporary 2FA solutions, as well as faster than passwords, which themselves can be safely retired now.

The rest of the paper is organized as follows: In Section 2 and Appendix A we present details on the problems and issues with current 2FA solutions compared with our ImageOTP. In section 3, we present an overview of ideas that led to the development of our mutual-authentication solution. In Section 4 we cover the security aspects incorporated into our design to address a broad range of threats (e.g. those covered in Appendix A). In Section 5, we present outcomes of the dialogue and meetings we have had with end user companies and government agencies to whom we presented our solution. In Section 6, we discuss the importance of Usability and applicability of the solution to identity services. Finally, we conclude the paper in Section 7 with future work.

2. Problems/Issues with current 2FA tech

Most 2FA has many shortcomings, summarized in table 2. It is important to keep all these in mind when designing or evaluating improved authentication. We urge you peruse Appendix A at this point, to appreciate the breadth of the 2FA problem we address.

2.1 Problems and Issues with non-2FA tech

2FA is not the only method used to bolster login security:

2.1.1 TLS Encryption.

TLS does nothing: One 2007 study found that no users whatsoever noticed when TLS was removed[9].

2.1.2 HSTS and HPKP.

Because users fail to notice TLS downgrades, RFC 6797 introduced HTTP Strict Transport Security (HSTS) in 2012, and RFC 7469 introduced HTTP Public Key Pinning (HPKP) in 2015, both designed to prevent TLS downgrade. Both standards require websites to make changes so browsers can prevent MitM.

We tested HSTS availability among internet banking and finance websites in May 2015 & March 2017. We found 19,006 sites from worldwide research[1] and scripted HTTPS "get" to measure HSTS headers. We found adoption is lacking, but growing.

Table 1. MitM mitigation via HSTS in online banking.

	2015	2017
HSTS	286 (1.5%)	1042 (5.4%)
HPKP	0	16 (0.08%)

We additionally tested the 64069 most visited websites according to Alexa in March 2017, and found 6671 (10.4%) supported HSTS; which is almost double that of online banks (5.4%).

2.1.3 Password protection widgets (moving on-screen keyboards)

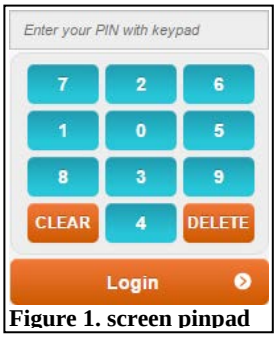

Figure 1. screen pinpad

Keyloggers (malware which steals user keystrokes to get passwords) were considered a major threat in the 1990's and as a result, numerous banks implemented assorted schemes to prevent user password theft by these tools. In general, these consist of showing an on-screen keypad to the user, who then uses their mouse (rather than keyboard) to enter their password. To further prevent mouse-loggers as well, some schemes randomize the location of the on-screen keypad digits, or randomly move the on-screen keypad after each character.

These revel your secret PIN/password to anyone observing your login, offer a frustrating experience to users, are not compatible with password-managers forcing users to memorize their secrets (a problem which itself does not scale), plus keyloggers and malware rapidly adapted, making these methods infective.

2.1.4 Knowledge-Based authentication (KBA).

Some websites challenge users for extra data, or when anti-fraud or transaction security decides step-up authentication is necessary.

Methods include:-
Secret questions and answers, confirming known personal details like birthdays, credit card or phone numbers, postal or email addresses, recent transaction history, and sometimes (especially in banking enrollment) official identity information like licenses, passwords, birthplace records etc. "Memorable Letter" mitigation is sometimes used, requiring users to enter only a few assorted parts of their data; at significant mental effort.

Figure 2. KBA leakage

In some cases, parts of users KBA data are leaked to unauthorized users, facilitating attacks against either the site in question, or against some other site (i.e. information leaked through recovery mechanisms can facilitate break-ins elsewhere).

Ironically, these methods are typically used when a password is considered insufficient or compromised, yet these methods perform KBA over the same channel. They offer weak additional assurance, and store significant privacy-invasive user data in online databases, putting users at risk online and in real life.

KBA can be frustrating and difficult for users, especially those who's circumstances change often, or who authenticate only occasionally, or who have no answers available for preset secret questions (e.g. a child has no first car, first love, etc).

2.1.5 Site Authentication Images.

Some sites ask users to choose a "memorable image" and instruct them to discontinue login if that image is wrong or missing. One 2007 study found this 92% infective[9]. This technique has since been discontinued by many of the larger banks who once used it.

Table 2. 2FA security+ usability comparison.

Refer Appendix A

Legend	
✔	Good
✘	Poor
≈	Limited
	N/A

= security issue		OTP fob	OTP+TV	App OTP	MFA App	SMS-OTP	TouchID	Biom	USB	PKI	TAN	ImageOTP
= useability issue		A-1.1	A-1.2	A-1.3	A-1.4	A-1.5	A-1.6	A-1.7	A-1.8	A-1.9	A-1.10	(this)
A-1.1(a)	Man-in-the-Middle OTP theft	✘	✘	✘	≈	✘	✔	✘	✔	✔	✘	✔
A-1.1(b)	No channel security	✘	✘	✘	✘	✘		✘			✘	✔
A-1.1(c)	Spoofing	✘	✘	✘	≈						✘	✔
A-1.1(d)	Single channel transport	✘	✘	✘	≈						✘	✔
A-1.1(e)	local OTP observable	✘	✘	✘	✔						✘	✔
A-1.1(f)	No signing transactions	✘	≈	✘	✔						✘	✔
A-1.1(g)	No PC malware protection	✘	≈	✘	✘	✘	✘	✘	✘	✘	✘	✔
A-1.1(h)	resistance friendly misuse	✘	✘	✘							✘	✔
A-1.1(i)	Intentional fraud	✘	✘	✘							✘	✔
A-1.1(j)	non-repudiation	✘	✘	✘							✘	✔
A-1.1(k)	PIN protection	✘	✔	✔							✘	✔
A-1.1(l)	mutual authentication	✘	✘	✘							✘	✔
A-1.1(m)	Low Entropy	✘	✘	✘							✘	✔
A-1.1(n)	Agent drivers needed	✘	✘	✔							✘	✔
A-1.1(o)	Seeds and Keys protection	✘	✘	✘							✘	✔
A-1.1(p)	quantum factoring risk	✘	✘	✘							✔	✔
A-1.1(q)	UX: hard to read	✘	✘	✔							✘	✔
A-1.1(q)	UX: Bulky to carry	✘	✘	✔							✘	✔
A-1.1(r)	Do not scale	✘	✘	≈							✘	✔
A-1.1(s)	Expire, or go flat	✘	✘	✔							✘	✔
A-1.1(t)	Prevent Fast logins	✘	✘	✘							✘	✔
A-1.1(u)	Slow setup	✘	✘	✘							✘	✔
A-1.1(v)	3rd party trust	✘	✘	✘							✘	✔
A-1.1(w)	Limited offline utility	✘	✘	✘							✘	✔
A-1.1(x)	Single token only	✘	✘	✔							✘	✔
A-1.1(y)	No self-service	✘	✘	✔							✘	✔
A-1.1(z)	High costs	✘	✘	✔								✔
A-1.2(b)	MitM transaction injection		✘								✘	✔
A-1.2(c)	Partial signatures only		✘					✘		✔	✘	✔
A-1.2(e)	UX: redundant entry chore		✘									✔
A-1.3(a)	Easy to Clone		✘								✘	✔
A-1.3(b)	No Key encryption			✘							✘	✔
A-1.3(c)	Enrollment attacks	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✔
A-1.3(d)	Serverside break-in	✘	✘	✘	✘	✘		✘ ✘	✔	✔	✘	✔
A-1.3(e)	Mobile malware			✘		✘	✘					✔
A-1.3(f)	Cloud backup risks			✘	✘	✘				✘		✔
A-1.3(g)	UX: hard to find right one			✘							✘	✔
A-1.3(i)	Compatibility			✘								✔
A-1.3(j)	Mobile authentication			✘							≈	✔
A-1.4(d)	Downgrade vulnerabilities	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✔
A-1.4(e)	UX: need network				✘							✔
A-1.4(f)	Banned-Camera policies				✘							✔
A-1.4(g)	In-device switching				✘							✔
A-1.4(h)	Offline usage				✘						✘	✔
A-1.4(i)	SIM change				✘							✔
A-1.4(j)	Developer mode				✘							✔
A-1.5(a)	Number porting					✘						✔
A-1.5(b)	SS7 redirection					✘						✔
A-1.5(c)	Malicious micro-cells					✘						✔
A-1.5(d)	Weak, or no, encryption					✘						✔
A-1.5(e)	iMessage sharing					✘						✔
A-1.5(f)	International functionality					✘					✔	✔
A-1.5(g)	Low local protection					✘					✘	✔
A-1.5(h)	3rd party Social-Engineering					✘						✔
A-1.5(i)	malicious replacement					✘						✔
A-1.5(k)	Third party trust					✘						✔

Refer Appendix A

= security issue		OTP fob	OTP+TV	App OTP	MFA App	SMS-OTP	TouchID	Biom	USB	PKI	TAN	ImageOTP
= useability issue		A-1.1	A-1.2	A-1.3	A-1.4	A-1.5	A-1.6	A-1.7	A-1.8	A-1.9	A-1.10	(this)
A-1.5(m)	SIM Change			✘	✘	✘						✔
A-1.5(n)	Unreliable delivery					✘						✔
A-1.5(p)	Poor coverage					✘						✔
A-1.5(q)	UX: random delays					✘						✔
A-1.5(r)	poor portability	✘	✘			✘					✘	✔
A-1.5(s)	Prevents Fast logins	✘	✘			✘						✔
A-1.5(t)	No secure self-service	✘	✘			✘						✔
A-1.5(u)	Expensive support/loss	✘	✘			✘				✘		✔
A-1.5(v)	High costs	✘	✘			✘				✘		✔
A-1.5(w)	Banned; NIST 800-63B					✘						✔
A-1.6(a)	Questionable code quality				✘		✘					✔
A-1.6(b)	overall security reduction						✘					✔
A-1.6(c)	False vendor claims				✘		✘					✔
A-1.6(d)	Low entropy						✘					✔
A-1.6(e)	Easily stolen keys					✘	✘					✔
A-1.6(f)	Easily copied					✘	✘				✘	✔
A-1.6(g)	Unchangeable keys						✘					✔
A-1.6(h)	Widely collected keys						✘					✔
A-1.6(i)	Vulnerable 3rd party fail				✘	✘	✘			✘		✔
A-1.6(j)	False negatives						✘					✔
A-1.6(k)	Environmental reliance					✘	✘					✔
A-1.6(l)	no Backups	✘	✘	✔	✔	✘	✘	✘	✘		✘	✔
A-1.6(m)	Portability	✘	✘				✘			✘		✔
A-1.7(b)	Trivially theivable							✘				✔
A-1.7(c)	transmitted in-the-clear					✘		✘				✔
A-1.7(d)	Illegal to use							✘				✔
A-1.7(e)	Easy to steal							✘				✔
A-1.7(f)	Dictionary attackable							✘				✔
A-1.7(g)	Imprecise							✘				✔
A-1.7(h)	negative privacy					✘		✘				✔
A-1.8(b)	MitM & DPI firewalls	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✔
A-1.8(c)	Injected transactions	✘	✘	✘		✘			✘	✘		✔
A-1.8(d)	Piggyback risks								✘	✘		✔
A-1.8(e)	Infection vector								✘	✘		✔
A-1.8(f)	social-engineering risks								✘			✔
A-1.8(g)	Limited compatibility			✘	✘				✘	✘		✔
A-1.8(h)	Workplace bans								✘			✔
A-1.8(i)	Storage security	✘	✘						✘			✔
A-1.8(j)	Difficult to scale	✘	✘			✘			✘	✘	✘	✔
A-1.8(k)	Single-device only	✘	✘			✘			✘	✔	✘	✔
A-1.8(l)	Inconvenience	✘	✘			✘			✘	✘	✘	✔
A-1.9(a)	Certificate compromise	✘	✘	✘	✘					✘		✔
A-1.9(c)	CA Compromise	✘	✘	✘	✘					✘		✔
A-1.9(d)	hard to verify									✘		✔
A-1.9(e)	UX: complicated	✘	✘							✘	✘	✔
A-1.9(g)	Expiry	✘	✘			✘				✘		✔
A-1.9(h)	Cost	✘	✘							✘		✔
A-1.9(i)	CA Revocation									✘		✔
A-1.9(j)	Portability	✘	✘							✘	✘	✔
A-1.10(b)	TAN Pharming										✘	✔
A-1.10(c)	serverside compromise			✘	✘						✘	✔
A-1.10(e)	Physical replacement issues										✘	✔
A-1.11.1	initial user id	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✔
A-1.11.2	compromised channel enrol	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✔
A-1.11.3	Loss handling	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✔
A-1.11.4	Social engineering of staff	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✔

Legend	
✔	Good
✘	Poor
≈	Limited
	N/A

2.2.1 CAPTCHA

To mitigate 2FA threats, a frequent but near-universally hated technique preventing robots and automated account dictionary attacks is often used. These scrambled-letters or picture-puzzles severely impact usability, and are often ineffective[2].

2.2.2 Know Your Customer (KYC) artifacts

KYC laws require important identity information & documents to be uploaded, typically including social identity numbers, licenses, passports, voting, address, tax, and other registrations. Websites usually exercise care with the storage of this information, but they typically exercise a lot less care with their collection security (refer table 1). Most websites do nothing about user security, other than bury antivirus suggestions in their terms.

Break-ins and malware on PCs and website hacking subject users to identity theft risks. Scanned/photographed documents, if not erased after upload from users' devices, backups, and cloud storage, remain, sometimes perpetually, at risk of future theft. The failure of 2FA has caused there to be no unified identity management solution, which would have overcome these risks.

2.2.3 Excessive attribute release

Using personal information to mitigate authentication ineffectiveness puts that information at great risk.

2.2.1 Unauthorized access and destruction

Ineffective authentication causes 63% of confirmed data breaches[10]; significant good can come from solving this.

3. BACKGROUND

We devised a TAN solution using an assortment of random images printed on it, each having an associated OTP code. During authentication, a user is presented with one randomly selected image on-screen (see fig.4) which differs each login. User then enters the corresponding OTP code from their TAN. This prevents them logging in to wrong websites, since no matching image will show, and it's also impossible for users to ignore the image. Humans are adept at quickly finding like-looking matches. This is also the first method of mutual-authentication between a website and a human that we know of; mutual-authentication is usually between two machines.

Figure 3. Image TAN

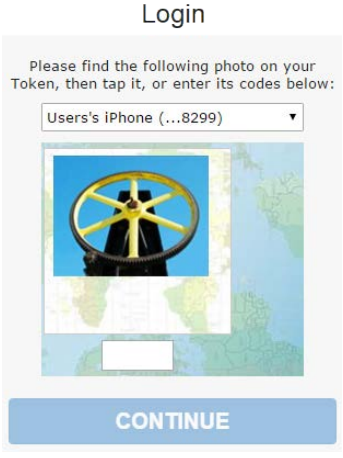

Figure 4. Image login widget

This TAN suffers some drawbacks, such as those described in appendix A-1.10 (however, we do introduce clientside mitigations for many), and it's not exactly an improvement in usability.

To address our TAN usability shortcomings, we then devised a Mobile-App solution based on our matching-image technique.

Figure 5. PC-Mode Image-Matching Mobile App

Figure 6. Mobile-Mode (in-device) Image-Matching App

Users "tap" the matching image on their app (figs. 5 or 6) rather than read and type OTP codes. Together with websockets and mobile PUSH, this reduces authentication to one single tap.

To address remaining security shortcomings not already solved by introducing the App, we devised an independent-authentication-appliance architecture (fig.7), with strict "separation of duties"; identity is managed exclusively by the host website, and authentication by the appliance, neither of which have access to the others' information. This architecture additionally improves privacy by not storing any user identity information in either the App or appliance.

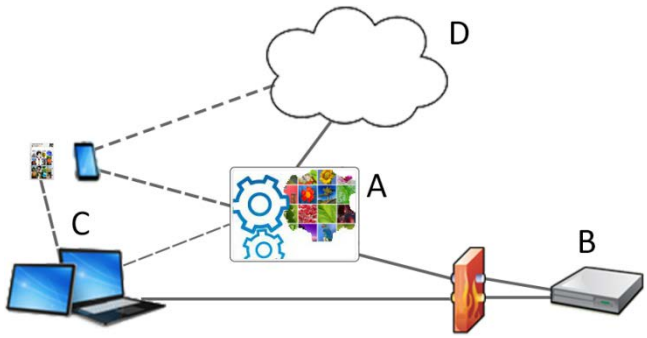

Figure 7. Solution architecture; User "C" (e.g. a banking customer) authenticates to website "B" (e.g. a Bank) with help of appliance "A" supported by cloud services "D".

This Image-OTP solution is robust enough to safely discontinue the use of passwords, since they add little security benefit.

A typical login experience for a user who does not clear their browser cookies (86% of users [4]) is as follows (this flow is identical on PC and mobile, including the mobile with the App):

1. User loads website, which shows them a random image (refer figure 4 for PC, or top half of figure 6 for mobile.)
2. Users' mobile device auto-displays some random images, and the user taps the matching one to authenticate (refer figure 5 for PC or lower half of figure 6 for mobile.)

A complete login (identification via cookie, plus Image-OTP authentication) takes one tap (plus, if necessary, a phone unlock), and takes around 5 seconds. Username/password are not required.

Users who need to identify (the 14% who clear cookies) do this:

1. User loads website and enters their login username (if they see a random image, as would be the case if their cookie existed but they wanted to use an alternate identity, they ignore the image and enter (or choose) their alternate identity at this step)
2. Users' mobile device auto-displays some random images, and the user taps the matching one to authenticate.

Websites uncomfortable omitting passwords put them in step 1.

Our method improves login usability, reducing it to one (usually), fast and fun, match-the-image-and-tap action. It is usually faster than passwords, and always faster than contemporary 2FA.

Enrollment and setup for all users involves directing to their app-store for the app, & pairing the app with their account. On mobile, we take them directly to app installation. For others, we provide a selection of methods to quickly obtain the correct app on their mobile (including scanable QR code, email or SMS with a clickable URL, or letting them manually type-in a shortened URL). In all cases, our enrollment mechanism prepares to allow the App, once installed, to understand which user account requested it, then it automatically provisions and pairs a token with images to the users account. Initial enrollment is 12 steps and takes 2 minutes on average. A second enrollment (by a user who already has the app) is one step and takes under 10 seconds. By contrast, two of the most common Mobile-OTP apps on the market each take 56 steps to complete a successful initial enrollment, and take more than 20 minutes from start-to-end.

4. SOLUTION DESIGN AND CONSTRUCTION

Our design blocks or neutralizes almost all the vulnerabilities outlined in appendix A, and overcomes almost all drawbacks. **More** importantly, typical use takes just one tap, requires practically no mental or other user effort, and takes mere seconds.

4.1 Mobile App design and construction

To facilitate rapid in-the-field app updates and increase code reusability, we designed as much as practical in responsive HTML and JavaScript (JS), and wrote four minimally supportive native outer- containers to host the HTML control (one each for iOS, Android, WindowsPhone, and Blackberry). The native component also provides PUSH, sensor and device-biometrics access, user-notifications, protected-storage facilities, QR scanning, and where necessary for the platform and version, cryptography. The HTML components do all display, and JS performs almost all processing. All supporting HTML and JS is digitally signed, and the native components verify signatures. Apps check for and download HTML+JS updates at open, periodically, and if requested via PUSH. Updates themselves are signed, and applied immediately. Data is exchanged and stored in encrypted JSON packages.

The soft-token structure containing the assortment of images is stored in JSON in device protected storage – an operating-system protected area which is not backed-up to user cloud or PC devices. Tokens are created & supplied to the App during user enrollment by the authentication security Appliance (see section 4.2). Our TAN cards (aka "Hard Tokens" – see section 4.3) are minted the same way, being printed from JSON rather than used in an App.

The Token JSON contains: (1) A random 12-digit TokenID expressed in numbers, EAN13 barcode, and QR. They carry no metadata. These have assorted uses, one is to help users distinguish between possible multiple tokens they might own for the same website. The barcode and QR are scannable by the App to facilitate rapid enrollment. The QR embeds an auto-discovery URL along with the TokenID, thus it works correctly even when scanned by a wrong reader. (2) A small subset of random photographic images, selected from a licensed set of 11,000 and manually inspected to remove all possibly controversial and ambiguous ones; the selection algorithm additionally rejects similar-looking images when assembling tokens. All images are watermarked, subtly mangled (digitally distorted to hamper simplistic machine matching), and digitally signed with EXIF tracking inserted to detect possible future misuse. (3) A random 64bit OTP code for each image. OTP codes are not stored on Appliances; we use "6+ cost" (cost being chosen based on appliance CPU power and expected peak login load) multi-round double-salted (64bit per-token salt plus 64bit per appliance salt = 128 TRNG bits) bcrypt versions with naming-deceptions to mitigate server-side break-in vulnerabilities. (4) Manual per-image typeable random alphanumeric OTP codes which are case intensive, but printed in differing case to eliminate ambiguity. We use base-35, since we accept ambiguous zero and letter "o" equivalently. OTP codes vary in length; they are used in the App in situations where no data connection is available and a users' only option is to manually type in a code. They are hidden from users when unnecessary (i.e. almost all the time), and screened during minting to avoid offensive words. These are stored as images (not text) with the same protection as the photographic images, to hamper potential in-device malware extraction. (5) Issuing metadata for auditing (versions, date, mint options, revocation provision). (6) A per-token 64bit random shared secret salt. (7) Branding logo and name of the website. (8) Hash of providers' customer identifier, which is itself hashed to enforce separation-of-duties. (9) Appliance endpoint URL. (10) QR-code resolution URL. (11) 2048 bit RSA keypair generated natively in-app during token installation (public key is uploaded to appliance) after generation. Images and associated OTP keys can be automatically replaced after use.

Token JSON is encrypted to a per-device static key, which is optionally combined with a users' per-token password and/or biometric. This prevents stolen JSON being decrypted without the device, and prevents (e.g.) friends and family from making use of tokens in the App. This latter protection adds an extra step to the login process: the user supplies their fingerprint or password to unlock their token before they can tap a matching image.

Mobile-malware risks required our app design to incorporate defensive techniques: we used a commercial anti-tamper wrapper to fortify our app object code, and a server-initiated integrity self-check whereby the app sends a digest of server-specified (randomized) areas of app runtime; any mismatch detected at the server will indicate a tampered-with app or runtime environment.

We built our native components maintaining backwards

compatibility with the widest practical range of legacy devices to afford protection to the largest numbers of users. Our apps support iPhone 3 and most older iOS, and almost all Android devices, plus old BlackBerry and WindowsPhone too.

4.2 Appliance

We chose CentOS 6 with SELinux for our appliance operating system, and commissioned a professional security-hardening and custom "two-man rule" policy; the "root" user has no permission to exercise their normal super-user rights unless granted that permission by an oversight operator, who themselves has no other permission but to grant "root" when required. We wrote an installer which makes use of "dd" and a custom net-install grub image to erase any existing operating system it finds itself on, and do an unattended fresh-install of a clean new O/S from verified media; this we felt was necessary for all cloud environments since it's impossible to know what might have been done to your O/S before you're given control. Linux auto-generates many keys during installation based on DPRNG (deterministic pseudo-random-number generators), and since most clouds provision from "clones", it's never clear how secure these might be. Our solution is incompatible with OpenVZ style containers, but this could be considered an advantage since those do not enjoy great separation from their host. We enforce LUKS encryption, with a modified netinstall bootloader capable of requesting the decryption key (using another appliance to notify the operator to approve the (re)boot). This provides reasonable protection against host-launched attacks and permits headless reboots while not leaving keys vulnerable to invisible theft. Our installer loads TRNG (True-Random-Number) hardware drivers in advance of key generation to ensure quality entropy is used for them (we used vmware on servers with USB TRNG devices). We "patched out" the "seed" functions in several underlying cryptographic libraries to prevent possibility of non-random number requests, and we modified the actual random number routines themselves adding an additional XOR step to mangle the routine-chosen random with random we draw direct from the TRNG hardware (this was done because the existing random routines have suspiciously complex code and we could not determine if this was for safety or backdoor purposes. Since one vendor of DPRNG algorithms was found to have taken payment to backdoor their code, we feel an abundance of caution here was justified). We configured TLS to achieve an SSLlabs A+ rating, disabled all plaintext and/or insecure protocols, activated HSTS, HPKP, Fallback Signaling Cipher Suite Value (SCSV – which prevents protocol downgrade attacks), Online Certificate Status Protocol (OCSP stapling – for revocation), and perfect-forward-secrecy. We fortified SSH with our PAM second-factor protection (to an unrelated appliance of course) – our appliances thus protect one another.

Our stateless auto-sync allows redundant appliances to add DDoS resilience, improve geographic speed (reduce latency), and increase load capability. Any appliance in a related constellation can service any user at any time. We chose physical servers with redundant BIOS, memory, power, storage, and networking, and located them in locked cages in datacenters with no-unescorted access, and 24/7 NOC; one in San Jose, USA to give low latency to USA users, and a peer in Cork, Ireland, chosen after an extensive search for a provider who would guarantee not to "take down" our appliance in the (unlikely) event of receiving a USA court order. We negotiated different utility providers for each of the 2 power supplies in USA, and different network providers for each of the two network connections in both. We felt it was necessary to protect not just physical appliance security, but also legal, regulatory, environmental, and logical security as well.

We rented one Amazon and one Azure cloud server to host a sample banking website and government tax website to demonstrate our solution live.

To add our protection, a website deploys and configures an appliance (or uses public ones), then takes the following steps;

1. One "blank" page is provisioned, and one menu option is added to their site navigation: the page is used to display the enrollment and token self-service maintenance subsystem which is drawn by the website from the appliance machine and shown to the user, & the menu option is used to reach this page.
2. The website modifies its login procedure; at user-identification stage, an API call is made to the Appliance to determine if protection is already set up: if yes, the appliance responds with the image-display widget (figure 4 and section 4.5.1) which the website send to the user's web browser. If not; the appliance responds with an enrollment wizard which is also sent by the website to the user's browser, and which guides the user through the process of getting the app with a token.
3. Any transactions the website deems needs protection against malware are also modified (e.g. financial transfers, cloud-server erasures, password/address changes, etc); the website adjusts its processing to send a suitably formatted representation of the proposed transaction to the user, who inspects this and when happy confirms it (digitally signs with their in-app private key). Prior to processing, the website checks the signature matches the supplied form fields and the user approved (did not deny) it.

4.3 TAN Cards

We retain the physical TAN card idea, but these hard tokens are given the new name "Recovery Token". They become part of the binding identity-to-authentication step in physical user provisioning (e.g. in-bank-branch account opening) for customers who did not bring their phone with them, or they can be mailed out, or printed at home by users. Users are instructed to safeguard these tokens, which will be used in future to recover access to their account in the event their phone can't be used (e.g. lost, stolen, flat, etc). They can also be used to support users who don't have, or don't want to use, or aren't allowed (e.g. military personnel on service) a mobile device for authentication.

4.4 Optional Anti-MitM Agent

Preventing man-in-the-middle attacks, including blocking unwanted TLS proxies and other certificate substitution mechanisms, while permitting wanted ones, is accomplished as per below. Since this step is considered impractical because it requires users to have installed an active-channel-binding agent on their web browser, we mark it as optional. Since transaction-verification already mitigates MitM attacks, the only benefit found from blocking them it to protect user's privacy (as opposed to the money in the bank account). It is hoped that our planned future standard will be incorporated into O/S and browser vendor products permitting all those with need to secure against MitM threats a safe, privacy-preserving mechanism to do so.

The challenge to solve was how to prevent something "in the middle" from simply relaying what they see on their screen to a victim, in order to facilitate an attack (including initial enrollment as well as authentication and transaction entry/signing), and how to prevent this without invading user privacy or subjecting users

to risk of future unwanted tracking or other side effects.

We modified our TLS server to provide our application code with a digest of the TLS symmetric session key (master secret), which thus offers us means for our client agent installed in a user web browser to detect an intermediary (the TLS session key is securely derived; neither end can force the other to generate the entirety of a key they might want; more specifically, no intermediary can force both its peers to use a matching key). We use a digest, not the key itself, so as not to weaken the encryption. The agent contains a compressed library of 11,000 optimized image file thumbnails. During authentication, our TLS server determines the index for the image thumbnail the client needs to show, and encrypts this index using a combination of the session key digest and agent-key (a random per-user per-site key derived at enrollment), and sends this to the agent. The agent decrypts this index, and displays the image thumbnail locally to the user (in this protocol, no images travel over the network). In the event of a session key mismatch, or the event of an incorrect or unexpected agent (e.g. an attacking intermediary's agent), the resulting wrong decrypted index will cause display of a wrong image, thus preventing a user from login because it will not match one they can tap (i.e. there is a 11,000:1 chance against the mismatch displaying the correct image).

In the event of a wanted intermediary (DPI firewall), the agent detects the certificate mismatch and displays a suitable warning and acceptance option to the user, which if chosen, re-computes the index using both TLS session keys (the user-side connection and the server-side one), which facilitates the correct image display. TLS-Proxy support is possible to prevent additional unwanted MitM attacks, and additionally allow wanted inspection (i.e 2 or more MitMs at once).

Secure enrollment over a possibly compromised channel is possible. The challenge to solve was how to prevent an intermediary from tricking a user such that the intermediary can enroll themselves, or more specifically, how do we ensure that the agent in use is the real users agent, and not the MitM one.

We solve this by using our App. At enrollment, the agent generates a key to be later used in combination with the TLS session key hash to enforce channel integrity, and it also sends to the website a signed copy of the TLS certificate it observed. The agent generates one real key, and several decoys, and displays them locally (refer figure 8), using a method unavailable to a web browser (writes the keys on the desktop and dims the screen). Beside the real key is the users logged in PC-username, with fake usernames alongside the decoys. Instructions for how to proceed are sent to their App, which asks them to minimize their browser and enter the key from their screen corresponding to their logged-in PC username. The intermediary is prevented because they cannot trick the real user for two reasons: (1) the intermediary cannot display *their* agent key in the correct manor, since the intermediary is constrained to the user's web browser, and (2) the intermediary cannot show the user the users' logged in username, since this information is not available to a web browser. The intermediary is prevented from "social engineering" an enrollment because the App itself guides the user to help prevent them from attack, which is why the app instructs the user to minimize their browser (to hide any decoy intermediary instructions); note that in a non-attacked enrollment, the agent will minimize the browser automatically – instructions are necessary on the App to since an intermediary might be preventing the agent from receiving the website instructions.

Figure 8. Agent binding PIN

This mechanism resists both modification and suppression attacks by intermediaries. Keys are computed such that no private information is leaked (no website can ascertain what other sites a user might visit, or what other identities a single user might use on that site, and no mechanism to retrieve indelible identifiers from user machines exists).

4.5 Protocol

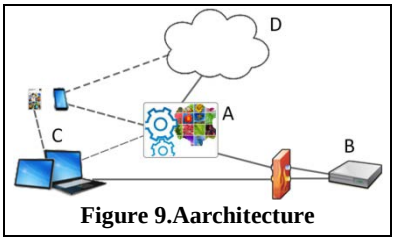

Figure 9.Aarchitecture

Customer C (figure 9) using their PC, Tablet, and/or phone and token accesses service B (via firewall). Appliance A helped by cloud D provides authentication, transaction verification, and security.

4.5.1 Authentication

Customer C loads website B and enters username (if not supplied by cookie) and optional password. Server B makes API call to appliance A to determine if customer is enrolled. If not, B logs customer in. If yes, appliance A returns challenge widget to B which it displays on customer's C's browser, and A additionally triggers a PUSH through cloud D to auto-open C's token for them. Customer C uses token to solve the challenge by taping the image on their phone which matches the one displayed by the widget. This tap triggers C's mobile device to securely communicate a signed (by C's token's private key) and encrypted (to A's public key) OTP response to A, which (if correct) signals C's browser to auto-proceed (for convenience, not security). Encryption and signing exist to defeat malicious TLS (cert substitution) MitM. B checks again the customer's OTP is correct, and logs them in. A is isolated from (has no knowledge of) C's identity.

When customer C has no mobile device, they submit the OTP manually into their browser from their physical token to login.

Our widget sends the image via in-device URI, switching context to the App and back, for users browsing on their mobiles (fig.6).

Our widget includes self-protecting malicious bot detection, for detecting scripted MitM attacks trying to steal images.

4.5.2 Transaction Verification (inline)

Customer C submits some intended action (e.g. a money transfer) via a web form to website B. Appliance A receives the purported intended transaction (via B or via C's browser, depending on implementation), prepares it for display to C', and triggers a PUSH through cloud D to C's mobile device, which securely retrieves the transaction to be displayed from A (signed by A's private key and encrypted to C's public key). C verifies this transaction shown on their phone is correct as they intended, then taps the "approve" or "decline" option, which generates a digital signature of their response and all transaction form elements, and communicates this signature to A, which in turn communicates it to B. (via C's browser if not directly). B checks signature match and customer approval, and processes the verified transaction. In

the event of a decline by C, or mismatch at B, A additionally triggers a D PUSH to request decline reasoning from C, which is passed to B (for attack and customer compromise reporting).

4.5.3 Transaction Verification (out of band)

B triggers a D PUSH to C's mobile device, which sounds an audio alert. C unlocks their phone (if not already) which retrieves the transaction from B and displays it C, who follows the on-screen instructions, and taps or selects an appropriate option. C's response and any associated data is signed and communicated to A which in turn sends it to B, which processes or displays it needed. "Transactions" suiting this flow include mutual authentication of two parties over telephone or in-person, control of no-screen IoT devices, 2FA Pluggable-Authentication-Modules (PAM) processing etc.

5. RESULTS AND RECEPTION

We succeeded arranging 1 hour meetings with the security teams from the largest 4 banks in our country. All teams were impressed with speed and ease-of-use, and only one team found an area for concern: deliberately fraudulent customers exploiting online banking security guarantees (which we since solved with transaction non-repudiation). We learned that describing new concepts to security professionals is incredibly difficult; they often mistake our techniques with similar past marketplace failures, and it took great effort to convey how visual mutual-authentication works, and differs to contemporary methods. We learned that most banks do not disclose their fraud levels, that social-engineering and malware are their top concerns, and that some banks use non-consensual anti-fraud biometrics collection.

We presented to corporate security teams from a large search engine, a large software vendor, and a large social network; the only concern raised was one team member expressing "friendly fraud" concerns (specifically; his girlfriend using his phone), which we since solved with password and biometric token encryption.

We were invited (after background checks) to our nation's capital to meet its peak defense agency. Four experts; including one pentester, one policy, one cryptographer and one who's role was classified, met us. They described themselves as the best in their roles countrywide, and reviewed our solution in-person (they understood immediately). We showed several live demonstrations, then answered rapid-fire questions for an hour. The only weakness they found was our (since fixed) handling of generated tokens after delivery. We asked them each what they believed was the most important part of our solution: two chose our speed, the other two chose simplicity. Keeping in mind these are the top security experts in our country, and none put security first, this confirmed our belief that usability is more important than security when it comes to real-word implementation.

5.1 Protection efficacy and user experience.

Our meetings informally confirmed that our solution improves authentication usability (speed, ease, convenience, ubiquity, and compatibility) for authentication, transactions, and enrollment, on both desktop and mobile for all users, savvy and problem alike.

Security and related topics explored during our meetings included: (1) Active and passive man-in-the-middle and spoofing attacks, (2) key-loggers, (3) mobile and desktop malware, (4) shoulder-surfing and other passive credential thefts, (5) social engineering against users, staff, and 3rd parties, (6) wateringhole attacks, (7) forgotten passwords, (8) weak/re-used passwords, (9) server-side break-ins, (10) typosquatting, (11) clicking malicious links, (12) opening phishing emails, (13) dictionary attacks and related denial-of-service problems, (14) friendly fraud, (15) deliberately fraudulent customers (non-repudiation), (16) phone loss/theft handling, (17) password replacement (using no passwords at all), (18) scalability, (19) infrastructure, privacy-respectful architecture and device identifiers, (20) separation-of-duties, (21) integration cost, effort, and methods, (22) support requirements and costs, (23) telephone call-center customer verification modes, (24) in-person mutual authentication, (25) revocation, (26) self-service (including replacement/re-enrollment), (27) updates, (28) expiry, (29) secure enrollment over compromised networks, (30) deep-packet-inspection device tolerance, (31) TRNG usage and no master keys/secrets, (32) entropy strength, (33) customers with multiple devices, (34) multiple customers sharing one device, (35) one customer with multiple personas on one or more devices and our strong privacy isolation for this (36) implementing multiple brands or disparate systems at once (37) users with no mobile devices, (38) international and offline usage modes, (39) secure storage, (40) security standards adherence & potential LoA3 accreditation strength, (41) mobile phone number porting, (42) risks of password-managers, (43) multi-channel transport strength, (44) minimal training, (45) configurations options & policies, (46) rapid/automatic logins, (47) factoring and post-quantum resistance, (48) backups, (49) cryptographic algorithm choices (50) agentless operating capability, (51) login-flow positive impact & customer security perception (52) compatibility across different machines, and (53) single-device usage mode.

6. DISCUSSION

6.1 Importance of usability

We believe usability is the most important aspect of security, because security protects no-one if it's not used: If it's hard to use, it won't get turned on, if it's slow, users won't want to use it, if it's hard to enroll or understand, many won't be able to use it, if it's inconvenient or doesn't scale it will get resisted, if it is banned or can't plug in it will be impossible to use, if it's too expensive many won't be able to afford it, if it won't work offline or abroad it will be unreliable, and it must work for everyone, everywhere, always. It's a good idea too, if it's secure and broadly effective!

6.2 Applicability to Identity services

Many modern privacy concerns could be solved if a respectful, identity framework was widespread. Uses could, for example, digitally prove (with anonymous non-repudiation) to a bartender they're old enough to drink, without being forced to show ID that displays their full birthday, name, address, registration numbers, signature, blood type, donor status, etc. Such an identity solution would fit well into our framework, protection, and architecture

7. CONCLUSION

Anecdotal results show exciting opportunity for our technique to improve user experience and security.

7.1 Future work

Our work and proof-of-concept would make an ideal basis for:-

- A study on authentication efficacy and usability; to the best of our knowledge, no broad comparison has been done on authentication methods/products & modern attack vulnerability.
- Work on a privacy-respectful identity/attribute assertion system.

- Empirical study on this solutions' efficacy and usability.

8. ACKNOWLEDGMENTS

We thank the defense and numerous industry security experts who freely and eagerly gave up their time to review our solution and their quest to try and find possible oversights in it.

9. REFERENCES

[1] Avast forum "List of Online banking sites in your country" *https://forum.avast.com/index.php?topic=83592.0*

[2] Bursztein, Elie. Aigrain, Jonathan. Moscicki, Angelika. Mitchell, John C. (Aug 2014) "The End is Nigh: Generic Solving of Text-based CAPTCHAs" *8th USENIX Workshop on Offensive Technologies https://www.usenix.org/system/files/conference/woot14/woot14-bursztein.pdf*

[3] Castelluccia, Claude. Narayanan, Arvind. (Oct 2012) "Privacy considerations of online behavioural tracking". *The European Network and Information Security Agency (ENISA)*

[4] Clifton, Dr. Brian (March 2010) "Understanding Web Analytics Accuracy"; https://brianclifton.com/pro-lounge-files/accuracy-whitepaper.pdf

[5] Dunkelman, Orr. Keller, Nathan. Shamir, Adi. "A Practical-Time Attack on the A5/3 Cryptosystem Used in Third Generation GSM Telephony" *Faculty of Mathematics and Computer Science, Weizmann Institute of Science*

[6] Krol, Kat. Philippou, Eleni. De Cristofaro, Emiliano. Sasse A, M. Angela (Jan 2015) "They brought in the horrible key ring thing!" Analysing the Usability of Two-Factor Authentication in UK Online Banking. *University College London*

[7] Marques, Diogo. Muslukhov, Ildar. Guerreiro, Tiago. Beznosov, Konstantin. Carriço, Luís. (June 2016) "Snooping on Mobile Phones: Prevalence and Trends" *Symposium on Usable Privacy and Security (SOUPS) 2016* https://www.usenix.org/conference/soups2016/technical-sessions/presentation/marques

[8] Panjwani, Saurabh. Prakash, Achintya. (July 2014) "Crowdsourcing Attacks on Biometric Systems" *Tenth Symposium On Usable Privacy and Security: SOUPS'14*

[9] Schechter, Stuart E. Dhamija, Rachna. Ozment, Andy. Fischer, Ian. (May 2017) "The Emperor's New Security Indicators An evaluation of website authentication and the effect of role playing on usability studies". *The 2007 IEEE Symposium on Security and Privacy.* http://www.usablesecurity.org//emperor/emperor.pdf

[10] Verizon. "2016 Data Breach Investigations Report" (DBIR) *http://www.verizonenterprise.com/verizon-insights-lab/dbir/2016/*

APPENDIX

A. Problems/Issues with current 2FA tech

This appendix supplements table 2 from page 3.

Most 2FA technology is based on one-time-passwords (OTP). 2FA has many shortcomings. It is important to keep all these in mind when designing or evaluating improved authentication.

A-1 Categories and vulnerabilities of 2FA

This appendix groups the different kinds of 2FA available into ten categories, and outlines the drawbacks and vulnerabilities of each. To avoid repetition, subsection A-1.11 afterwards addresses general failures that all ten 2FA categories suffer.

A-1.1 OTP hardware.

Figure 10. OTP token

Hardware-based or keyring-style OTP tokens are the most well-known 2FA category. They generate new random codes every one minute or so based on a per-token ID, the time, and seed or key material programmed by the vendor. Codes are typically valid for double or more the length of time they're displayed (to accommodate clock skew and slow typists). When invented[2] in 1984 (8 years before the invention of the world-wide web), time-limited OTP passcodes had better chance of improving security because networked machines and real-time attacks were rare.

Security vulnerabilities of hardware OTP include:-

a) Man-in-the-Middle (MitM) attacks; intermediary can steal OTP
b) No channel security; there is no association between OTP code and a secure channel, leaving the protection of codes against theft out-of-scope: it's the website's job to use TLS with HSTS and HPKP etc, & the user's job not get tricked or downgraded.
c) Spoofing; there is no binding of tokens to resources. Imposters can capture codes, and have several minutes to use them.
d) Single channel transport; techniques which steal passwords like keyloggers, phishing, malware, and social engineering of the user equally succeed stealing OTP codes too.
e) No local protection; codes are typically displayed on a screen which has no protection against unauthorized viewing
f) No utility for signing transactions; OTP codes bear no relation to user activity so are inappropriate to confirm user instructions
g) No malware protection; Because OTP cannot sign transactions, malware can inject/modify instructions, which get innocently permitted by users unaware the OTP code is being hijacked.
h) Very low resistance to misuse by friends, family, or peers.
i) Intentional fraud: Sometimes it's not the bad guys defrauding a user, but bad users defrauding (for example) their bank. Fraud-free guarantees are often abused by unscrupulous customers.
j) No non-repudiation; OTP does not prove user intent.
k) No PIN protection; most OTP tokens have no keypad.
l) Lacking mutual authentication; OTP code-use is one-way only; no mechanism to verify authenticity of the website exists.
m) Low Entropy; only short numeric codes are supported.
n) Serverside OTP support typically requires installation of hardware and drivers, which carry their own risks of compromise. The $1.1-trillion hack against the US Office of Personnel Management was ironically facilitated through privilege escalation attack against their OTP Driver software.
o) Seeds and Keys protection; OTP tokens are based on a master secret, which when stolen, compromises all user OTP tokens at once. This infamously occurred in 2011 when a phishing email stole keys from an OTP vendor which were subsequently used to facilitate military contractor organizations break-ins. Upto 40 million compromised tokens were subsequently replaced.
p) Most OTP is based on asymmetric cryptography, threatened by quantum computing and advances in factoring techniques.

Drawbacks of OTP hardware include:

q) Multiple Usability issues: they interrupt and dramatically slow down user authentications. They have no backlight making them sometimes difficult to read. They are bulky and require physical carriage. Usability is so poor; banking customers have switched banks to avoid being forced to use OTP hardware[6].
r) They do not scale: Users require a new physical OTP token for every website login requiring protection. At time of writing, this Author (a long-time internet user) has 2838 unique accounts across 2277 websites; if all were protected by OTP-token, that would cost $100,000 in tokens, weigh 93lbs (42kg), take half an hour to locate the correct one for each login, prevent logins when away from the token-room, and require 56 replacement tokens each week as batteries go flat, taking 40 hours to reenroll, costing $20,000p.a. to buy the replacements.
s) They fail, expire, and go flat: OTP tokens typically last 5 years. Some policies expire them sooner (prior to battery exhaustion) some fail through clock sync, battery or environmental issues.
t) Prevent Fast and Automatic logins; OTPs require manual code reading and typing. They cannot support automatic/rapid use.
u) Slow setup; OTP's require shipping, and once received, usually require ~ 30mins setup and enrollment procedures.
v) 3rd party trust; OTP keys are typically made at and kept with the token vendor. Any theft of misuse of these keys allows an OTP token to be emulated by an adversary; see (2.1 above.
w) Limited offline utility; OTP tokens are rarely used to authenticate customers over the phone or in person.
x) Single token only; Most OTP client implementations allow for just one user token; there is no provision for users needing more (e.g. one token at home and a second at work).
y) No self-service; OTP are hardware devices, which require costly deployment/handling which users cannot do themselves.
z) High costs; OTP devices themselves are expensive, the serverside hardware and licenses are likewise expensive, and the support costs and periodic replacements also expensive.

[2] 1984 OTP Patent http://www.google.com/patents/US4720860

A-1.2 OTP with transaction-signing (OTP+TV)

Figure 11. OTP+TV

Some OTP hardware includes a keypad, useable for Transaction Verification (TV). These are typically PIN protected and also capable of providing plain OTP codes for authentication. Signing consists of entering numbers (e.g. PIN, source, destination, and $ amount of financial transfers) to produce a verification code based on all the information keyed in, which the user then types back into the website.

Security vulnerabilities of hardware OTP+TV include:

a) When used in OTP-only mode (as opposed to TV mode), these suffer all the same problems as plain OTP except for the ones mitigated through the use of the PIN pad protection.
b) Rogue transactions via MitM, spoofing, and malware: In banking context, the limited no-prompts OTP-TV display makes it hard for users to understand the meaning of the numbers they key in and to know and check they're correct in the following three different places: (1) their original transaction they submitted (e.g. through their PC). (2) the on-PC-screen prompts telling the what to type on their OTP+TV keypad, and (3) the numbers they manually enter on it.
An adversary with privilege to modify user screens can substitute the intended receiving account destination with their own, and can adjust transaction amount almost unperceivably. For example: to transfer $100, a user keys in 00010000. If malware told them 00100000 instead, it's unlikely they'd notice. Similarly, recipient partial-account numbers might be subtly or completely adjusted, and/or the bank to which the payment is intended, not being part of the signature at all, is free to be modified by the attacker.
c) Partial signatures only: no facility exists to sign the actual submitted transaction (which would include recipient names, routing numbers, banks, dates, other instructions, and notes); signatures are limited only to the least-significant digits of recipient account identifiers; the rest is at risk to malware.

Drawbacks of OTP+TV hardware include:

d) These tokens also suffer all the drawbacks of OTP tokens discussed in section A-1.1.
e) Usability; entering every transaction twice on the small and low-quality keypad becomes a major chore for users. Many users, including this author, dread using these exhausting devices so fiercely, that avoiding transactions as much as possible becomes common practice.

A-1.3 Mobile App OTP

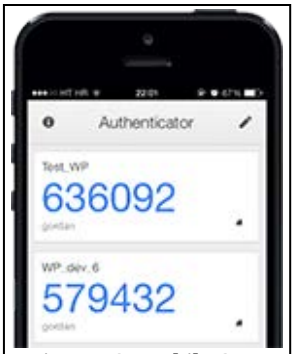

Figure 12. Mobile OTP

Some mobile apps replicate OTP hardware, thus they suffer most of the vulnerabilities and drawbacks discussed in section A-1.1 in addition to more discussed here.

Security vulnerabilities include:

a) Cloning; Mobile-OTP keys live usually without protection on the user's mobile device.
b) No Key encryption; most Mobile-OTP does not have PIN or passwords protecting OTP codes. While phones themselves are usually locked, 31% of us still suffer a "snoop attack" against our phones every year anyhow[6].
c) Enrollment attacks; Enrolling a Mobile-OTP requires sending the key material to the device; this is usually done via QR code or typeable text string. Intercepting these codes allow adversaries to generate future OTP codes at will.
d) Serverside break-in; The webserver must store the per-user OTP key in their database; this is usually kept in the same table that usernames and passwords are in. Any webserver flaw resulting in a password breach will also result in the loss of all OTP keys as well. Such break-ins and thefts are common.
e) Mobile malware; In-device malware might have access to steal user keys. On "rooted" or "jailbroken" devices, and unpatched /older devices with escalation flaws, nothing protects the keys.
f) Cloud backup; Most mobile devices backup their storage to cloud servers, putting OTP keys at risk of serverside theft.

Drawbacks of Mobile-OTP include:

g) Usability; while Mobile-OTP enjoys the benefit of being always available to most users most of the time, it does still require the user to unlock their phone, locate the requisite app and open it, then hunt through their list of OTP codes for the one relevant to their account and username, before finding and typing back in their OTP code.
h) Scalability; finding the right code to use at each login is an N-squared complexity problem. Each extra login makes it slower and harder for all other logins across all accounts every time.
i) Compatibility; many OTP apps refuse to run on older devices "for security reasons". Ironically, this misguided protection effort guarantees those users get no protection at all.
j) Mobile authentication; Using Mobile-OTP to access a Mobile account on the same device requires a competent user who can quickly switch between apps, and remember random 8 digit codes. Millions of users, especially elderly, young children, and others most vulnerable will be unable to do this.

A-1.4 Modern multifactor mobile Apps with signing

Newer mobile apps are significantly more advanced than the Mobile-OTP category, carrying vastly improved usability, good transaction verification (TV) and signing, and sensible protections like password or biometric key protection, thus can guard against some of the more obvious attack scenarios. Since many incorporate GPS, biometrics, device-ids and more, they are more accurately described as multifactor (MFA) than just second-factor.

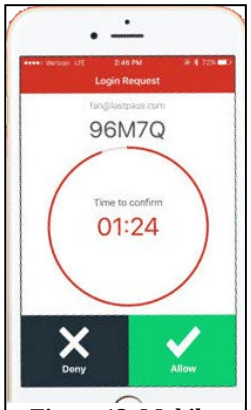

Figure 13. Mobile-MFA

Mobile phones travel almost everywhere with nearly every person who would want to have 2FA. They're a central feature in the lives many, who take great care to protect them. They do still get lost or stolen, but we think it's fair to say that there is no single thing that humans put more collective effort into ensuring not to lose, than their phones.

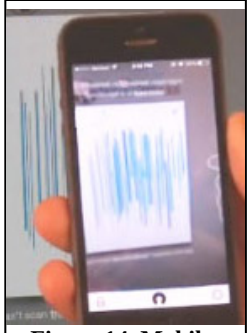
Figure 14. Mobile-

With their ubiquity, sensors, power, and network connections, mobile phones are ideal authenticators.

Security vulnerabilities include:

a) No MitM, spoofing; or malware protection; An imposter can cause a legitimate Mobile-MFA user to authenticate the wrong person (the imposter). There are some apps which use a phone camera to scan onscreen codes in a partial attempt to prevent simplistic MitM, but these too fail to prevent authenticating the attacker (since the attacker is free to simply present the scanable challenge to the legitimate user.)
b) No channel protection; No Mobile-MFA implements working mutual authentication – absent a skilled and attentive user, no protection exists to ensure the users connection to their webserver is uncompromised.
c) Cloud backup; Modern Mobile-MFA is less susceptible to insecurities of backup data on cloud servers, since they are expected to be making use of PINs, biometrics, device-ids, and protected storage (non-backed-up) features of the modern mobile OS, however, implementations between vendors vary, and not all of them take these precautions.
d) Downgrade vulnerabilities; most Mobile MFA supports insecure fallback methods such as resorting to code-entry Mobile-OTP for situations where the app has connectivity issues, subjecting them to the vulnerabilities and drawbacks discussed in the previous section A-1.3.

Drawbacks of Mobile-MFA include:

e) Usability drawbacks vary widely across Mobile-MFA vendors. Some apps auto-open using PUSH and auto-communicate codes and signatures so users don't need to type things in. Others require users to manually open apps and find tokens.
f) Banned-Camera policies; Mobile-MFA requiring cameras will not function in workplaces (e.g. military, secure) prohibiting them or their use (especially recording screens with phones).
g) In-device switching; Using an app or browser on the same mobile device as the Mobile-MFA requires user's adept at using their mobile OS to switch back and forth between apps.
h) Offline usage; Mobile-MFA requires a working data (wifi or cellular) connection to function. International travelers and low-credit mobile users will find this expensive and frustrating.
i) SIM change; Many Mobile-MFA apps cease to function when SIM cards are changed, purportedly for "security reasons" (we assume stolen phones or hijacked apps). Since most international travelers change SIMs when abroad to keep their roaming costs low, this causes cost and usability problems.
j) Developer mode; again for "security reasons", many Mobile-MFA apps refuse to open if the phone is in "development mode". People with "rooted" or "jailbroken" their devices are permanently blocked from using these Mobile-MFA apps.

A-1.5 SMS OTP

Mobile phone text-messages are the mode widespread OTP in use, and the least secure, and the least reliable.

Figure 15. SMS OTP

Security vulnerabilities are:

a) Number porting; Many ways exist to hijack a user's phone number and SMS messages; this is a common and successful attack.
b) SS7 redirection; Cell-network protocols permit unscrupulous operators anywhere in the world to inject commands rerouting (thus intercepting) SMS, voice, and cellular data traffic for any subscriber. Public, with-permission (but without-assistance) attacks against high-profile victims have been demonstrated.
c) Malicious micro-cells, and radio sniffing; Software-Defined Radios (SDR) sell for under $10 on eBay, and free opensource software turns them into local (and remote) SMS sniffers.
d) Weak, or no, encryption; Mobile network encryption is weak, taking (depending on generation) between 2hrs to less than 1 second to crack on a single PC [5]. Modified cell traffic attacks which disable encryption entirely are relatively easy to mount, are commonly found active in cities, and proceed undetected on all but purpose-designed secure-cell handsets.
e) iMessage sharing; SMS-OTP messages often distribute across different accountholder devices and show up on multiple user screens at once. This further subjects SMS to thefts since intruders with user cloud account access can register their own devices on this account to receive them.
f) Downgrade situations Many organizations recommend users disable their SMS-OTP when travelling; a risky decision for most users since this is the time they will most need 2FA!
g) Low local protection; many handsets display messages on lock-screens, with no protection against

Figure 16. A governments' advice to citizens urging to disable SMS-OTP before travelling abroad.

being observed by malicious 3rd parties.

h) Social-Engineering against 3rd parties; Many customer service workers in the communications industry can be successfully convinced by deception or bribery to affect a SIM porting or other adjustment to deliver SMS-OTP to attackers.
i) Malicious replacement of SMS-OTP number at the website; Software or operators running the website can be tricked into changing the phone number to which codes get sent. Attacks involving combinations of social engineering against multiple third parties exist which provide an adversary direct access to change the SMS-OTP phone number themselves online.
j) Mobile Malware; iOS and Android operating systems both include a "permissions" setting which permits Mobile-Apps to read and interfere with SMS. Malicious apps exist which forward SMS to attackers and hide their display to the user.
k) Third party trust; The SMS-OTP itself travels through many different networks before reaching the user; any breakdown of trust along the way affords malicious opportunity.
l) Most OTP Hardware vulnerabilities also apply to SMS-OTP; Including: MitM; no channel security; spoofing; single channel transport; keyloggers, phishing, malware, social engineering; no utility for signing transactions; no malware protection (distinct from mobile malware), low resistance to misuse by friends, family, or peers; intentional fraud; no non-repudiation; no mutual auth; (full descriptions in subsection A-1.1)

Drawbacks of SMS OTP include:

m) SIM Change; SMS-OTP stops working when users change phone numbers. This is common for international travelers.
n) Unreliable delivery; SMS message delivery is often delayed or fails (a significant problem since OTP codes expire quickly).
o) No offline usage; SMS will never arrive unless a user has a valid connected and paid-up cellular account.
p) Poor coverage; Many places exist with no cellular coverage.
q) Usability: SMS-OTP dramatically slows all logins; this can be minutes or more in on poor cellular networks.
r) SMS-OTP does not scale well and suffers poor portability. Imagine changing your phone number on 1000 accounts.
s) Prevents Fast / Automatic logins; Waiting for and typing-in an SMS-OTP makes fast and/or automated logins impossible.
t) No secure self-service replacement; Lost phones (or non-working SMS delivery of any kind) require operator-assisted bypass. Phones often get lost, so help-desks become used to allowing users to bypass SMS-OTP. Spotting malicious users in the flood of legitimate bypasses is difficult.
u) Expensive support and losses; help desks are needed to handle customer SMS-OTP bypass. Fraud teams and products are needed to mitigate attacks overcoming SMS-OTP protection.
v) High costs; Sending SMS with reliably delivery costs more.
w) Banned; NIST 800-63B says not to use SMS, and that it will be banned in future. Many telcos have said this for years.

A-1.6 In-Device biometrics

Broadly speaking, there are two types of biometrics:-

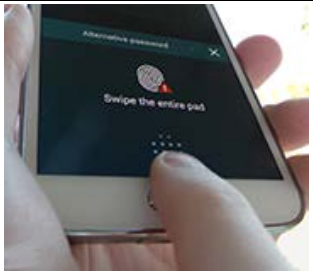

Figure 17. In-Device biometrics

(1) In-Device, which typically make use of secure hardware within a device to record and later compare user biometric features, but never send biometric features or scans over networks, and
(2) Remote biometrics, where the user biometric (e.g. their voice) is sent to a remote machine for processing. In-Device are considered "secure", since considerable effort is typically applied by the manufacturer to prevent theft and feature extraction. Remote biometrics are considered extremely dangerous, since raw biometrics data is subject to theft both in transit and at rest. Because biometrics can never be changed once compromised, many jurisdictions and countries completely ban the transmission and/or storage of biometric data through networks for all or part (e.g. just children) of their population.

Security vulnerabilities of In-Device biometrics include:-

a) Not all phone manufacturers implement biometrics technologies well. Some create purpose-built secure enclaves for biometric processing & offer well designed API interfaces, others do none of that. One popular platform SDK includes a key-enumeration API; any app can extract every fingerprint key from the phone. It also has no biometric cryptography API at all; developers have no option but to write insecure code.
b) All biometrics reduce overall user security, because they all offer PIN or password bypass for situations where user biometrics fail (e.g. fingerprints after swimming or rough manual labor). An adversary now has 2 different ways to compromise protection; steal a fingerprint or guess a password.


Figure 18. Why adding extra security makes things weaker.

Some argue that passwords become stronger since they're used less, and thus harder to observe, however, adversaries with that level of access can engineer password-theft scenarios (e.g. fail a fingerprint several times to force the user to enter their code)

c) False vendor claims; The world's strongest and most advanced (for those who recall vendor advertising at the time) fingerprint biometrics with subdermal imaging and secure enclave was hacked less than 48 hours after release using a laser printer and

wood glue. Marketing messages were posthumously amended, the vendor claiming they meant "more secure because more people will use it instead of leave their phones unlocked" (which is true), despite the fact it reduced security for their customers already using passcodes, who opted in.
Most biometrics use extracted features and approximation to calculate probabilities of match, making them unsuitable for hashing-technique protection, yet many vendors make clearly untrue "completely safe against theft" claims on these grounds.

d) Low entropy (depending on the type of biometric and sensors); biometric efficacy is a tradeoff between false negatives and positives; mimicry can defeat voiceprints 33% of the time[8].
e) Easily stolen keys; A fingerprint protected mobile phone will spend almost all its life covered in legitimate user fingerprints.
f) Easily copied; Custom silicone finger-caps (e.g. to defeat shift-work timeclocks) made to copy any prints you supply cost $20.
g) Unchangeable keys; there is no recovery after theft.
h) Widely collected keys; Travelers, criminals, and voters routinely provide fingerprints. Many of these collections are shared or have been hacked and stolen (or will be in future).
i) Vulnerable to failures in unrelated systems; Biometrics stolen online may be useable to defeat those used in-device.

Drawbacks of In-Device biometrics include:-

j) False negatives; biometrics often don't work. (refer Figure 18).
k) Environmental reliance; some biometrics rely on the conditions of collection. Face-recognition often fails at night time.
l) Backups; In-Device biometrics are not useful for protecting remote resources (e.g. cloud storage).
m) Portability. Complete re-enrollment is needed on new devices.

A-1.7 Biometrics collected remotely

These are the worst and most reckless form of security: refer explanation at A-1.6(2). They are already widely banned.

Security vulnerabilities of remote biometrics include:-

a) In-Device biometric vulnerabilities also apply to these.
b) Trivially vulnerable to theft during use, outside of use, from public archives and directly from stored feature databases.
c) Often transmitted in-the-clear; (e.g. most voice remote-biometrics take place over unsecured telephone networks)

Drawbacks of remote biometrics include:-

d) Illegal to use in many places and on certain people (e.g. kids).
e) Easy to steal. No way to change once stolen.
f) Dictionary attackable; not all remote-biometrics have rate-limits on guessing, and combined with the low entropy of many remote-biometrics, brute-force access is feasible.
g) Imprecise; most remote-biometrics must suffer the inadequacies of the "weakest acceptable collection device" (e.g. poor voice connections for voice).
h) Enormous negative privacy implications; biometrics facilitate automated non-consensual surveillance and tracking of subjects in a wide and increasing range of circumstances.

A-1.8 USB Gadgets and Smartcards

These screenless devices which attach to your computer (e.g. pluggable USB keys), or attach to a reader which is itself attached to your computer (e.g. keyboard with card-reader).

Figure 19. USB OTP

Security vulnerabilities of connectable gadgets include:

a) Malware; all connectable gadgets are at full mercy of whatever infections might be present on their host machine.
b) MitM; USB OTP has 2 options: (1) defend MitM attacks (e.g. certificate-substitution), making them unusable in workplaces with DPI firewalls, or (2) accept intermediaries (and attackers).
c) Injected transactions; with no on-device screen, the signing user has no means to verify what they're signing.
d) Piggyback risks; USB memory sticks can be disguised as USB tokens, facilitating unauthorized carriage and use at work.
e) Infection vector; USB-OTP tokens are computing devices; programmable to infect host computers. USB attacks like hardware keyloggers, PC wifi bugs, and DMA-memory-theft bootloaders can also be disguised to look like USB-OTP.
f) Increased social-engineering risks; plausible bypass excuses exist (e.g. tokens left at home, not carried on vacation, etc) making it hard for help-to desks recognize intruders.

Drawbacks of connectable gadgets include:

g) Limited compatibility; there are many different kinds of plugs used across phones and PCs, like USB-A, USB-B, Micro-USB, Mini-USB, USB-C, iPhone 30pin, lightening and whatever-comes-next. No USB-OTP supports all these. Users with multiple devices, or who change devices, or don't have slots on their device may find their USB-OTP will no longer connect.
h) Workplace bans; security conscious organizations do not allow the use or connection of USB devices.
i) Storage security; Workplaces that do allow USB often prohibit the transport of USB devices into or out of the workplace, forcing employees to leave them unattended after hours.
j) Difficult to scale; different devices, vendors, and standards are incompatible. Multiple different USB-OTP's will be needed to protect many accounts, each one suitable for only a small subset, leaving it for the user to remember which-is-for-what.
k) Single-device only; USB-OTP works only with one device at a time usually; there is no way to have a spare for emergencies.
l) Inconvenience; carrying devices everywhere so you can login when you need also raises the risk of USB-OTP loss or theft.

A-1.9 Client TLS certificates

Most browsers natively support X.509 client certificates. So does other software, and custom applications exist also making use of similar Public Key Infrastructure (PKI).

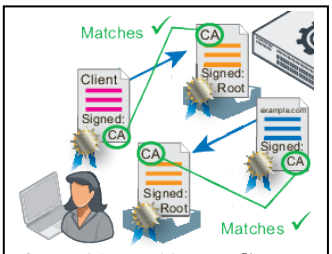

Figure 20. X.509 PKI diagram

Vulnerabilities include:-

a) Certificate compromise; client certificates are stealable computer files. They have passwords, but can be brute-force and dictionary-attacked

attacked offline, or passwords stolen.

b) Malware; PKI offers no protection against malware.
c) CA Compromise; Certificate Authorities issuing client certificates can and have be compromised.
d) Checking certificate legitimacy is difficult, (impossible on some devices). Users rarely verify certificates or legitimacy.

Drawbacks of PKI include:

e) Usability; PKI is one of the least useable 2FA methods. It requires highly competent users. Enrollment, use, and renewal are challenging. Implementation is radically different across devices and vendors, and frequently changes with upgrades.
f) Compatibility; There are many PKI compatibility differences, file types, encoding formats, ciphers and digests. Only a fraction those work in any particular O/S and software.
g) Expiry; certificate lifetime is usually short, (typically one year, or much less for trial certificates). Users must re-endure the challenging reissuance process often. Old certificates must still be kept for future signature checking, and these make ongoing usage even worse (user need to select their current login certificate, named identically to all their expired ones).
h) Cost; Most client PKI requires payment, often high, to a Certificate Authority (CA), usually annually.
i) CA Revocation; this invalidates all user certificates at once.
j) Portability; Certificate re-use is possible across many devices, but the steps needed to make this work are extremely complex.

A-1.10 Paper lists (TAN)

Transactional Access Numbers (TAN) are codes typically printed in a grid requiring users to locate via some index number or a row and column id the OTP code to use. Some are single-use only.

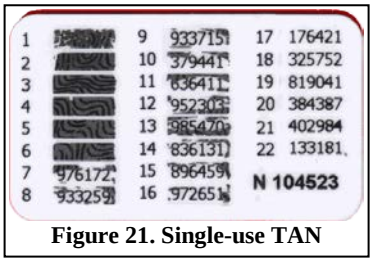

Figure 21. Single-use TAN

Security vulnerabilities of TAN include:

a) TAN's suffer the same vulnerabilities as OTP hardware listed in section A-1.1(a) through (m).
b) TAN Pharming is an attack technique which tricks users into revealing TAN codes to an attacker, who is then free to use them in subsequent attacks. They are facilitated by the predictability of the TAN index (e.g. a TAN card with rows and columns will always have a TAN code at location A1).
c) A server needing to verify TAN correctness necessarily holds sufficient information to do this, which is then susceptible to theft (and offline dictionary attack if necessary); one server-side break-in can invalidate all issued TANs at once.

Drawbacks of TAN include:

d) Do not scale; If activated on thousands of accounts, a user would need thousands of individual TAN lists or cards.
e) Physical replacement issues; If used regularly, expiring TANs would require frequent replacement, and reusable TANs would become reconstructable to eavesdroppers.

A-1.11 Scope failures across all 2FA (and non-2FA)

Within every category, many vendors & products exist, each with their own and differing shortcomings (not covered in this paper). The broadest shortcoming across all 2FA categories (and indeed, most non-2FA alternatives as well) is "scope". Most vendors push responsibility for "difficult" security problems to their customers.

A-1.11.1 Reliable initial user identification

The intersection between identity and authentication is hard to secure; so much so that all 2FA technologies chose not to address this problem. This leaves a gap between the identification of the new user, and their enrollment in 2FA. All 2FA categories leave opportunity for intermediaries to hijack or subvert the deployment process. Many providers mix deployment with verification such as by physically shipping devices, keys, unlock codes, and TANs in postal mail, or by using SMS, phone, or email to deliver PINs or enrollment keys. All those shipping measures are unreliable, offering interception, substitution, and facilitating a range of social-engineering opportunities against both users and staff alike. They also require soliciting personal address information from users. Google, during the 2011 AISA National Conference, revealed the single biggest issue preventing uptake of their SMS-OTP product was user reluctance to provide their phone number.

A-1.11.2 Enrolment across compromised channels

2FA is deployed because risk is identified among users, so it's clearly an oversight to ignore this risk during the 2FA enrollment.

Assuming a user took delivery of their 2FA solution without incident, none offers satisfactory protection to prevent the attacker (1) either stealing the 2FA for themselves, (2) tricking the 2FA into enrolling the attacker instead of the user, or (3) downgrading the protection or preventing and/or spoofing enrollment entirely.

A-1.11.3 Loss handling

All 2FA is subject to loss or destruction, or dependent on secrets that users might forget, particularly the elderly, & especially when 2FA is used infrequently. Some 2FA is version dependent, and fails when updates take place (for example; Java) or machines change (e.g. pluggable USB devices when users switch to an iPad), or after certain intervals of time or when batteries go flat.

2FA bypass is an often-exploited shortcoming across all 2FA categories. It is the fault of the 2FA leaving loss-handling outside the scope of protection which caused this problem. Each deployment requires its own re-enrolment procedure, and most make use of fallback/recovery mechanisms that do not use 2FA.

A-1.11.4 Social engineering of staff

For all users who cannot log in with their 2FA for any reason (e.g. section A-1.11.3), some method of bypass is introduced. Support staff with access to change or remove 2FA is one common method. Since these staff are so accustomed to dealing with average legitimate users and everyday problems, it becomes very difficult for them to detect an account takeover attack being performed by a social engineer. Many headline news stories of high-profile 2FA-bypass account takeovers and online banking thefts facilitated through 2FA-bypass have been published.